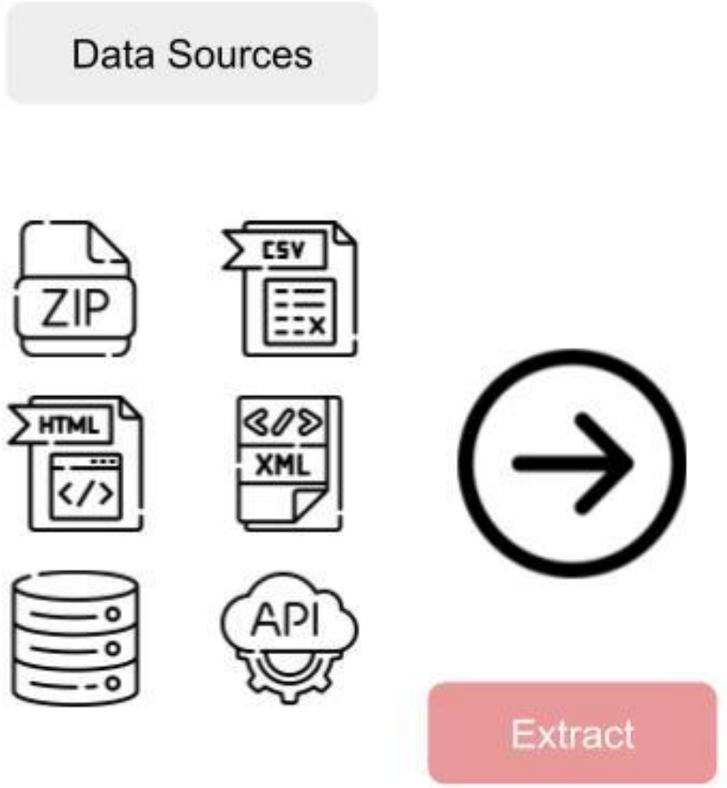
Introduction to ETL in Python

ETL IN PYTHON

Stefano Francavilla
CEO - Geowox

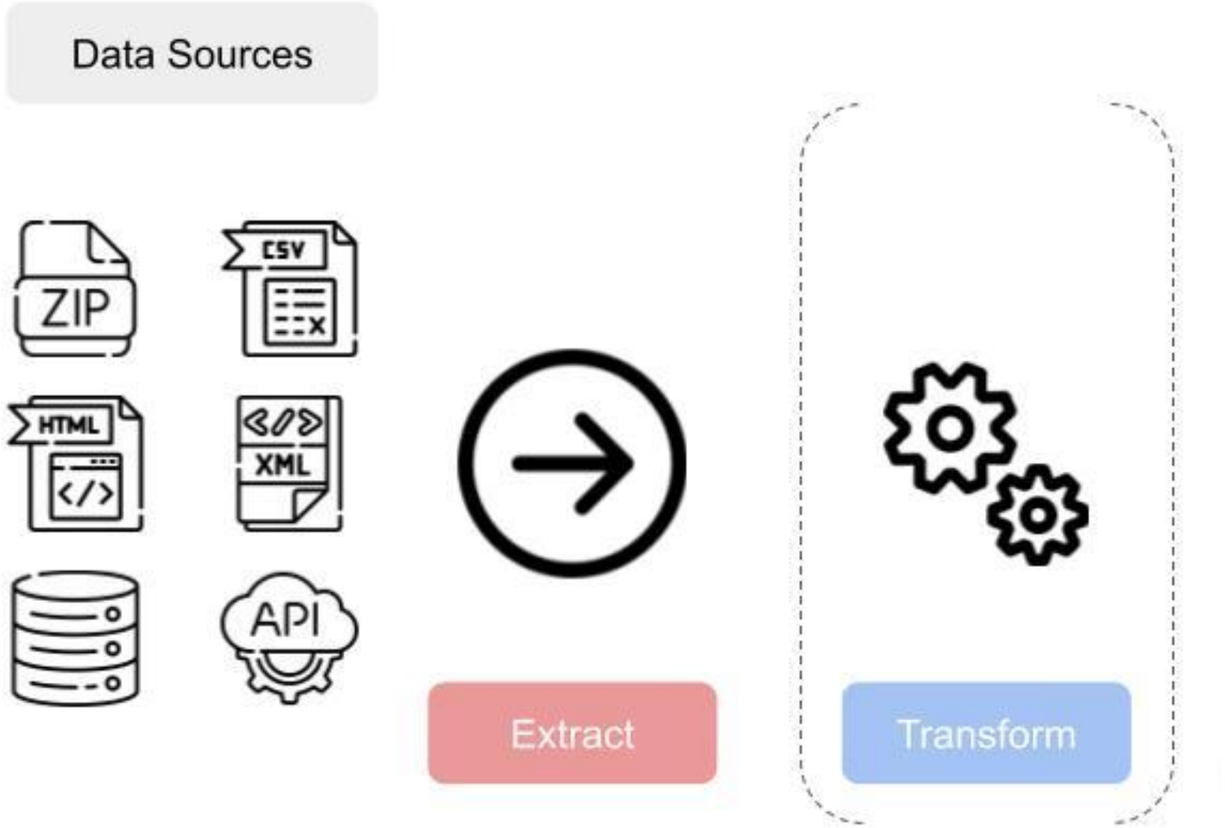
What is ETL?

Extract, Transform, Load



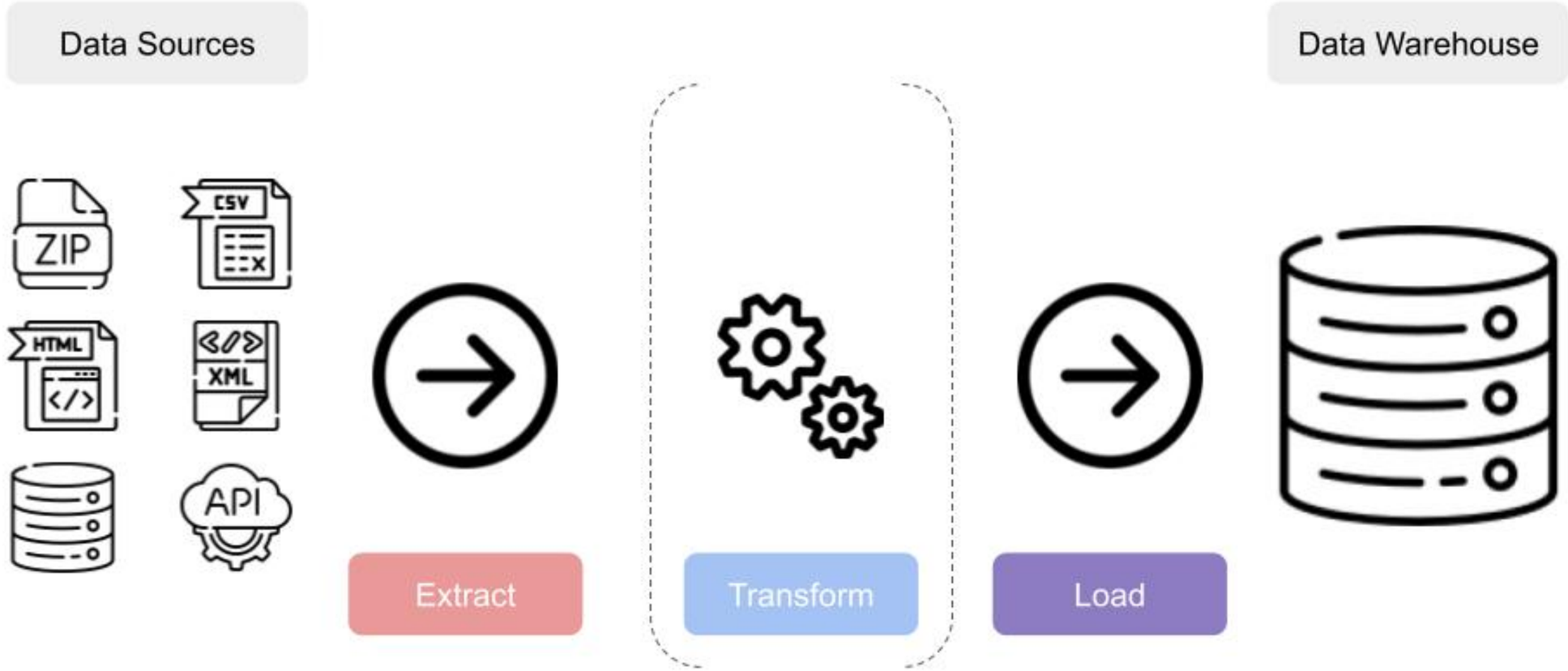
What is ETL?

Extract, Transform, Load



What is ETL?

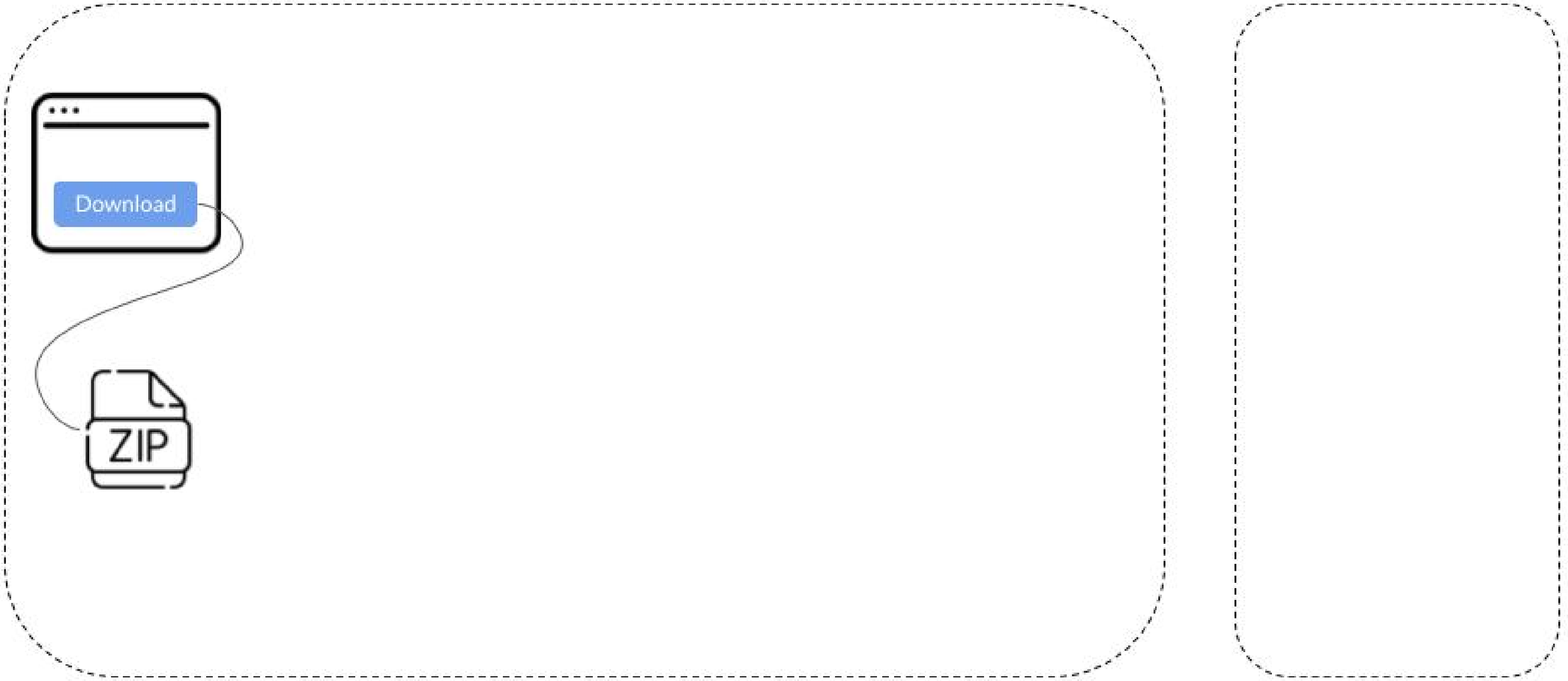
Extract, Transform, Load



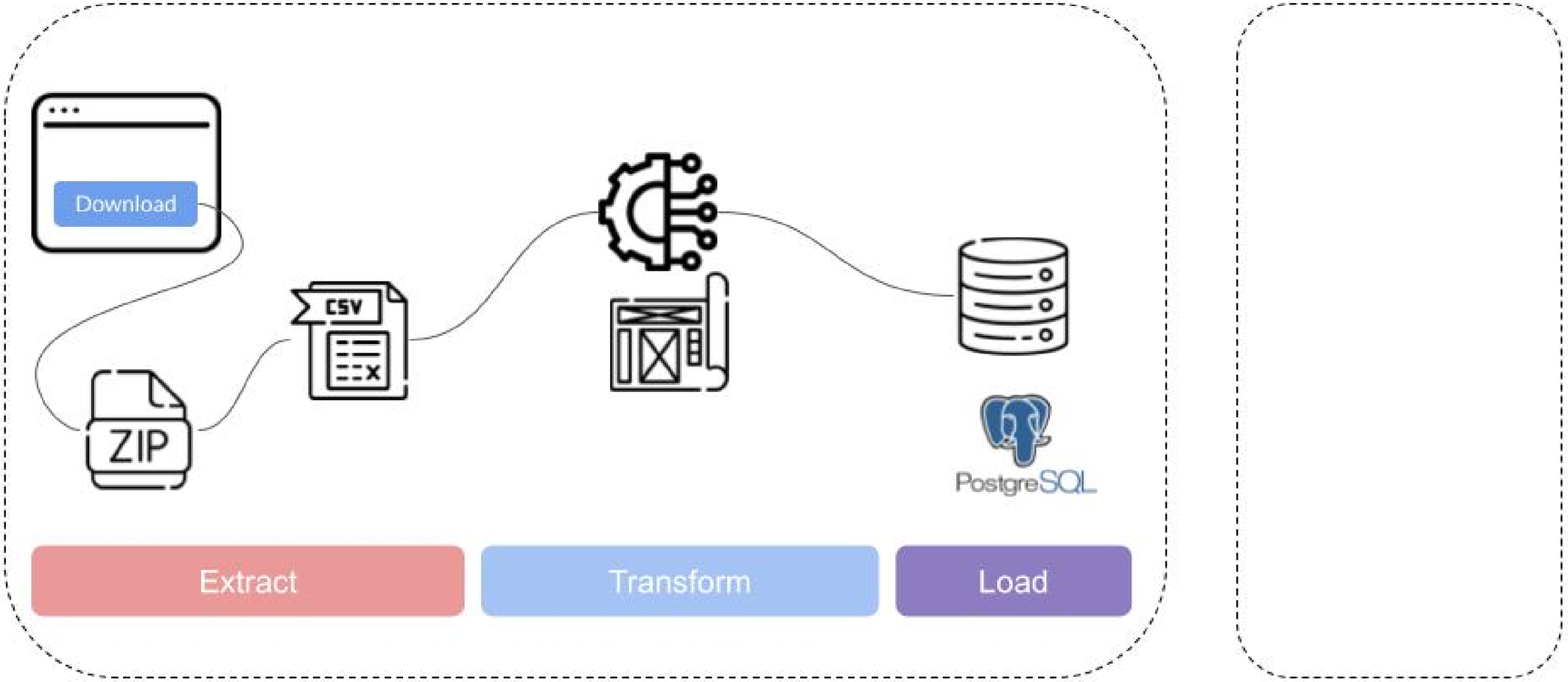
The scene

- **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**

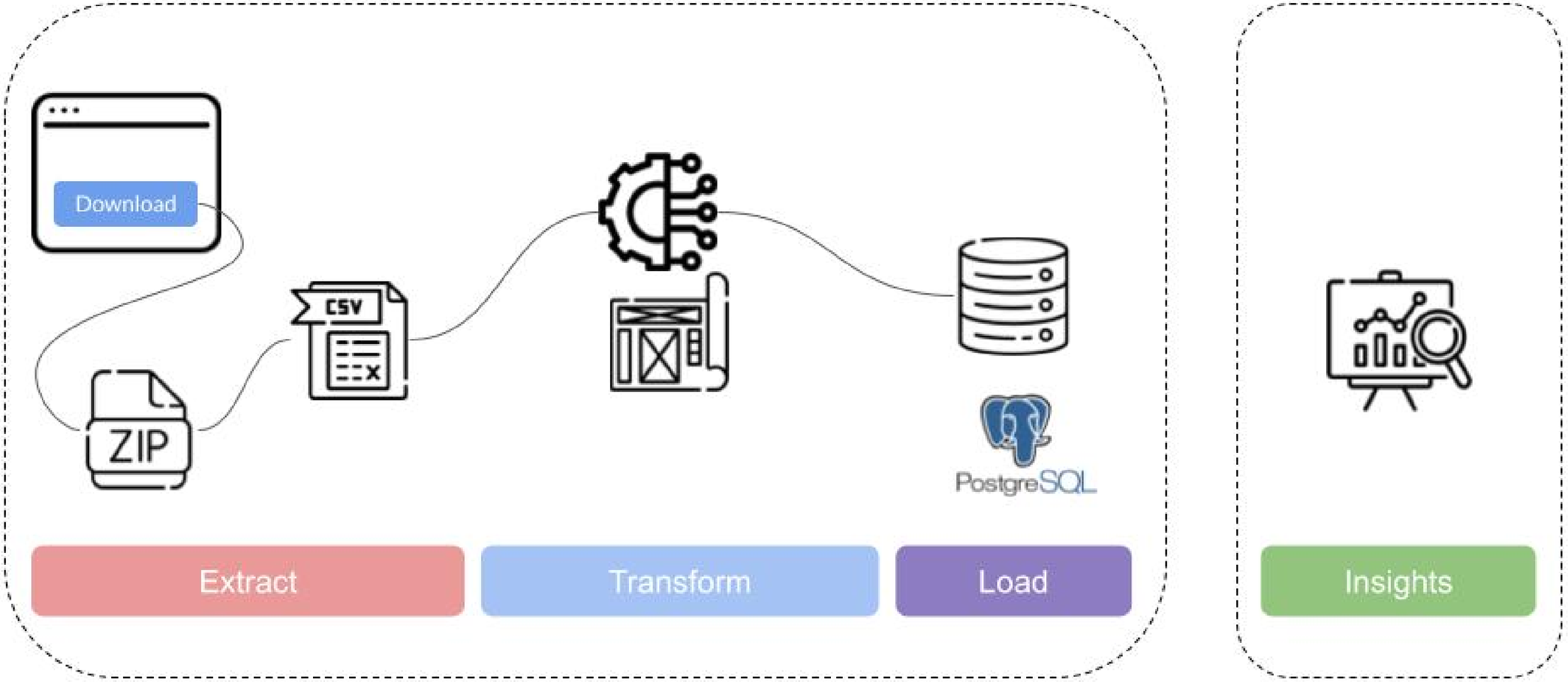
The pipeline



The pipeline

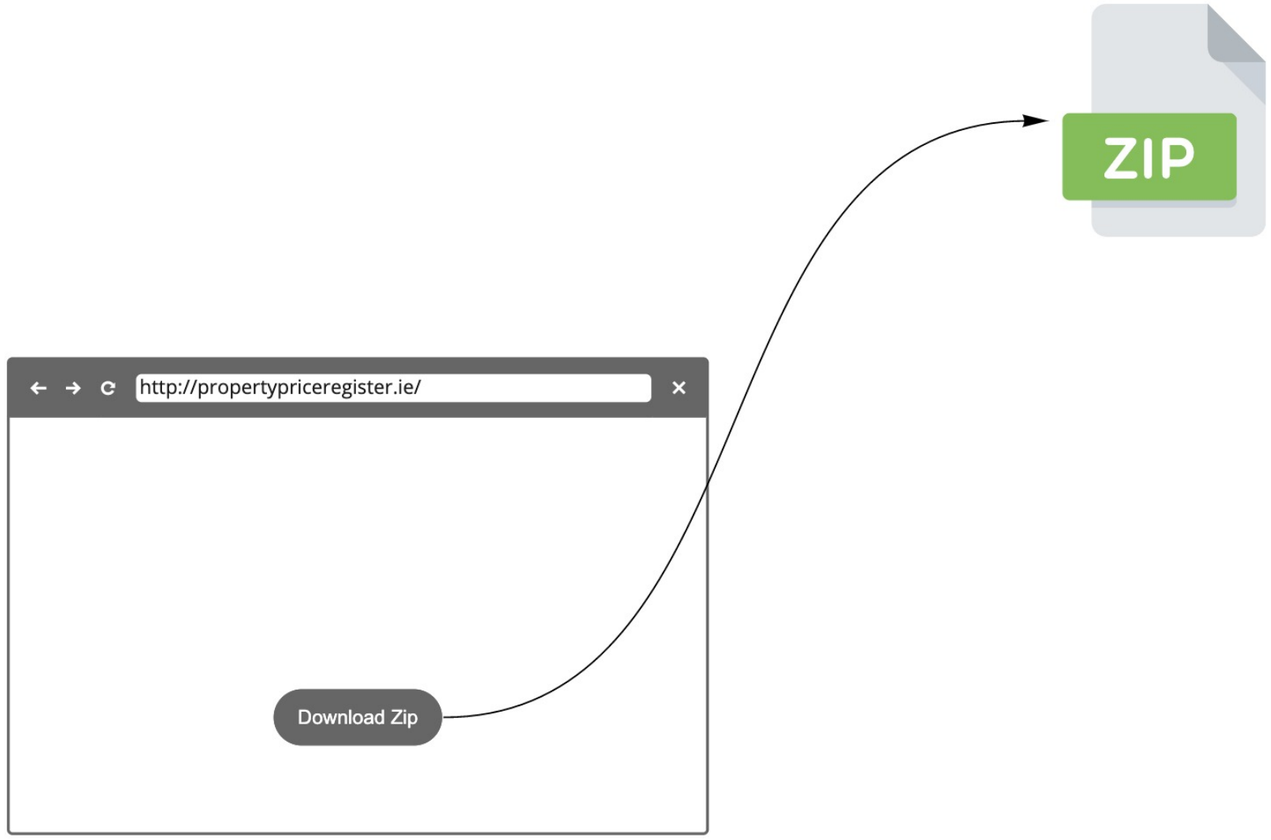


The pipeline

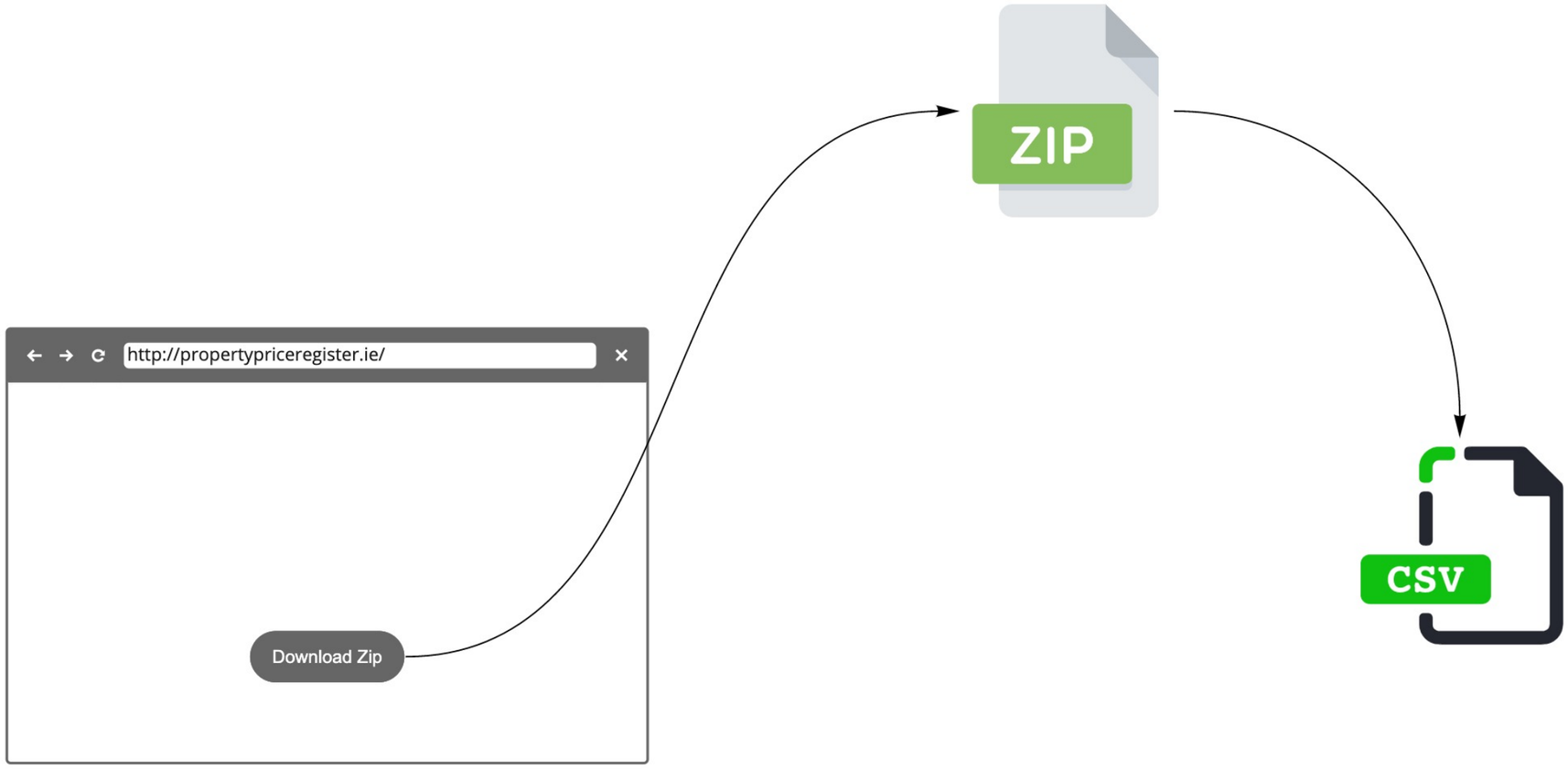


In this lesson

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>')`

```
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'})`

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

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

```
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!

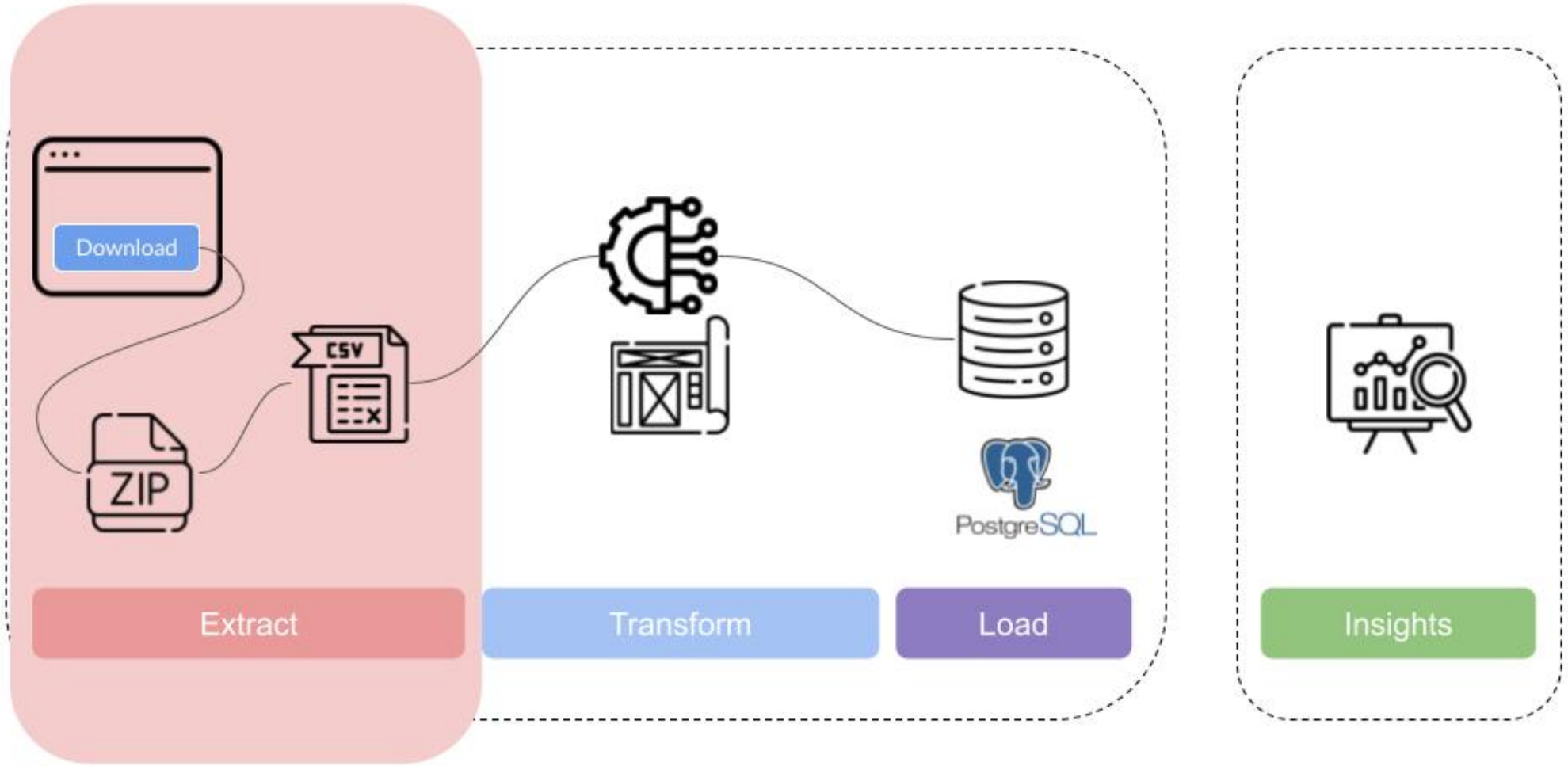
ETL IN PYTHON

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

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

Write mode

```
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

```
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

```
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

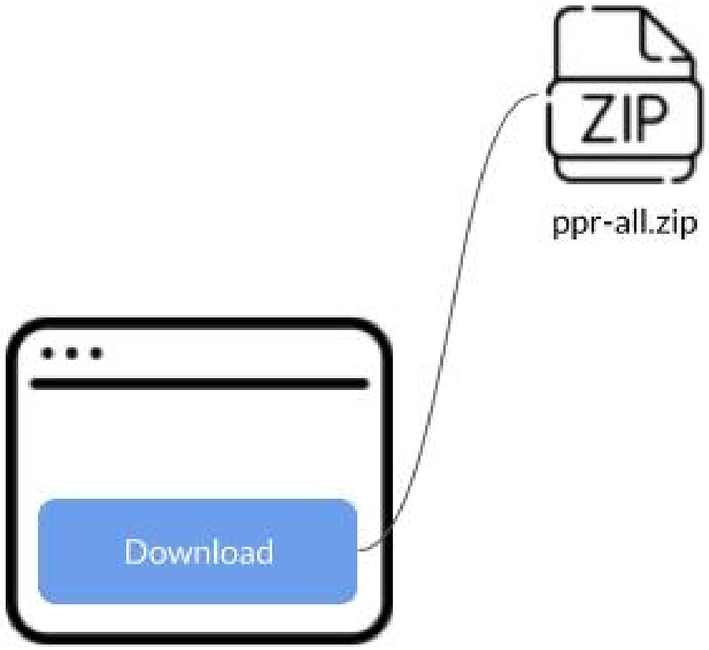
End goal

Automated pipeline

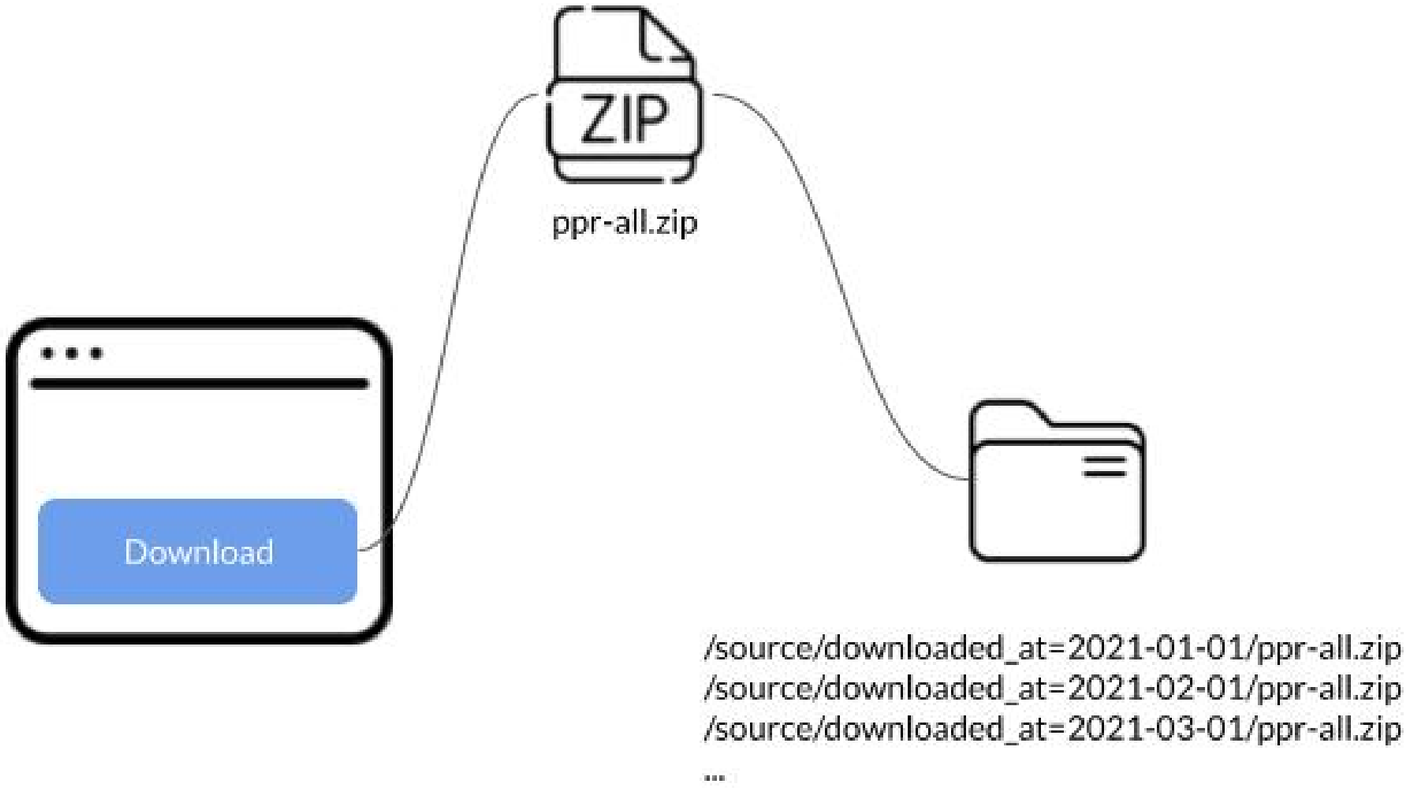
- `cron`
  - Command line utility used for scheduling
- `execute.py`
  1. `extract.py`
  2. `transform.py`
  3. `load.py`
- Download and process property transactions

E(xtract)TL

E(xtract)TL



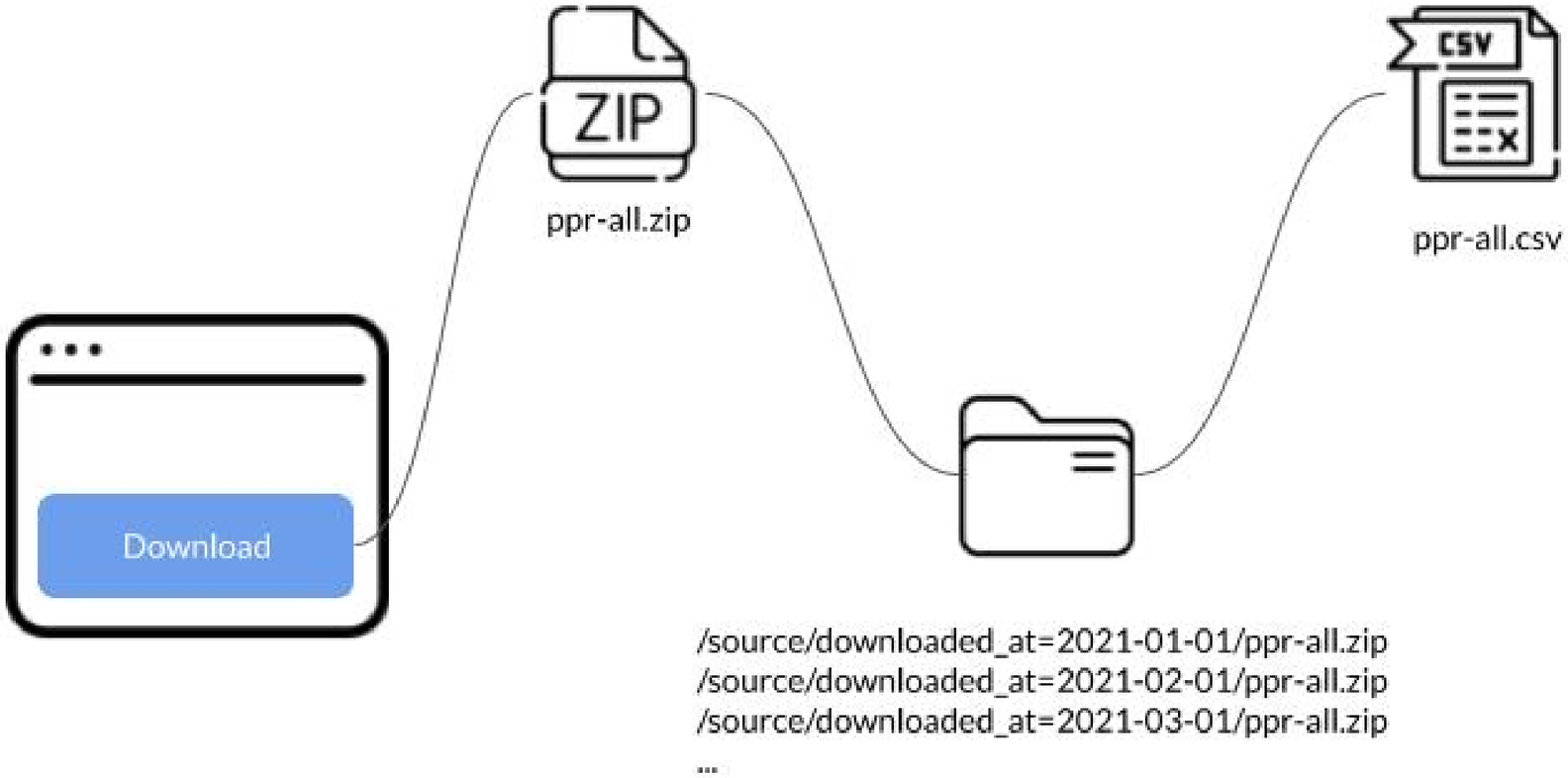
E(xtract)TL


ZIP
ppr-all.zip
Download
/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
...

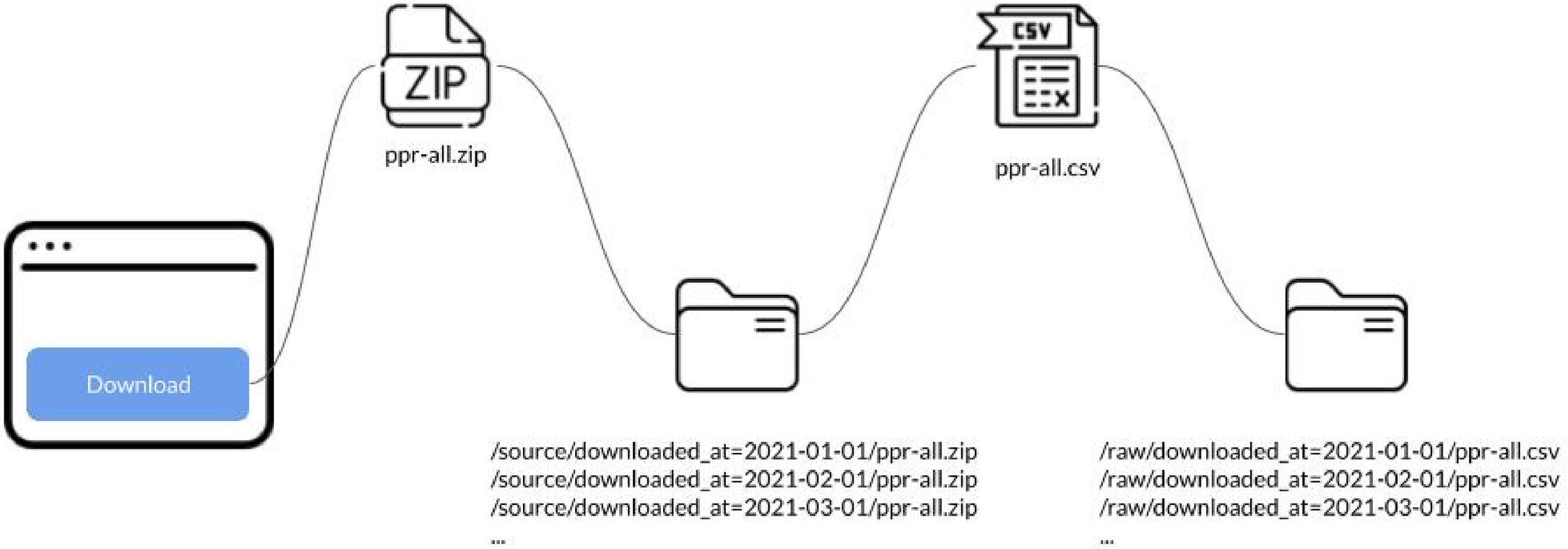
E(xtract)TL


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


ZIP
ppr-all.zip
CSV
ppr-all.csv
Download
/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
...

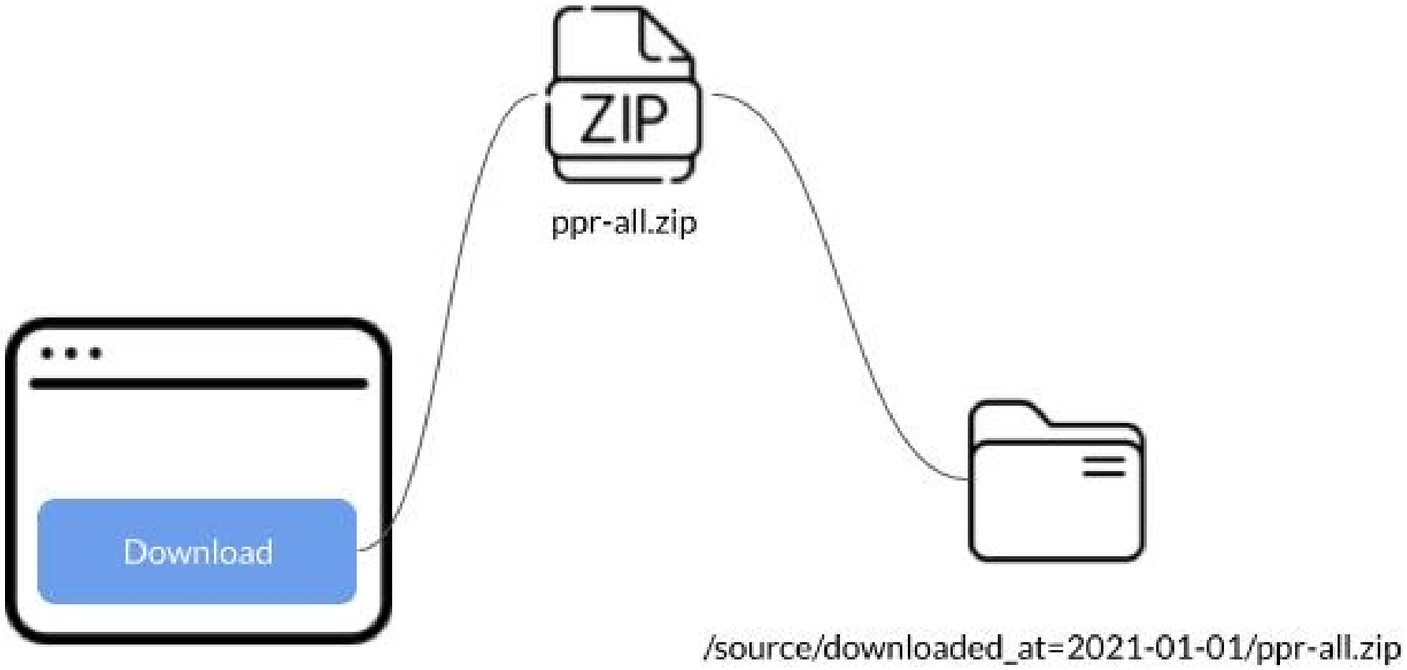
E(xtract)TL


ZIP
ppr-all.zip
CSV
ppr-all.csv
Download
/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


/raw/downloaded_at=YYYY-MM-01/ppr-all.csv
Download
/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!

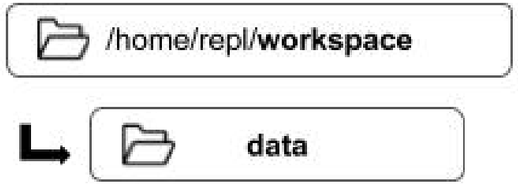
ETL IN PYTHON

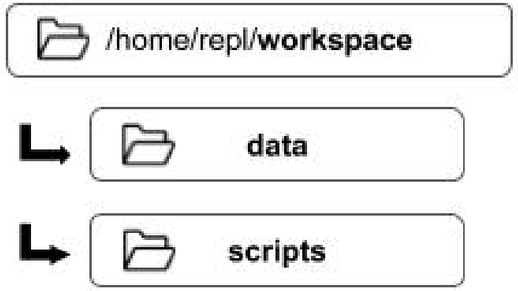
Project folder structure

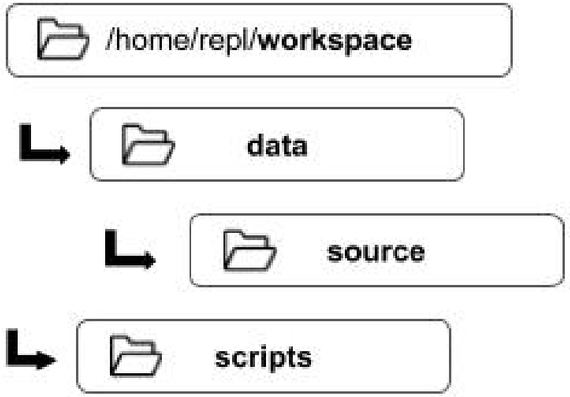
ETL IN PYTHON

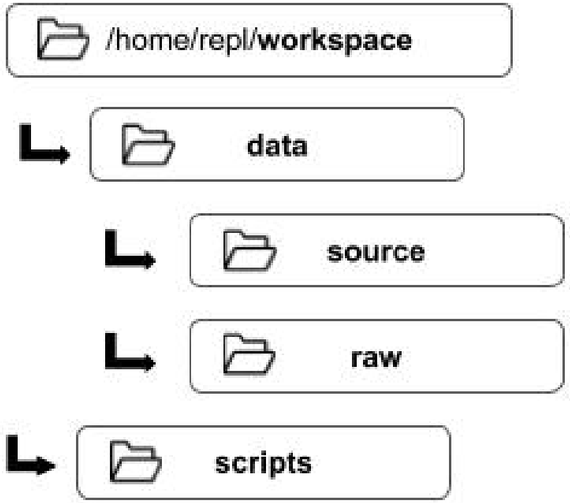
Stefano Francavilla
CEO Geowox

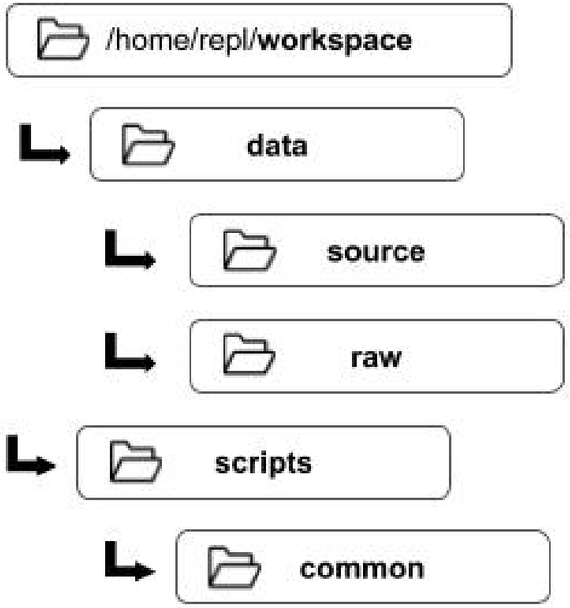

/home/repl/workspace

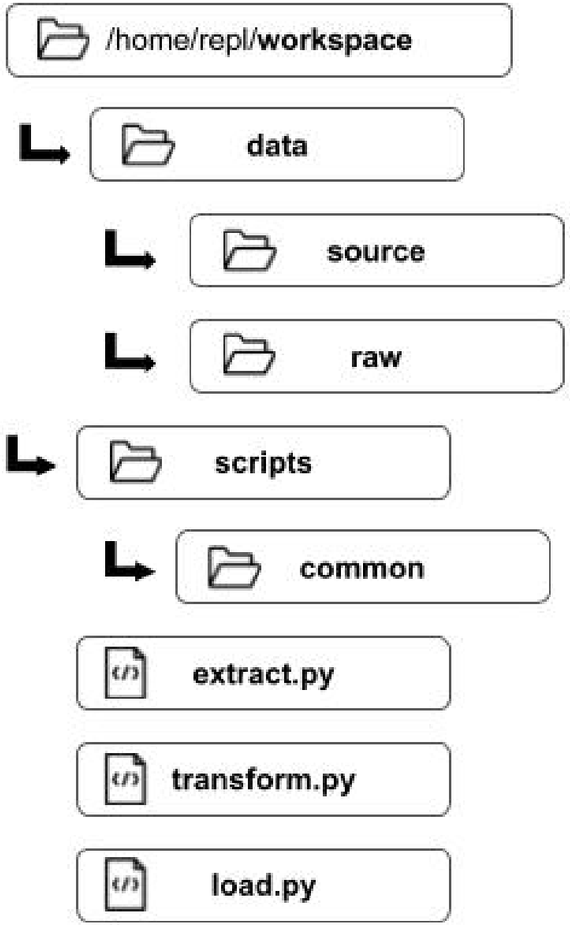

/home/repl/workspace
data

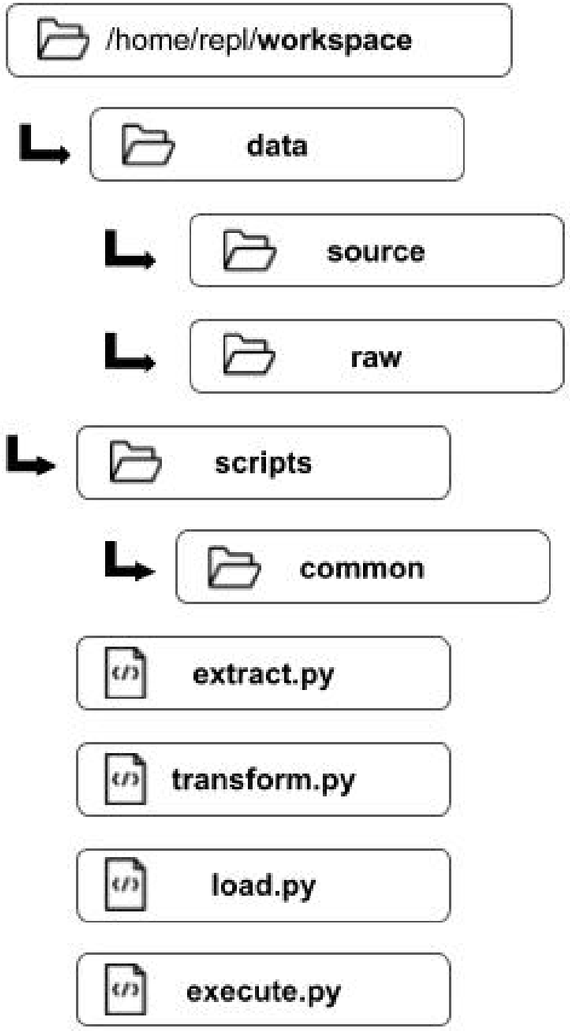

/home/repl/workspace
data
scripts

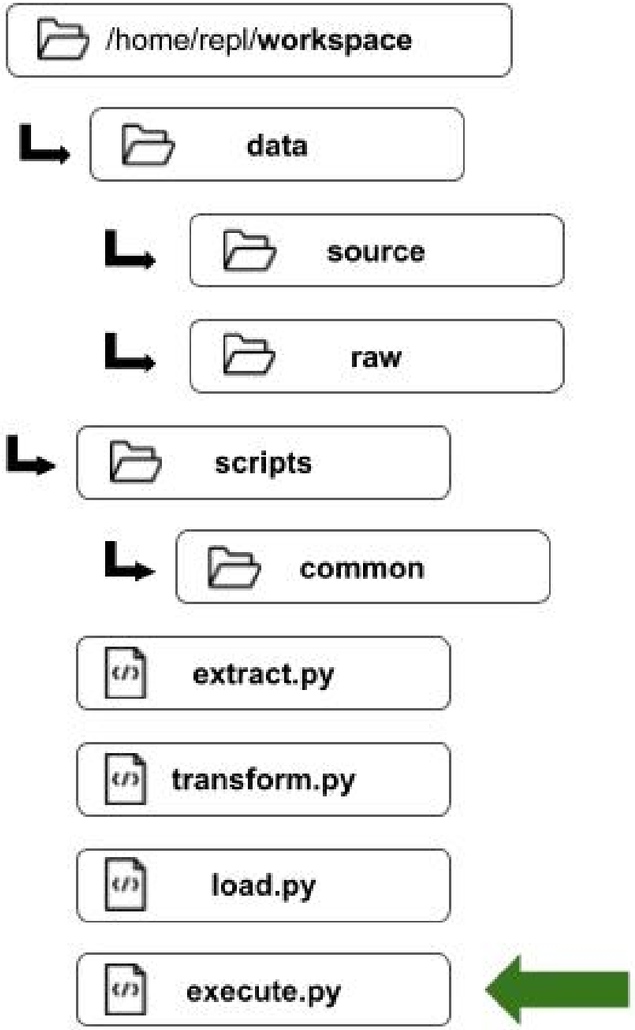

/home/repl/workspace
data
source
scripts

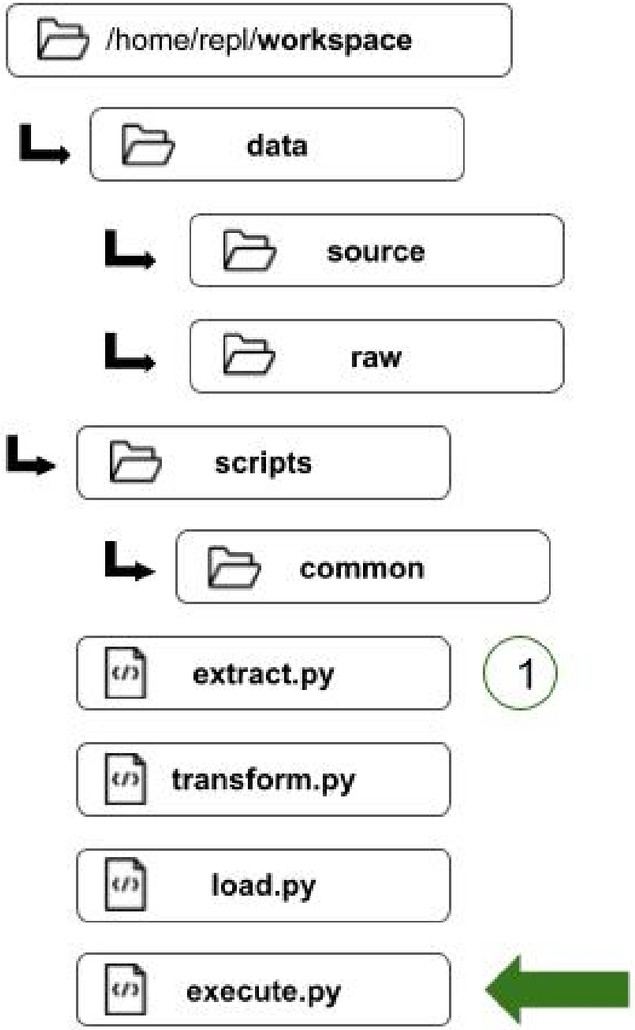

/home/repl/workspace
data
source
raw
scripts

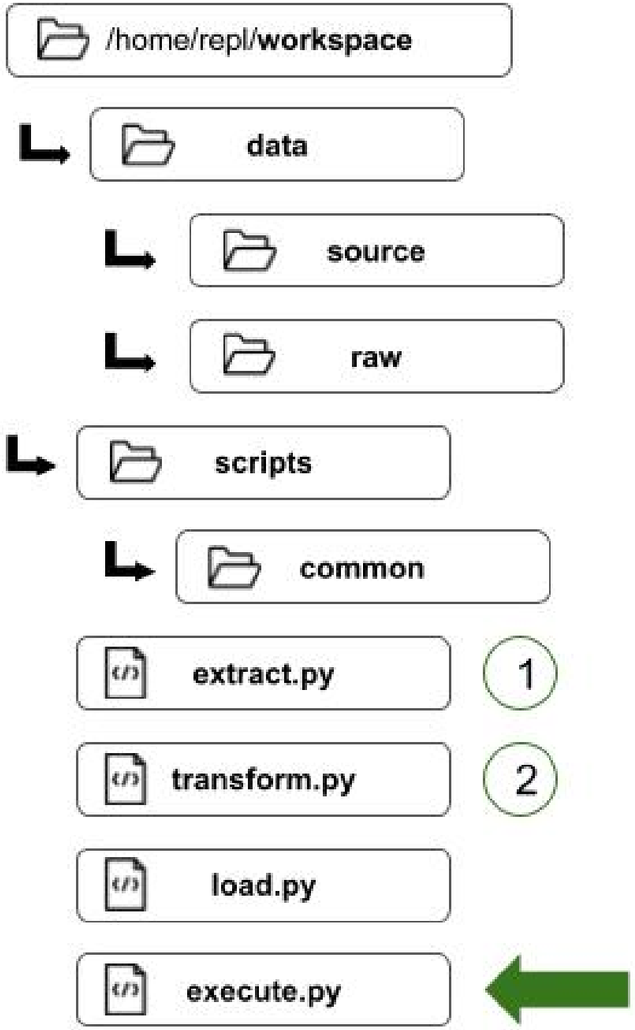

/home/repl/workspace
data
source
raw
scripts
common

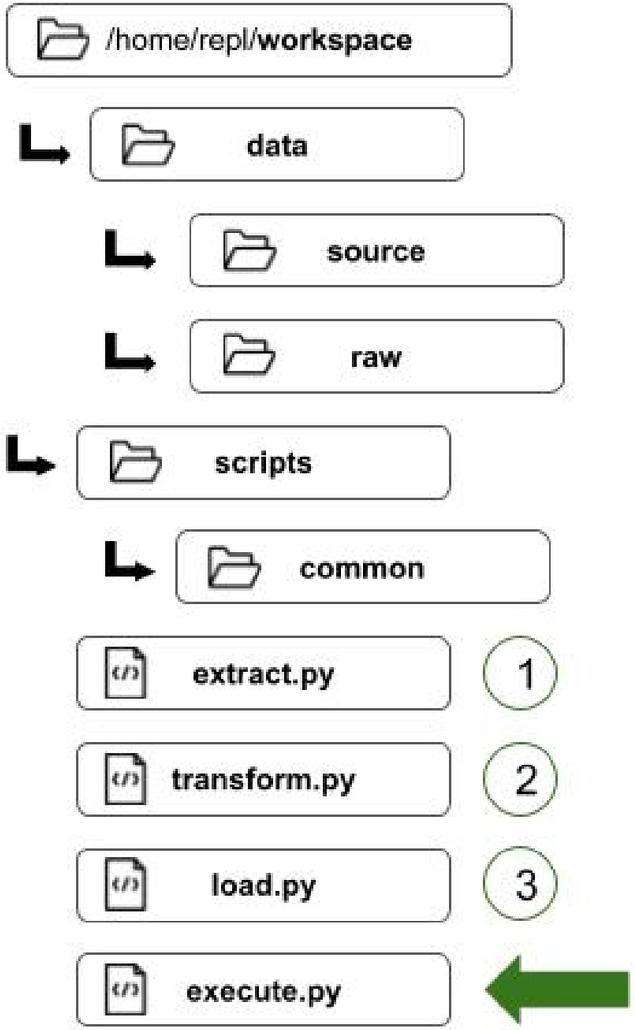

/home/repl/workspace
data
source
raw
scripts
common
extract.py
transform.py
load.py

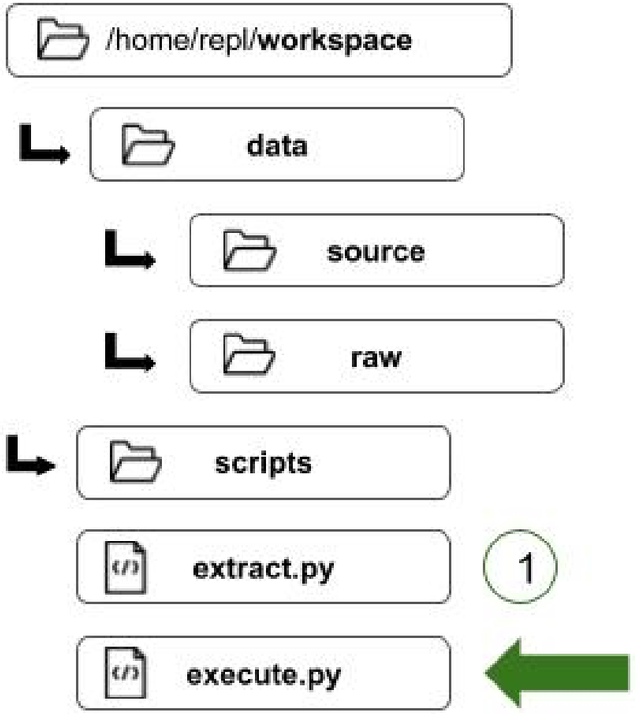

/home/repl/workspace
data
source
raw
scripts
common
extract.py
transform.py
load.py
execute.py

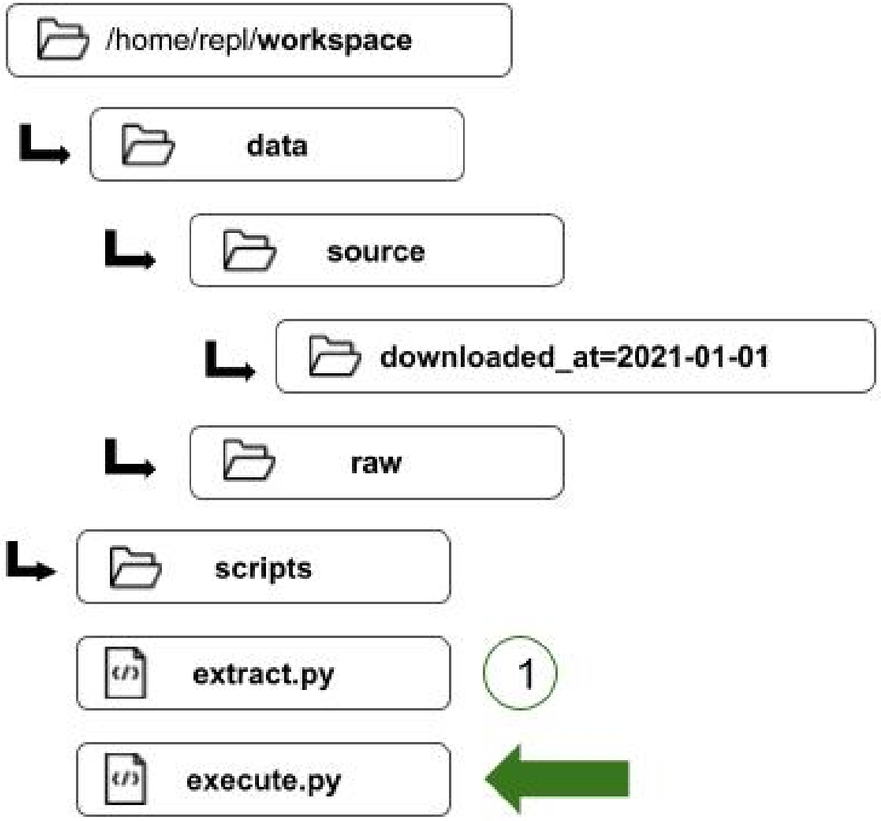

/home/repl/workspace
data
source
raw
scripts
common
extract.py
transform.py
load.py
execute.py

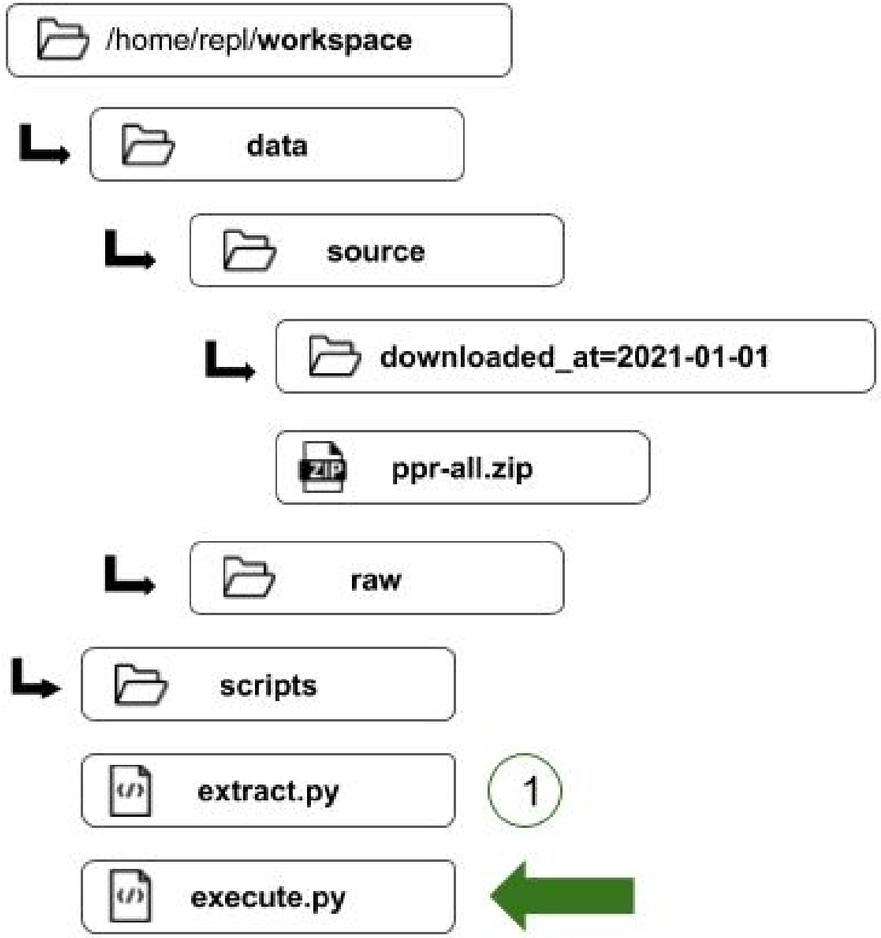

/home/repl/workspace
data
source
raw
scripts
common
extract.py
1
transform.py
load.py
execute.py

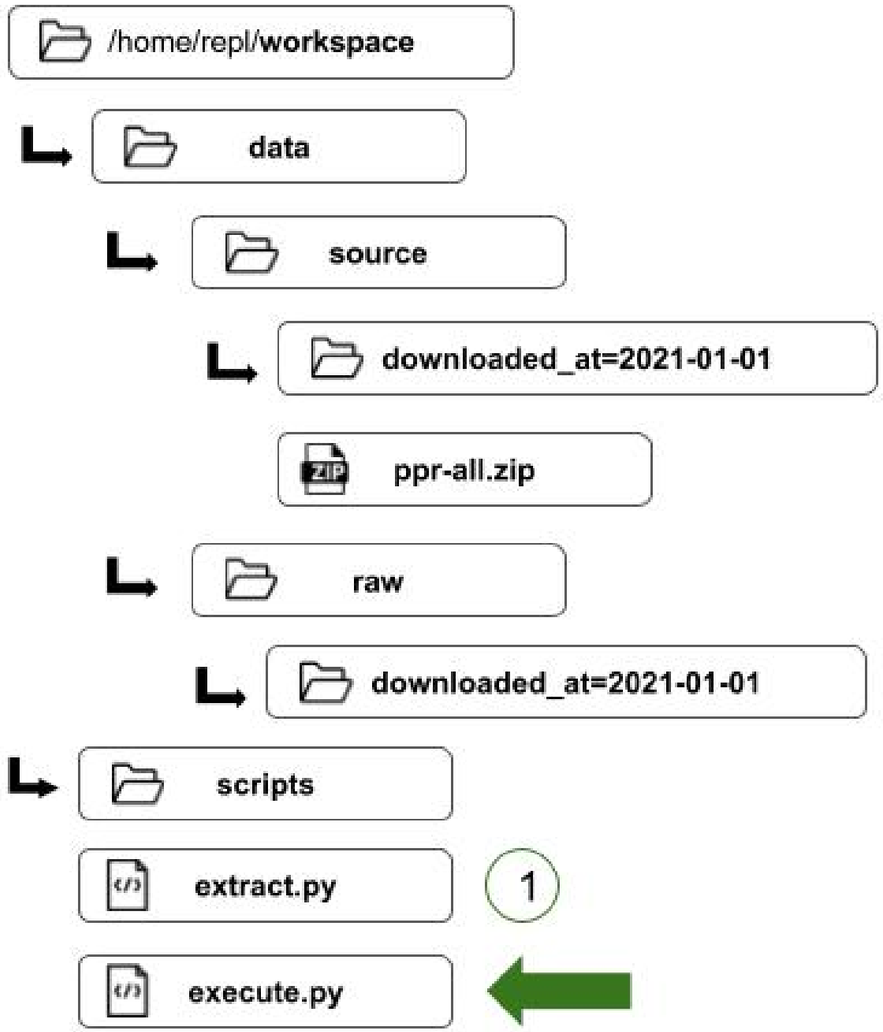

/home/repl/workspace
data
source
raw
scripts
common
extract.py
1
transform.py
2
load.py
execute.py

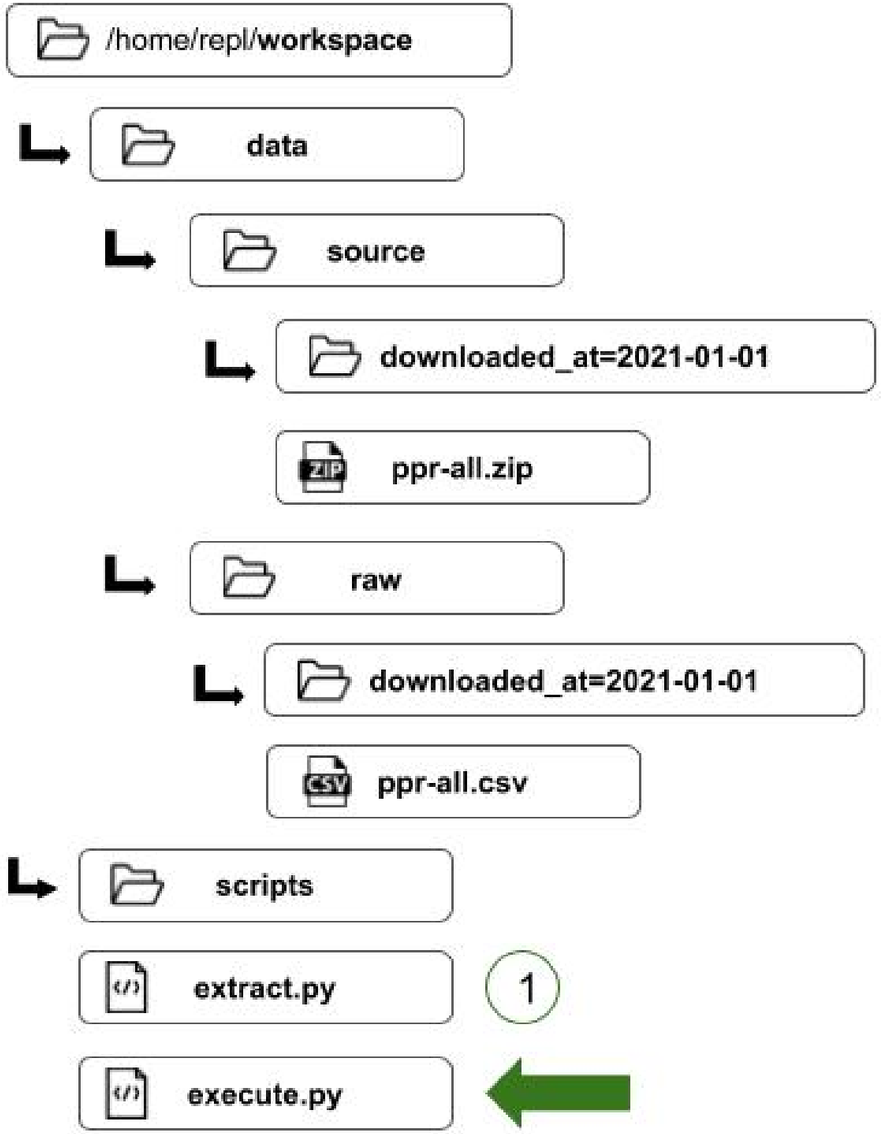

/home/repl/workspace
data
source
raw
scripts
common
extract.py
1
transform.py
2
load.py
3
execute.py


/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

```
# 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

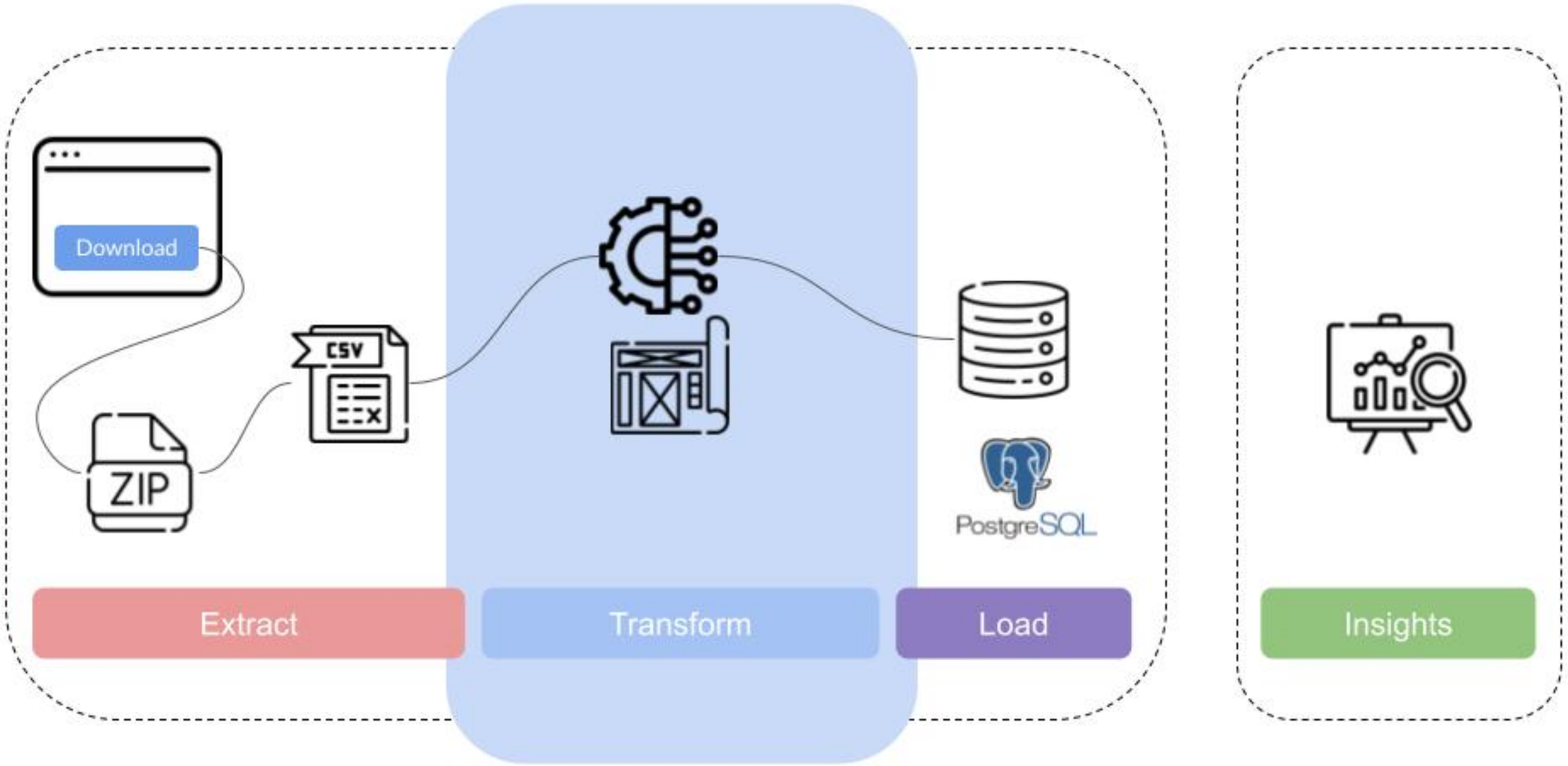
Let's talk with the database

ETL IN PYTHON

Stefano Francavilla
CEO - Geowox

datacamp

Where we are in the pipeline



What is SQLAlchemy

- **SQL toolkit** written in Python
- Object-Relational Mapper (**ORM**)
  - Translates Python **classes to tables**
  - Translates **function calls to SQL statements**
- **Supported dialects:** PostgreSQL, MySQL, SQLite and more

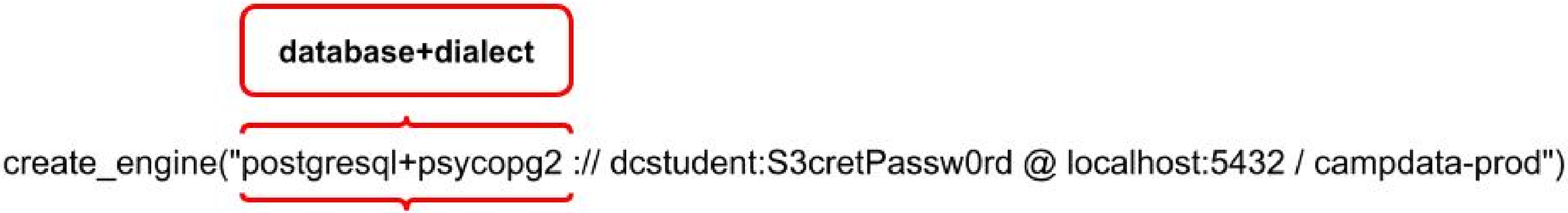
```
TableName.select(...)
```

```
SELECT * FROM TableName
```

SQLAlchemy engines

- Engine:
  - **Starting point** of SQLAlchemy applications
  - Allows **interaction** with the database
- `from sqlalchemy import create_engine`
- `create_engine()`

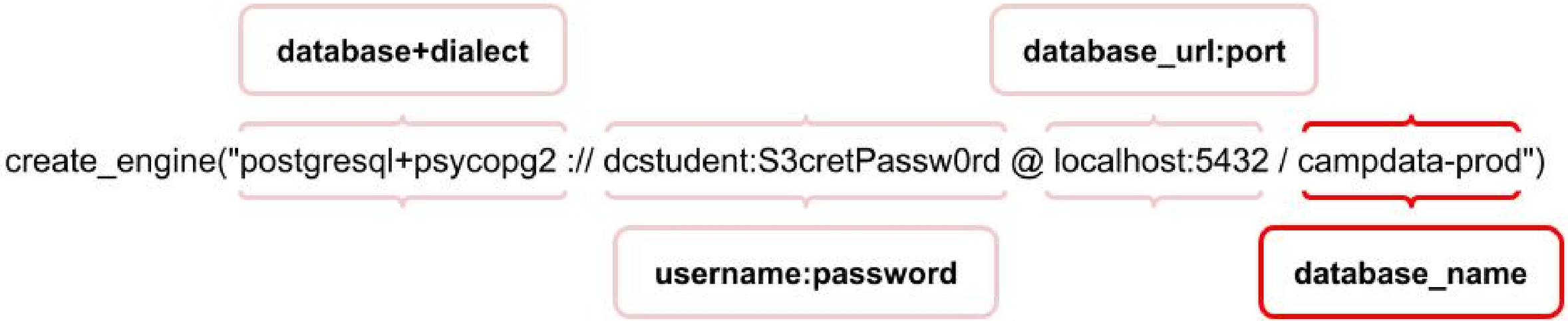
SQLAlchemy engines



SQLAlchemy engines

SQLAlchemy engines

SQLAlchemy engines



SQLAlchemy sessions

- Establish **conversations** with the database
  - Holds **modifications** before committing
  - **Update** and **delete** rows, remove tables...

```
from sqlalchemy import create_engine
from sqlalchemy.orm import Session

engine = create_engine("postgresql+psycopg2://steve:a1!@localhost:5432/mydatabase")
session = Session(engine)
```

Let's practice!

ETL IN PYTHON

Database tables

ETL IN PYTHON

Stefano Francavilla
CEO - Geowox

Object-oriented programming

```
def my_function():`
    # body

class MyClass:
    # body

obj = MyClass()
```

- Classes can have their own **attributes** and **methods**
- Classes can **inherit** from other classes:
  - The class that inherits is a **child**
  - The class the child inherits **from** is a **parent**

Object-oriented programming

```
class Parent():
    parent_attr = 'I am a parent'

class Child(Parent):
    child_attr = 'I am a child'

child = Child()

print(child.child_attr, " and ", child.parent_attr)
```

```
I am a child and I am a parent
```

Declarative base class: an example

- **Associate** user-defined **Python classes**, called data models, **with database tables**

```
from sqlalchemy.orm import declarative_base
```

```
Base = declarative_base()
class TableName(Base):
    __tablename__ = 'database_table_name'
```

Columns and types

Table Name: `movies`

Column name	type
id	integer
title	varchar(55)
description	varchar(55)

```
from sqlalchemy import Column,
                       Integer,
                       String
class Movies(Base):
    __tablename__ = "movies"
    id = Column(Integer)
    title = Column(String(55))
    description = Column(String(255))
```

Primary key definition

Table Name: `movies`

Column name	type
id	integer (Primary Key)
...	...

```
class Movies(Base):
    __tablename__ = "movies"
    id = Column(Integer,
                primary_key=True)
```

- **Column:**
  - argument **primary_key** set to True
  - **Series** of integers

Let's practice!

ETL IN PYTHON

Data cleaning

ETL IN PYTHON

Stefano Francavilla
CEO - Geowox

Raw CSV content

date_of_sale	address	postal_code	county	price	description
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

Lower strings

date_of_sale	address	postal_code	county	price	description
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 lsl, balbutcher lane, ballymun	dublin 11	dublin	€192,951.00	new dwelling house /apartment

Lower strings: example

- `lower()` method
- Converts all uppercase characters in a string into lowercase

```
string_to_lowercase = "This iS A TesT"
string_to_lowercase = string_to_lowercase.lower()

print("String to lowercase: ", string_to_lowercase)
```

```
"String to lowercase: this is a test"
```

Date

From

date_of_sale
2021/02/12
2021/01/04

To

date_of_sale
2021-02-12
2021-01-04

Date

date_of_sale	address	postal_code	county	price	description
2021/02/12	123 walkinstown park, walkinstown, dublin 12	dublin 12	dublin	€297,000.00	second-hand dwelling house /apartment
2021/01/04	12 oileain na cranoige.cranogue lsl, balbutcher lane, ballymun	dublin 11	dublin	€192,951.00	new dwelling house /apartment

Date

date_of_sale	address	postal_code	county	price	description
2021-02-12	123 walkinstown park, walkinstown, dublin 12	dublin 12	dublin	€297,000.00	second-hand dwelling house /apartment
2021-01-04	12 oileain na cranoige.cranogue lsl, balbutcher lane, ballymun	dublin 11	dublin	€192,951.00	new dwelling house /apartment

Date: datetime

- `from datetime import datetime`
- `strptime()` and `strftime()`
  - `strptime()` creates a **datetime object from a given date** in string format
  - `strftime()` returns a **string** representing the date and/or time **from a datetime object**

Date: example

```
datetime.strptime(date_string, format_string)
```

```
date_string = "2021/02/12"


date_object = datetime.strptime(date_string, "%Y/%m/%d")
```

```
datetime.datetime(2021, 2, 12, 0, 0)
```

```
new_date_string = date_object.strftime("%Y-%m-%d")
print(new_date_string)
```

```
2021-02-12
```

Date: common format strings

directive	meaning	example
`%d`	Day of the month as a zero-padded number	01, 02, ..., 31
`%m`	Month as a zero-padded number	01, 02, ..., 12
`%Y`	Full year as number	2021, 2018, ..., 1999

Price

From

price
€297,000.00
€192,951.00

To

price
297000
192951

Price

date_of_sale	address	postal_code	county	price	description
2021-02-12	123 walkinstown park, walkinstown, dublin 12	dublin 12	dublin	€297,000.00	second-hand dwelling house /apartment
2021-01-04	12 oileain na cranoige.cranogue lsl, balbutcher lane, ballymun	dublin 11	dublin	€192,951.00	new dwelling house /apartment

Price

date_of_sale	address	postal_code	county	price	description
2021-02-12	123 walkinstown park, walkinstown, dublin 12	dublin 12	dublin	297000	second-hand dwelling house /apartment
2021-01-04	12 oileain na cranoige.cranogue lsl, balbutcher lane, ballymun	dublin 11	dublin	192951	new dwelling house /apartment

Price

- Remove € symbol
- Remove the `,` character
- Convert the returning number to float by using `float()` function
- Convert float to integer by using `int()` function

Price: an example

- Input price "€297,000.00"
- Replace € symbol
- Remove the `,` character
- Convert the returning number to float
- Convert float to integer
- Print the result

```
price_in = "€297,000.00"
price_in = price_in.replace("€", "")
price_in = price_in.replace(",", "")
price_in = float(price_in)
price_in = int(price_in)
print("Price: ", price_in)
```

```
Price: 297000
```

Description

From

description
second-hand dwelling house /apartment
new dwelling house /apartment

To

description
second-hand
new

Description

date_of_sale	address	postal_code	county	price	description
2021-02-12	123 walkinstown park, walkinstown, dublin 12	dublin 12	dublin	297000	second-hand dwelling house /apartment
2021-01-04	12 oileain na cranoige.cranogue lsl, balbutcher lane, ballymun	dublin 11	dublin	192951	new dwelling house /apartment

Description

date_of_sale	address	postal_code	county	price	description
2021-02-12	123 walkinstown park, walkinstown, dublin 12	dublin 12	dublin	297000	second-hand
2021-01-04	12 oileain na cranoige.cranogue lsl, balbutcher lane, ballymun	dublin 11	dublin	192951	new

Description: example

- Check if a substring is present:
  - `second-hand`
  - `new`
- `in` operator

```
description_input = "new dwelling house / apartment"
if "new" in description_input:
    description_input = "new"
elif "second-hand" in description_input:
    description_input = "second-hand"
print("Description:", description_input)
```

```
Description: new
```

Let's practice!

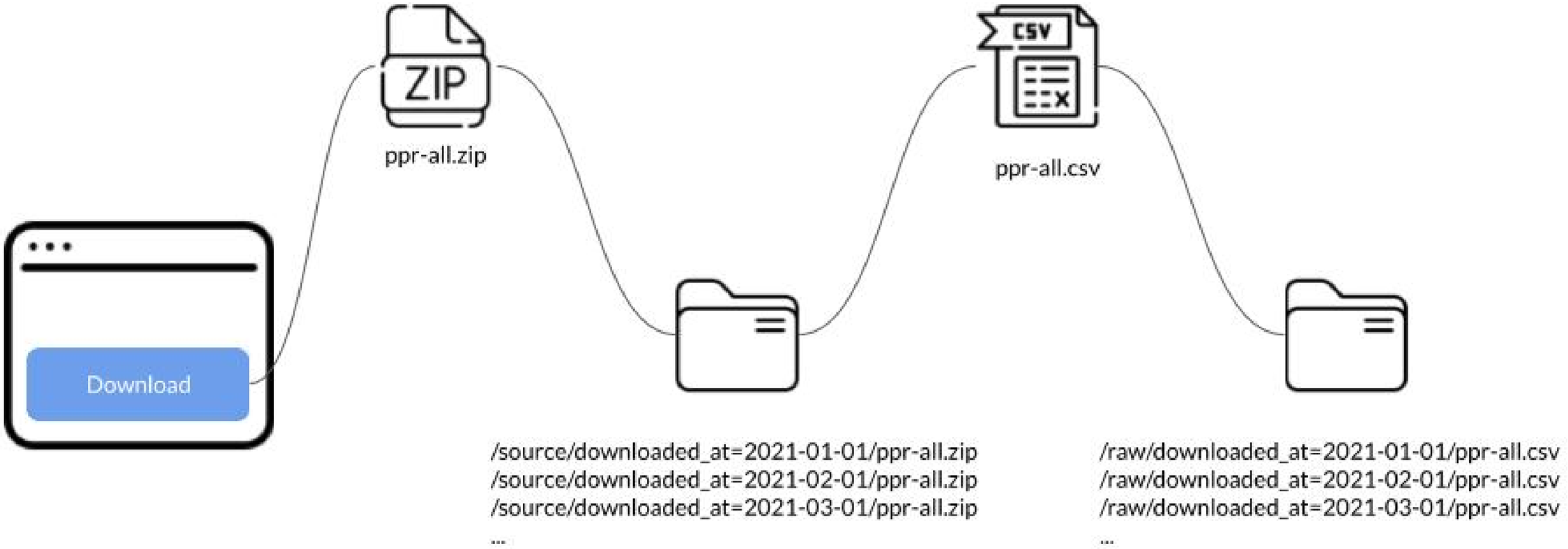
ETL IN PYTHON

Put transform operations together

ETL IN PYTHON

Stefano Francavilla
CEO - Geowox

Where we have left



E(Transform)L

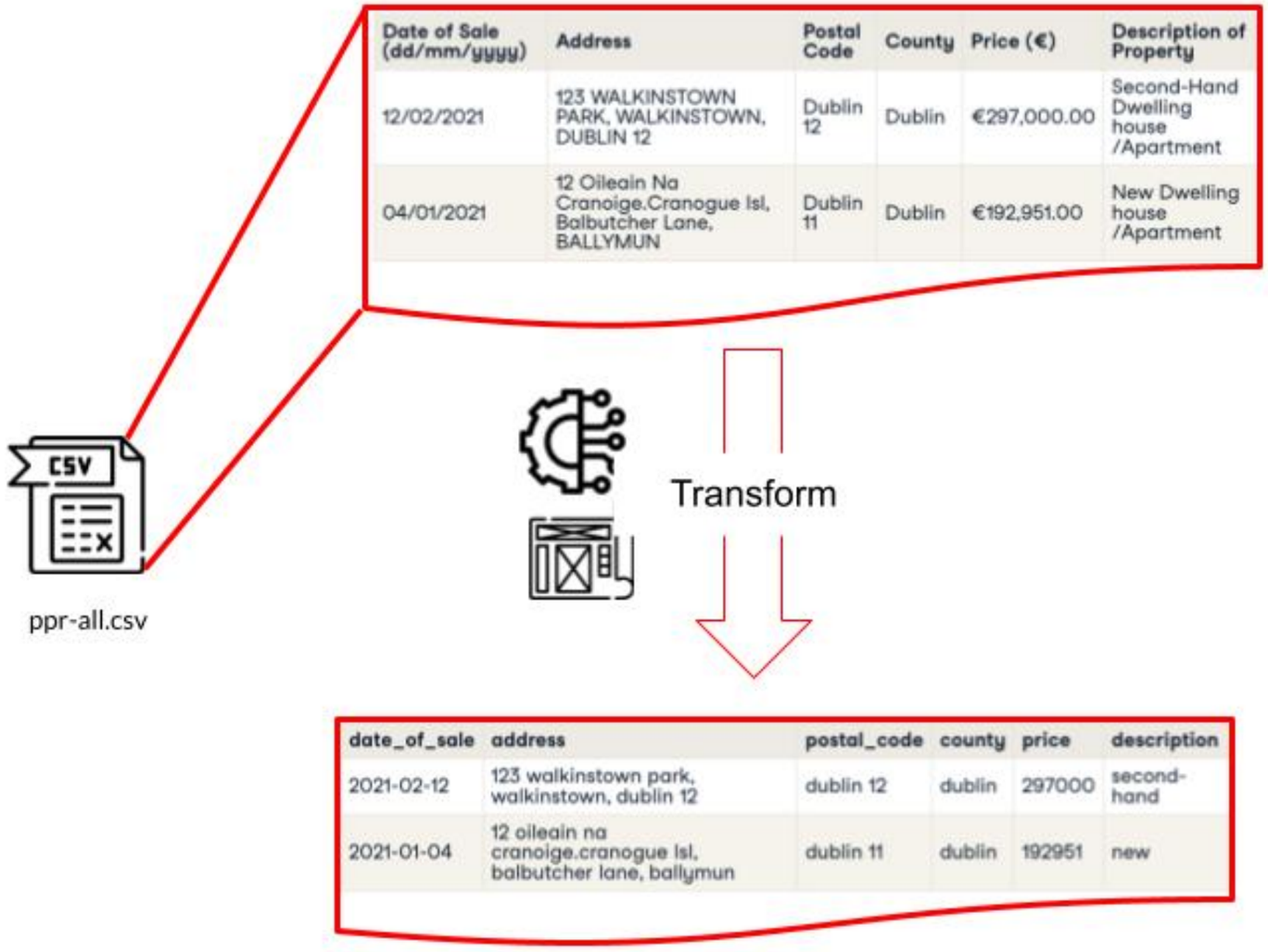
ppr-all.csv

E(Transform)L

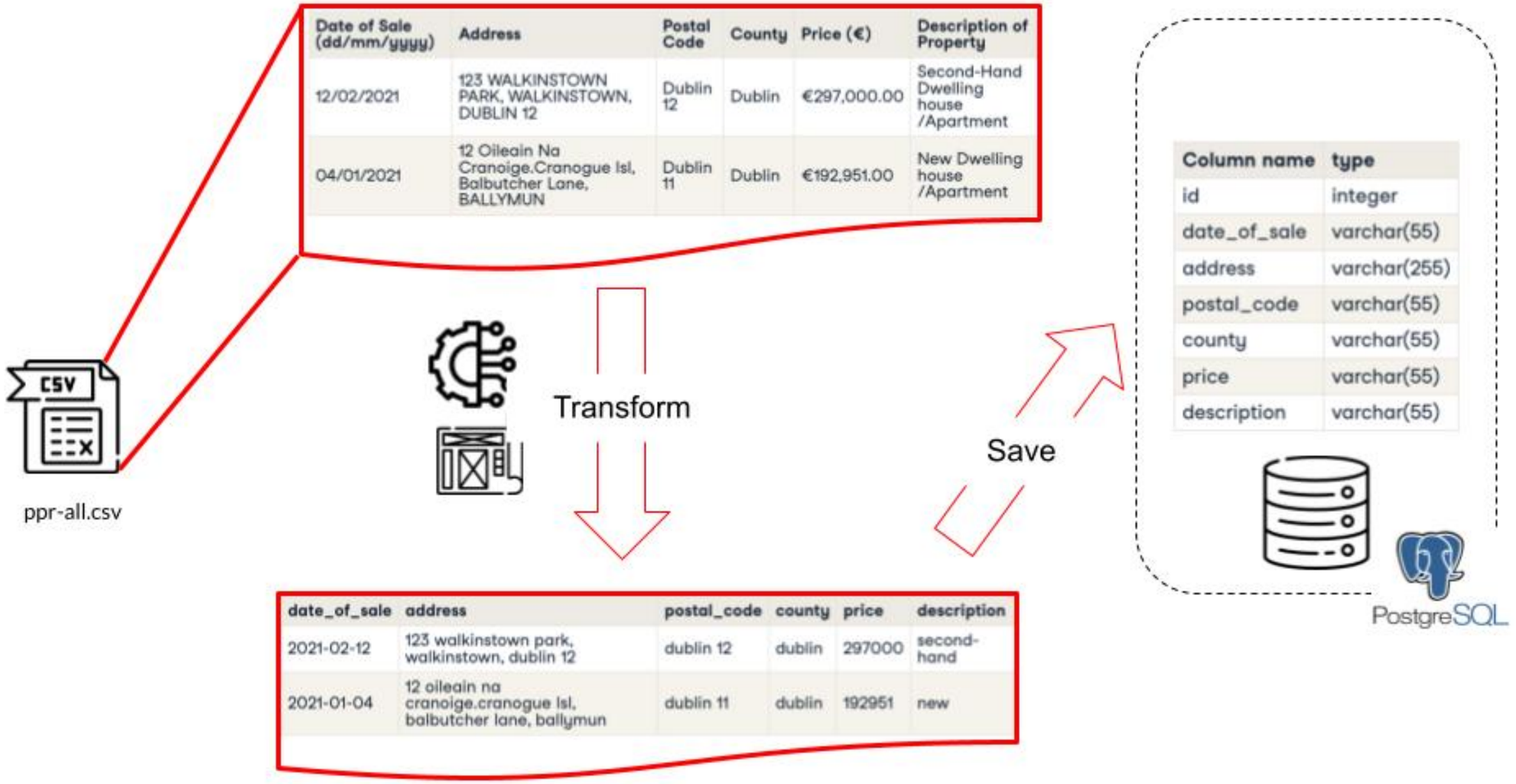
Date of Sale (dd/mm/yyyy)	Address	Postal Code	County	Price (€)	Description of Property
12/02/2021	123 WALKINSTOWN PARK, WALKINSTOWN, DUBLIN 12	Dublin 12	Dublin	€297,000.00	Second-Hand Dwelling house /Apartment
04/01/2021	12 Oileain Na Cranoige.Cranogue Isl, Balbutcher Lane, BALLYMUN	Dublin 11	Dublin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

E(Transform)L

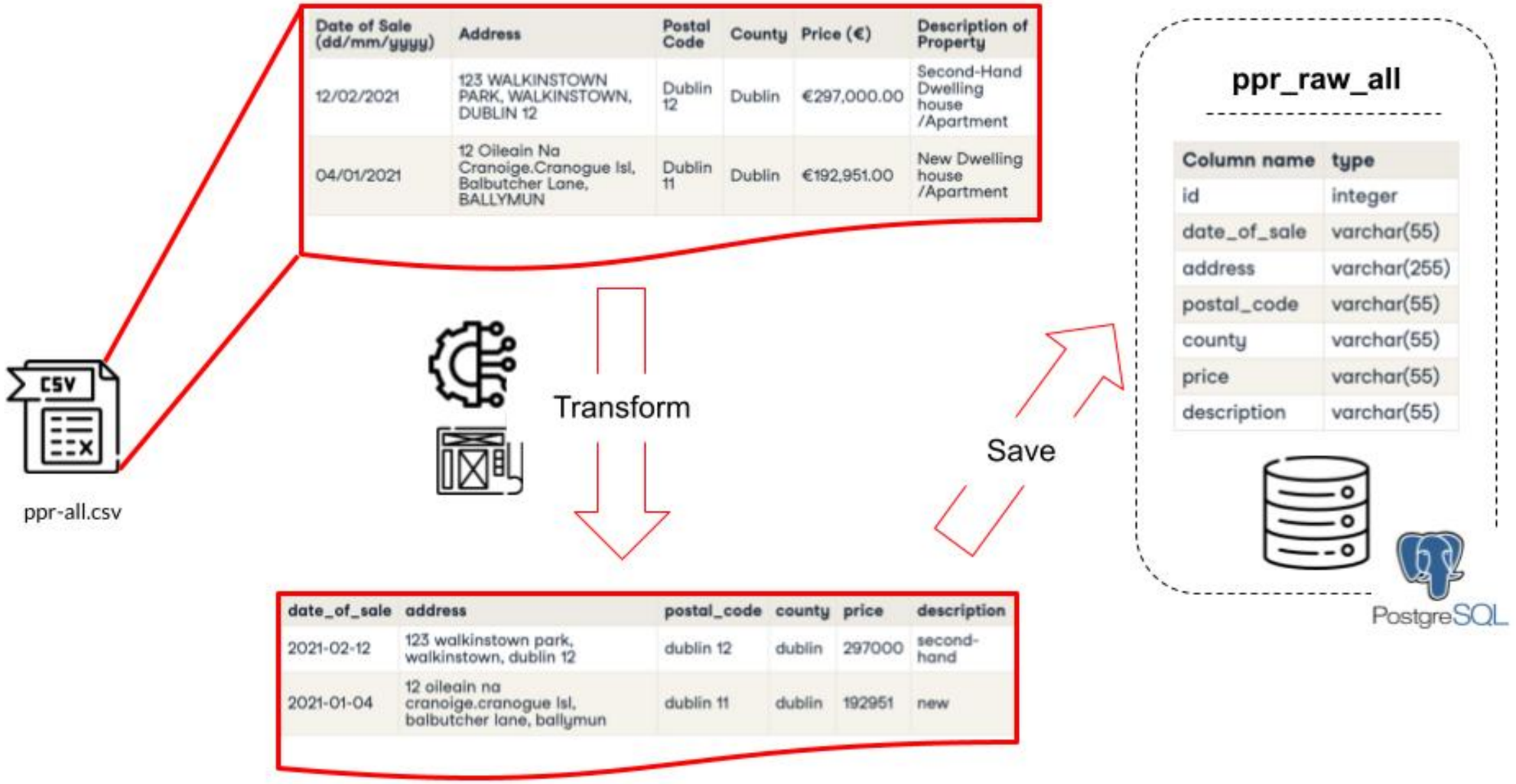
E(Transform)L



E(Transform)L



E(Transform)L



Common folder files

- Stored in the `script/common` folder
- `base.py` initializes engine and declarative base
- `tables.py` stores classes definition
- `create_tables.py` creates all tables defined in `tables.py`

```
- script
  - common
    - base.py
    - tables.py
    - create_tables.py
```

Base file

common/**base.py**

```
from sqlalchemy import create_engine
from sqlalchemy.orm.declarative import declarative_base

engine = create_engine(
    "postgresql+psycopg2://dcstudent:S3cretPassw0rd@localhost:5432/campdata-prod"
    )

Base = declarative_base()
```

Tables file

common/**tables.py**

```
from sqlalchemy import Column, Integer, String

from base import Base

class PprRawAll(Base):
    __tablename__ = "ppr_raw_all"

    id = Column(Integer, primary_key=True)
    # Rest of columns definition
```

Create tables

common/**create_tables.py**

- `Base.metadata` contains schema construct
- `Base.metadata.tables` is a list of tables

```
from base import Base
from tables import PprRawAll

for table in Base.metadata.tables:
    print(table)
```

```
ppr_raw_all
```

Create tables

common/**create_tables.py**

- `Base.metadata.create_all(engine)`

```
from base import Base, engine
from tables import PprRawAll

if __name__ == "__main__":
    Base.metadata.create_all(engine)
```

```
python common/create_tables.py
```

Bulk save objects

- `session.bulk_save_objects(list_of_objects)`
- `session.commit()`

Bulk save object: an example

```
session = Session(engine)
ppr_raw_objects = [PprRawAll(date_of_sale="2021-01-01",
                             address="7 bow street"),
                             postal_code="dublin 7",
                             county="dublin",
                             price="450000",
                             description="second=hand",
                   ),
                   PprRawAll(...)]

# Bulk save all new processed objects and commit
session.bulk_save_objects(ppr_raw_objects)
session.commit()
```

Let's practice!

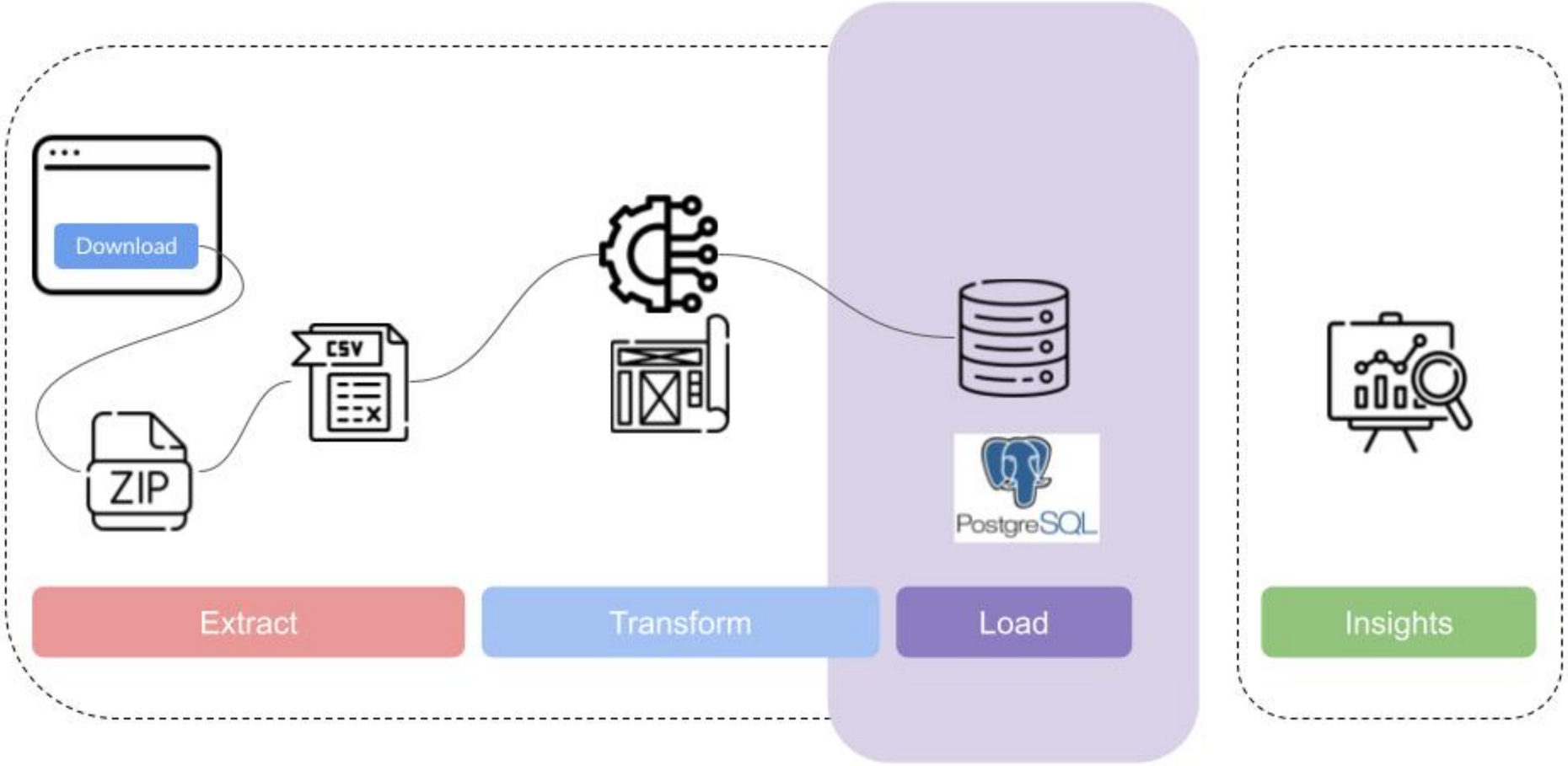
ETL IN PYTHON

Unique key definition and clean table

ETL IN PYTHON

Stefano Francavilla
CEO - Geowox

Where we are in the pipeline



What it looks like

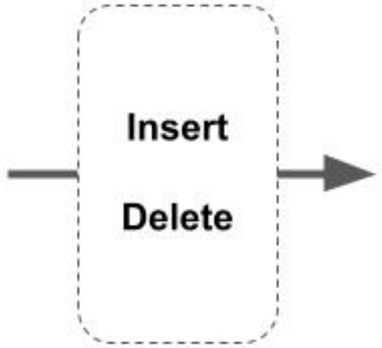
ppr_raw_all

Column name	type
id	Integer (Primary Key)
date_of_sale	String(55)
address	String(255)
postal_code	String(55)
county	String(55)
price	String(55)
description	String(255)

What it looks like

ppr_raw_all

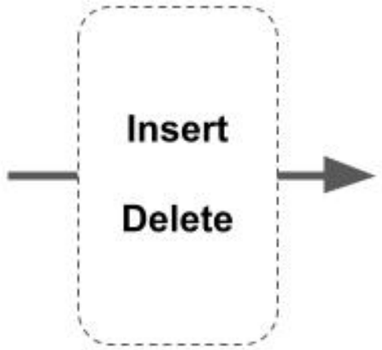
Column name	type
id	Integer (Primary Key)
date_of_sale	String(55)
address	String(255)
postal_code	String(55)
county	String(55)
price	String(55)
description	String(255)



What it looks like

ppr_raw_all

Column name	type
id	Integer (Primary Key)
date_of_sale	String(55)
address	String(255)
postal_code	String(55)
county	String(55)
price	String(55)
description	String(255)



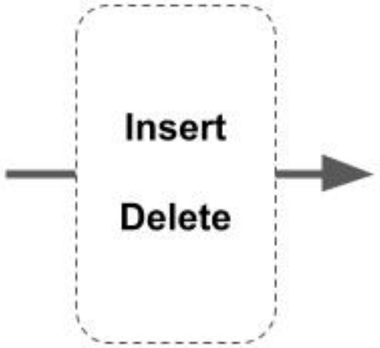
ppr_clean_all

Column name	type
id	Integer (Primary Key)
date_of_sale	Date
address	String(55)
postal_code	String(55)
county	String(55)
price	Integer
description	String(55)

What it looks like

ppr_raw_all

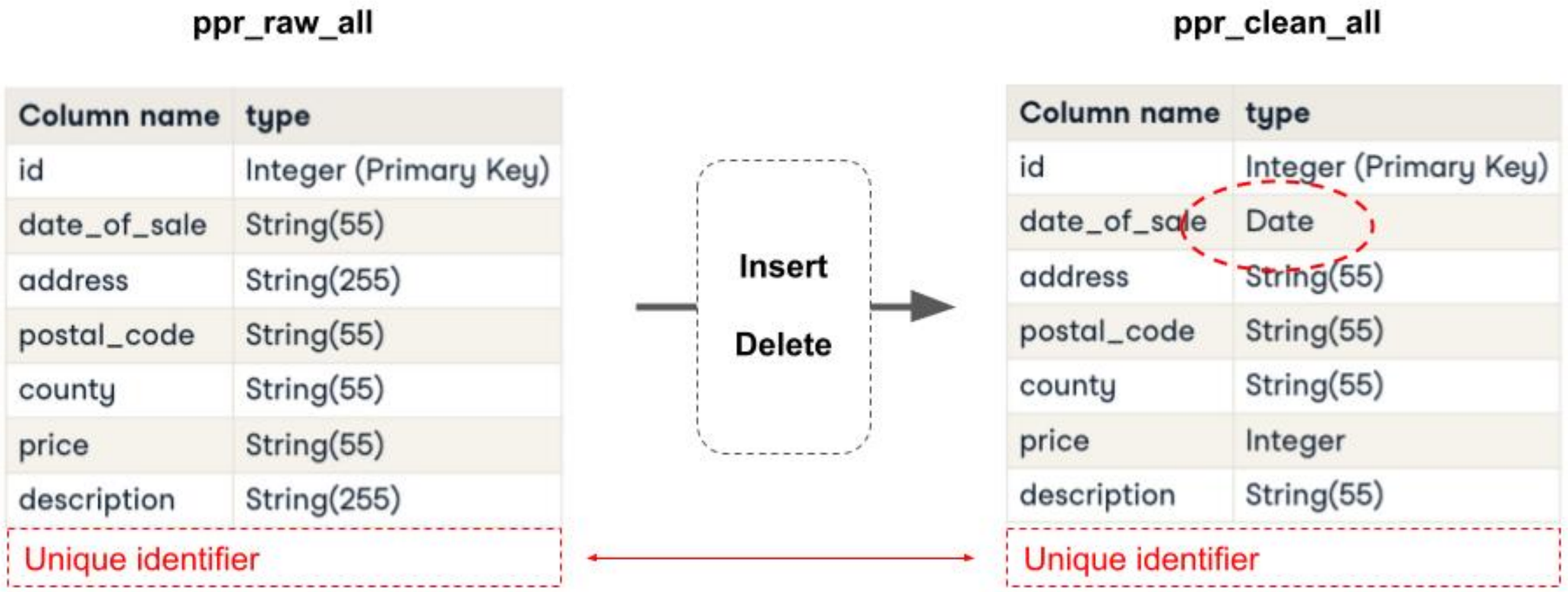
Column name	type
id	Integer (Primary Key)
date_of_sale	String(55)
address	String(255)
postal_code	String(55)
county	String(55)
price	String(55)
description	String(255)



ppr_clean_all

Column name	type
id	Integer (Primary Key)
date_of_sale	Date
address	String(55)
postal_code	String(55)
county	String(55)
price	Integer
description	String(55)

What it looks like



Date datatype

Table Name: `movies`

Column name	type
id	integer
title	varchar(55)
description	varchar(55)
release_date	date

```
from sqlalchemy import Column,
                       Integer,
                       String
                       Date
class Movies(Base):
    __tablename__ = "movies"
    id = Column(Integer)
    title = Column(String(55))
    description = Column(String(255))

    release_date = Column(Date)
```

Uniqueness

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

Uniqueness

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

Date of Sale (dd/mm/yyyy)

Uniqueness

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

Date of Sale (dd/mm/yyyy) + **Address**

Uniqueness

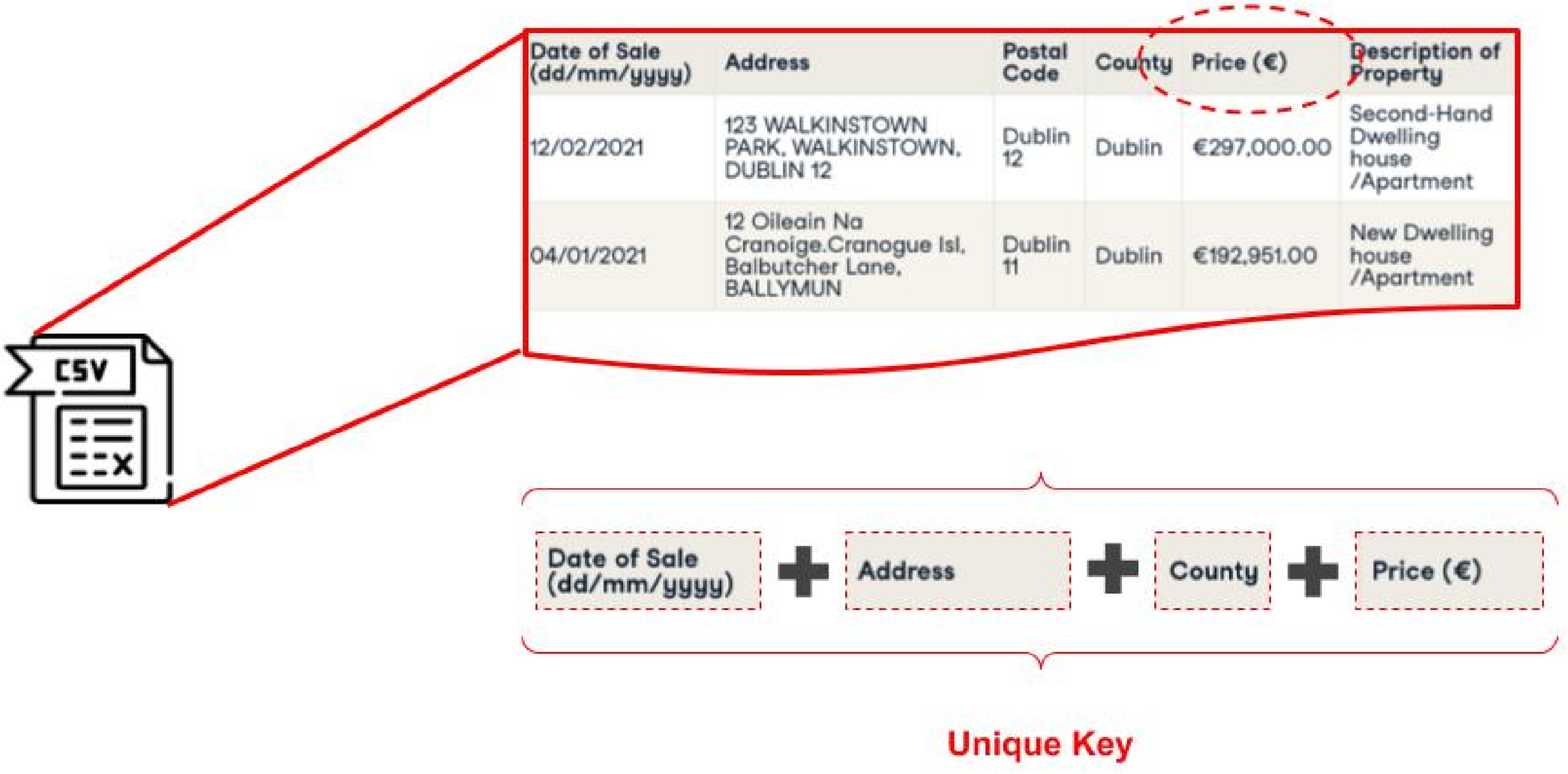

CSV
Date of Sale (dd/mm/yyyy)
Address
Postal Code
County
Price (€)
Description of Property
12/02/2021
123 WALKINSTOWN PARK, WALKINSTOWN, DUBLIN 12
Dublin 12
Dublin
€297,000.00
Second-Hand Dwelling house /Apartment
04/01/2021
12 Oileain Na Cranoige.Cranogue Isl, Balbutcher Lane, BALLYMUN
Dublin 11
Dublin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

Uniqueness

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

Date of Sale (dd/mm/yyyy) + Address + County + Price (€)

Uniqueness


CSV
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
Date of Sale (dd/mm/yyyy)
Address
County
Price (€)
Unique Key

Column property

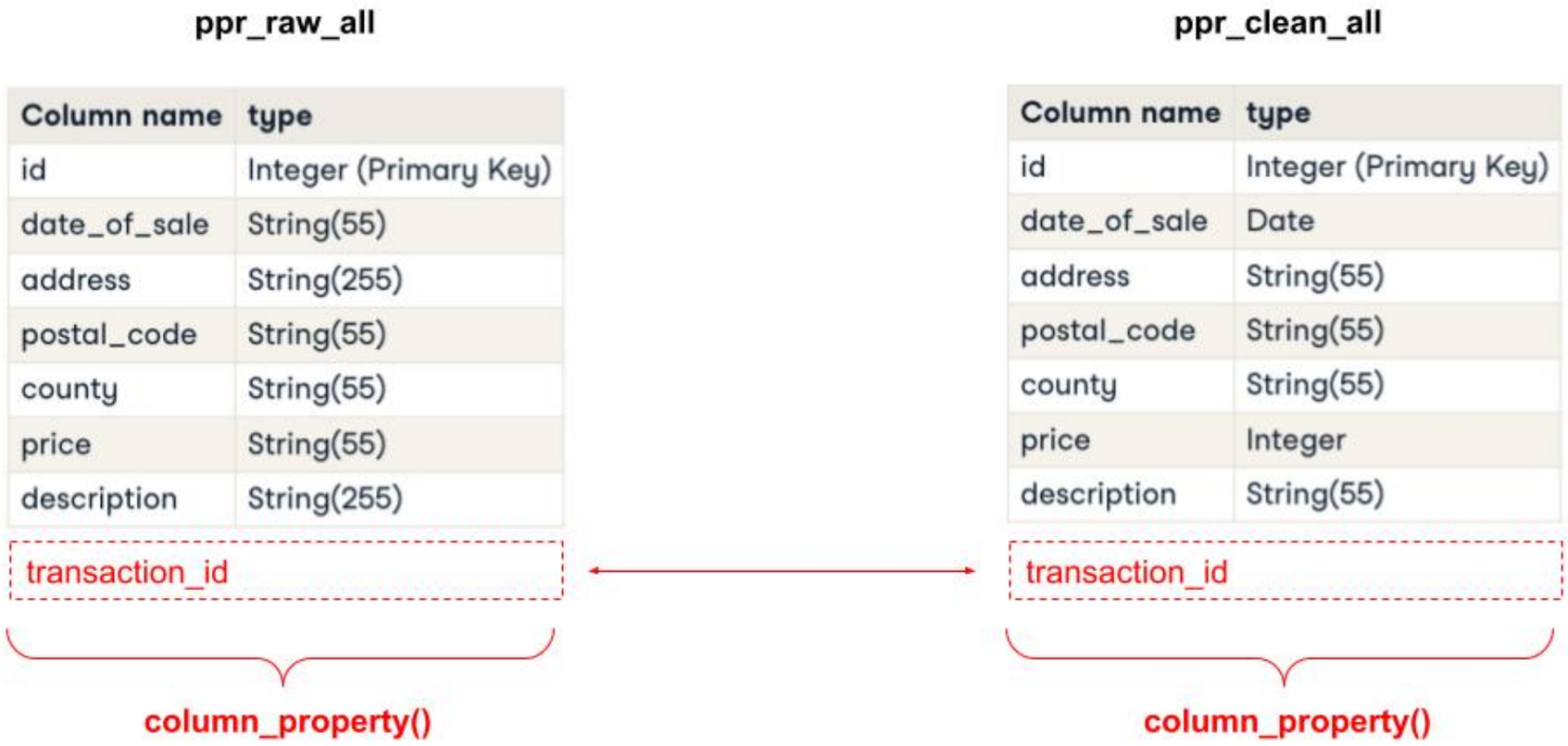
ppr_raw_all

Column name	type
id	Integer (Primary Key)
date_of_sale	String(55)
address	String(255)
postal_code	String(55)
county	String(55)
price	String(55)
description	String(255)

ppr_clean_all

Column name	type
id	Integer (Primary Key)
date_of_sale	Date
address	String(55)
postal_code	String(55)
county	String(55)
price	Integer
description	String(55)

Column property



Column property

- `from sqlalchemy.orm import column_property`
- Loaded at load time

Column property: example

```
from sqlalchemy.orm import column_property
class User(Base):
    __tablename__ = 'user'
    id = Column(Integer, primary_key=True)
    firstname = Column(String(50))
    lastname = Column(String(50))
    fullname = column_property(firstname + " " + lastname)

    user = User(firstname="John", lastname="Smith")
    print("User:", user.fullname)
```

```
User: John Smith
```

Let's practice!

ETL IN PYTHON

Insert and delete operations

ETL IN PYTHON

Stefano Francavilla
CEO - Geowox

Query API

- `SELECT * FROM movies`
- `session.query(Movies)`
- `session.query(Movies).all()`

Query API: an example

Table name: movies

id	title
1	The Big Short
2	The Social Network
3	The Avengers

```
from sqlalchemy import Column, Integer
from sqlalchemy.orm import
                        declarative_base

Base = declarative_base()

class Movies(Base):
    __tablename__ = "movies"
    id = Column(Integer,
                primary_key=True)
    title = Column(String(50))
```

Query API: an example

```
SELECT * FROM movies
```

```
session = Session(engine)
result = session.query(Movies).all()
for row in result:
    print("Title: ", row.title)
```

```
Title: The Big Short
Title: The Social Network
Title: The Avengers
```

Query API: an example

```
SELECT * FROM movies
WHERE id=1
```

```
session = Session(engine)

result = session.query(Movies)
                .filter(Movies.id == 1)

for row in result:
    print("Title: ",row.title)
```

```
Title: The Big Short
```

Delete

- `session.query().filter()`
- `session.query().filter().delete()`
- `session.query(Movies).filter(Movies.title == "The Big Short").delete()`
- `session.query(Movies).filter(Movies.title == "").delete()`

Insert

```
from sqlalchemy.dialects.postgresql import insert

values = [{"title": "Luca"}, {"title": "The Lord of the Rings"}]

insert(Movies).values(values)
```

Commit into table

```
stm = delete(Movies).filter(Movies.id == 1)


session.execute(stm)


session.commit()
```

Let's practice!

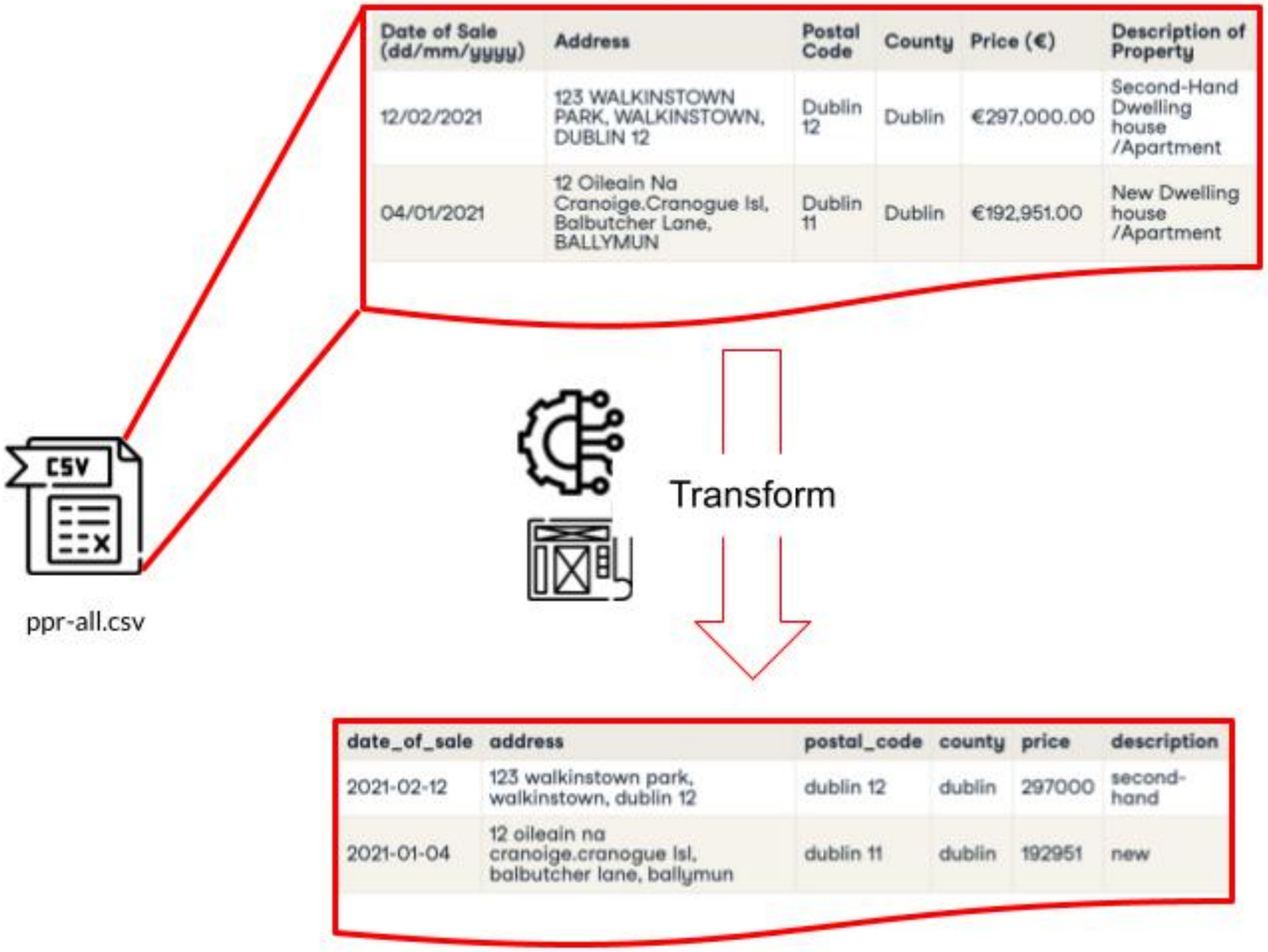
ETL IN PYTHON

Put load operations together

ETL IN PYTHON

Stefano Francavilla
CEO - Geowox

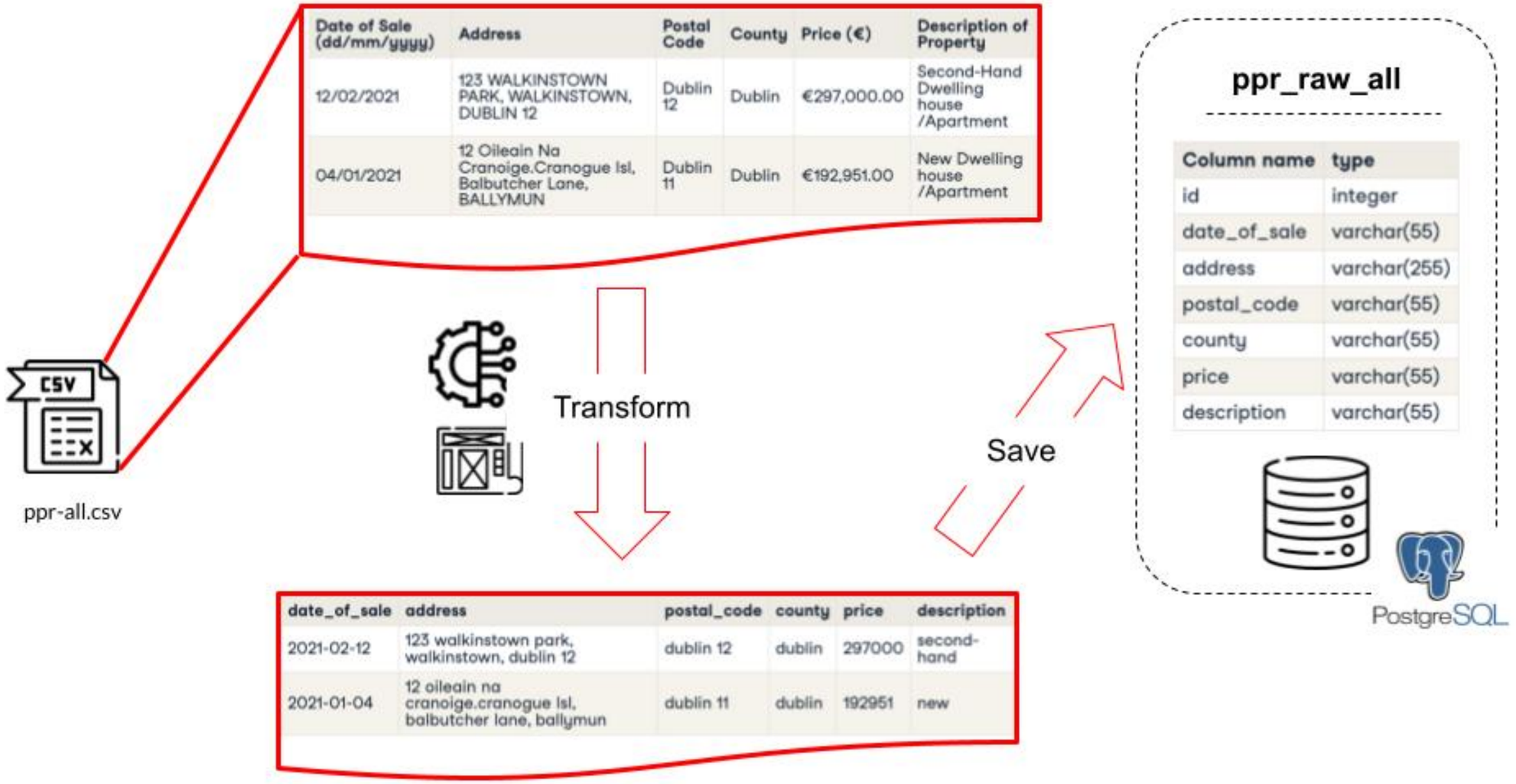
Where we have left



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

date_of_sale	address	postal_code	county	price	description
2021-02-12	123 walkinstown park, walkinstown, dublin 12	dublin 12	dublin	297000	second-hand
2021-01-04	12 oileain na cranoige.cranogue isl, balbutcher lane, ballymun	dublin 11	dublin	192951	new

Where we have left

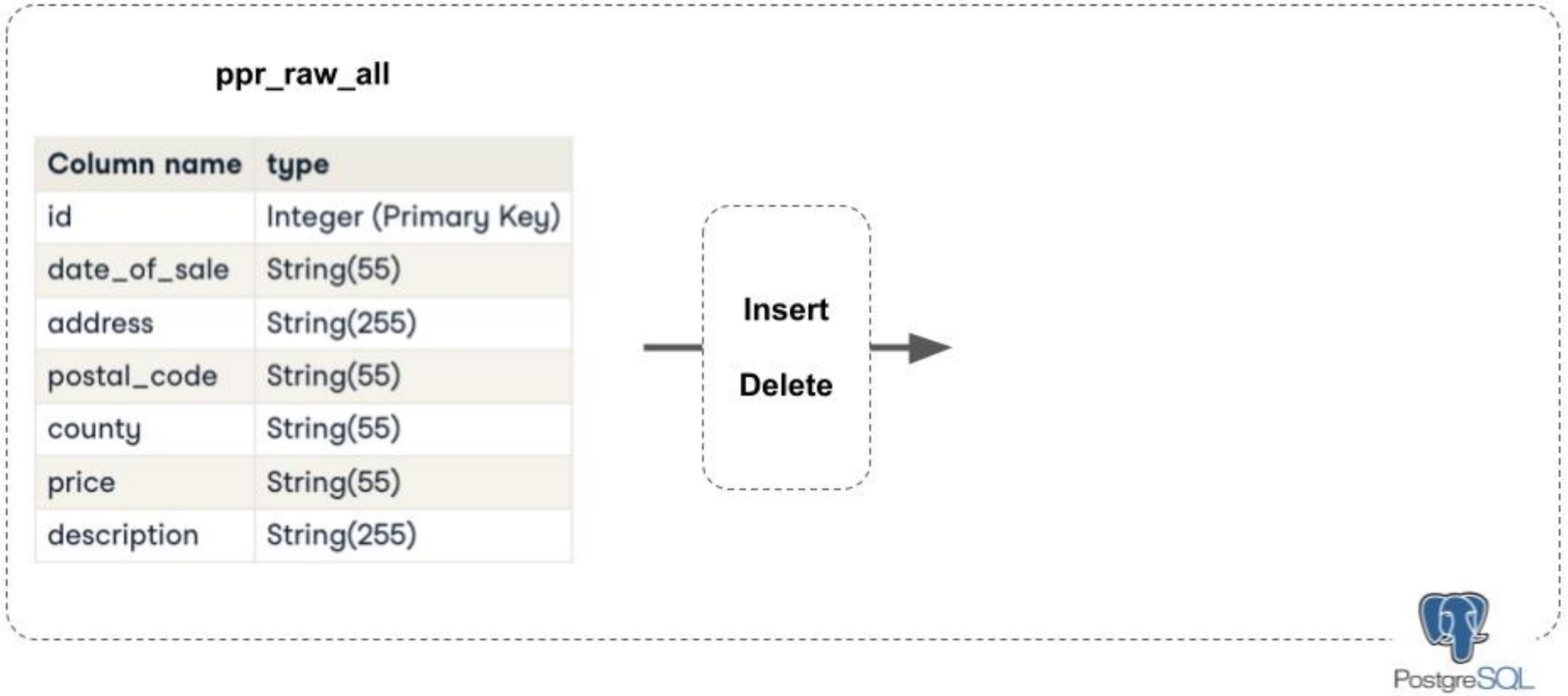


ETL(oad)

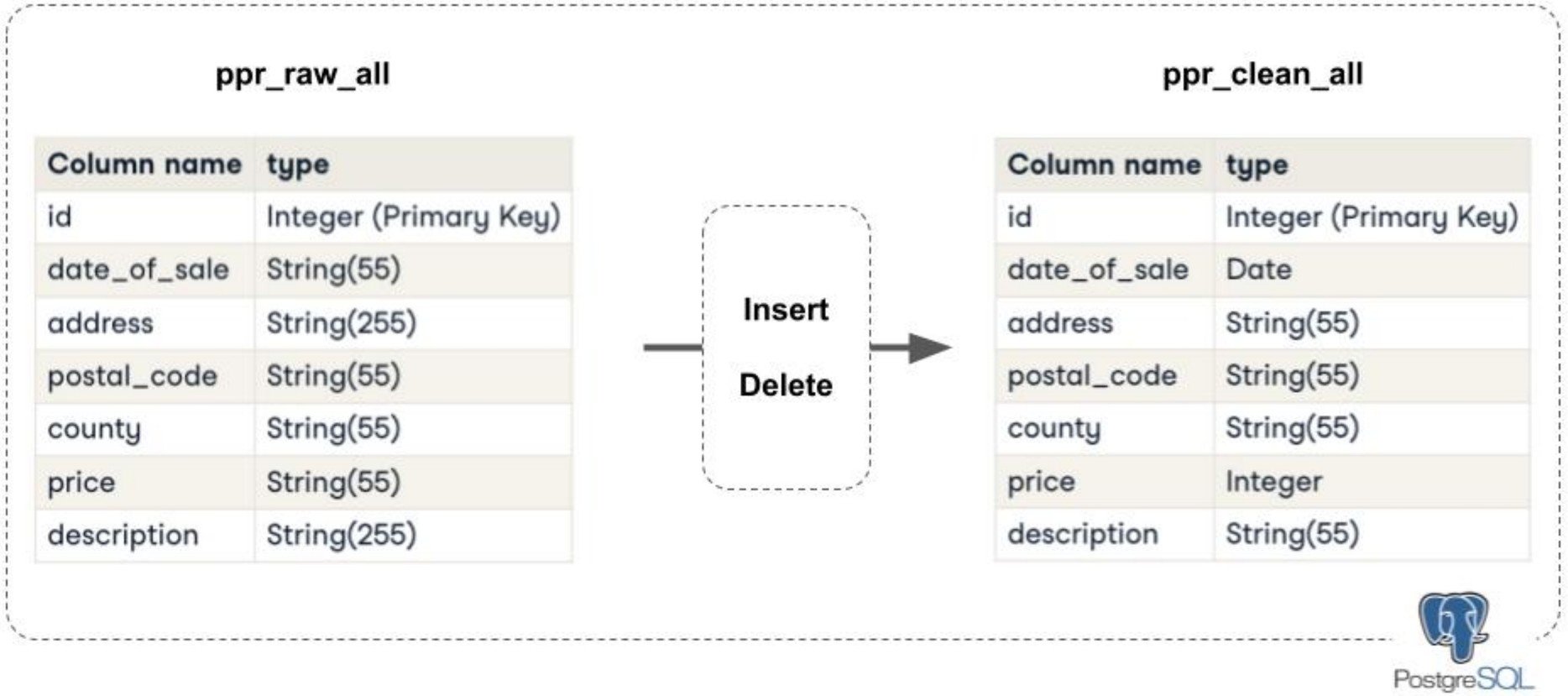
ppr_raw_all

Column name	type
id	Integer (Primary Key)
date_of_sale	String(55)
address	String(255)
postal_code	String(55)
county	String(55)
price	String(55)
description	String(255)

ETL(oad)


ppr_raw_all
Column name
type
id
Integer (Primary Key)
date_of_sale
String(55)
address
String(255)
postal_code
String(55)
county
String(55)
price
String(55)
description
String(255)
Insert
Delete
PostgreSQL

ETL(oad)


ppr_raw_all
Column name
type
id
Integer (Primary Key)
date_of_sale
String(55)
address
String(255)
postal_code
String(55)
county
String(55)
price
String(55)
description
String(255)
Insert
Delete
ppr_clean_all
Column name
type
id
Integer (Primary Key)
date_of_sale
Date
address
String(55)
postal_code
String(55)
county
String(55)
price
Integer
description
String(55)
PostgreSQL

Insert

ppr_raw_all

id	date_of_sale	address	...	transaction_id
1	2021-02-12	123 wanlkinstown park	...	2021-02-12_123 wanlkinstown park_...
2	2021-01-21	13 bow street	...	2021-01-21_13 bow street_...
3	2021-02-02	14 heytesbury street	...	2021-02-12_14 heytesbury street

ppr_clean_all

id	date_of_sale	address	...	transaction_id
125	2021-02-12	123 wanlkinstown park	...	2021-02-12_123 wanlkinstown park_...
280	2021-01-21	13 bow street	...	2021-01-21_13 bow street_...

Insert

ppr_raw_all

id	date_of_sale	address	...	transaction_id
1	2021-02-12	123 wanlkinstown park	...	2021-02-12_123 wanlkinstown park_...
2	2021-01-21	13 bow street	...	2021-01-21_13 bow street_...
3	2021-02-02	14 heytesbury street	...	2021-02-12_14 heytesbury street

ppr_clean_all

id	date_of_sale	address	...	transaction_id
125	2021-02-12	123 wanlkinstown park	...	2021-02-12_123 wanlkinstown park_...
280	2021-01-21	13 bow street	...	2021-01-21_13 bow street_...

Insert

ppr_raw_all

id	date_of_sale	address	...	transaction_id
1	2021-02-12	123 wanlkinstown park	...	2021-02-12_123 wanlkinstown park_...
2	2021-01-21	13 bow street	...	2021-01-21_13 bow street_...
3	2021-02-02	14 heytesbury street	...	2021-02-12_14 heytesbury street

Already in the clean table?

ppr_clean_all

transaction_id
2021-02-12_123 wanlkinstown park_...
2021-01-21_13 bow street_...

Insert

ppr_raw_all

id	date_of_sale	address	...	transaction_id
1	2021-02-12	123 wanlkinstown YES		~~2021-02-12_123 wanlkinstown park_...~~
2	2021-01-21	13 bow street	...	2021-01-21_13 bow street_...
3	2021-02-02	14 heytesbury street	...	2021-02-12_14 heytesbury street

Already in the clean table?

ppr_clean_all

transaction_id
2021-02-12_123 wanlkinstown park_...
2021-01-21_13 bow street_...

Insert

ppr_raw_all

id	date_of_sale	address	...	transaction_id
1	2021-02-12	123 wanlkinstown	YES	~~2021-02-12_123 wanlkinstown park_...~~
2	2021-01-21	13 bow street	YES	~~2021-01-21_13 bow street_...~~
3	2021-02-02	14 heytesbury street	...	2021-02-12_14 heytesbury street

Already in the clean table?

ppr_clean_all

transaction_id
2021-02-12_123 wanlkinstown park_...
2021-01-21_13 bow street_...

Insert

ppr_raw_all

id	date_of_sale	address	...	transaction_id
1	2021-02-12	123 wanlkinstown	YES	~~2021-02-12_123 wanlkinstown park_...~~
2	2021-01-21	13 bow street	YES	~~2021-01-21_13 bow street_...~~
3	2021-02-02	14 heytesbury stre	NO	2021-02-12_14 heytesbury street

Already in the clean table?

ppr_clean_all

transaction_id
2021-02-12_123 wanlkinstown park_...
2021-01-21_13 bow street_...

Insert


ppr_raw_all
id
date_of_sale
address
...
transaction_id
1
2021-02-12
123 wanlkinstown
YES
2021-02-12_123 wanlkinstown park_...
2
2021-01-21
13 bow street
YES
2021-01-21_13 bow street_...
3
2021-02-02
14 heytesbury stre
NO
2021-02-12_14 heytesbury street
Insert
ppr_clean_all
transaction_id
2021-02-12_123 wanlkinstown park_...
2021-01-21_13 bow street_...
2021-02-12_14 heytesbury street

Insert

- To select rows to insert:
  - In SQL: `NOT IN`
  - In Python: `~` and `.in_()`
- `insert().from_select([<columns>], <ids>)`

Insert: an example

SQL

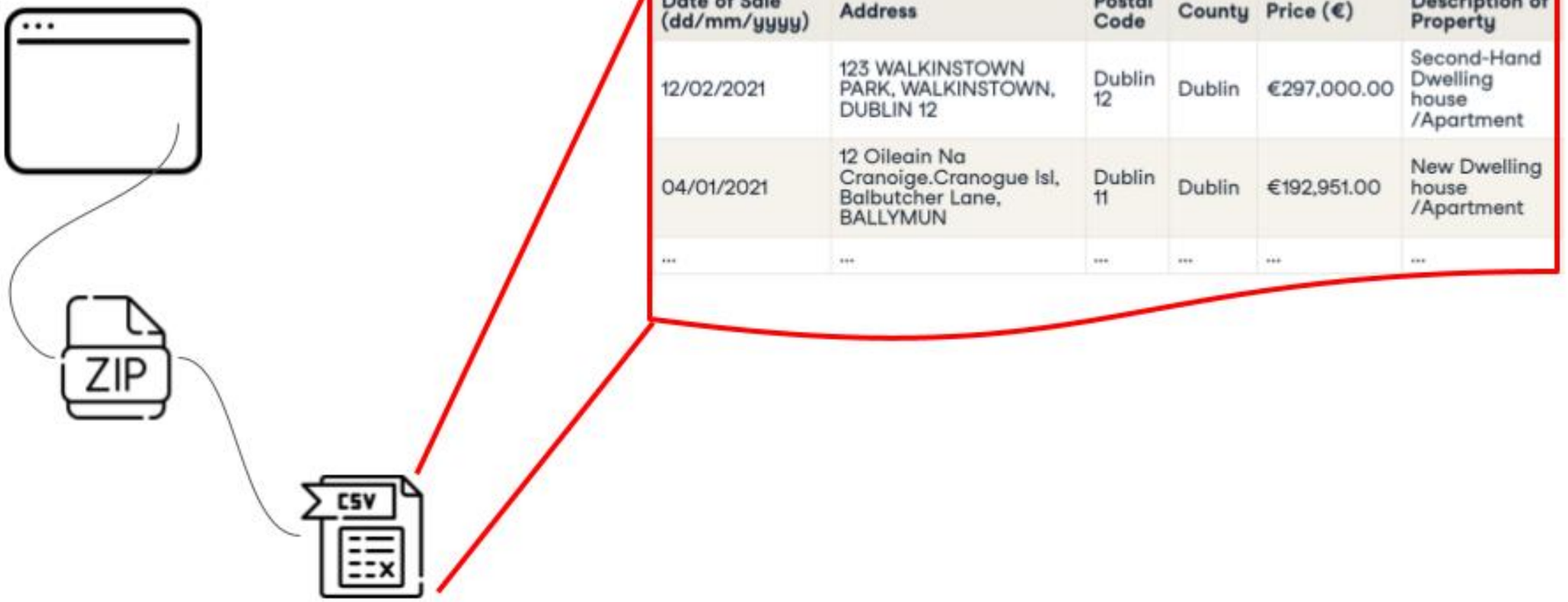
```
INSERT INTO ppr_clean_all
(SELECT date_of_sale, address, ...
FROM ppr_raw_all
WHERE transaction_id
NOT IN
    (SELECT transaction_id
     FROM ppr_clean_all)
)
```

Insert: an example

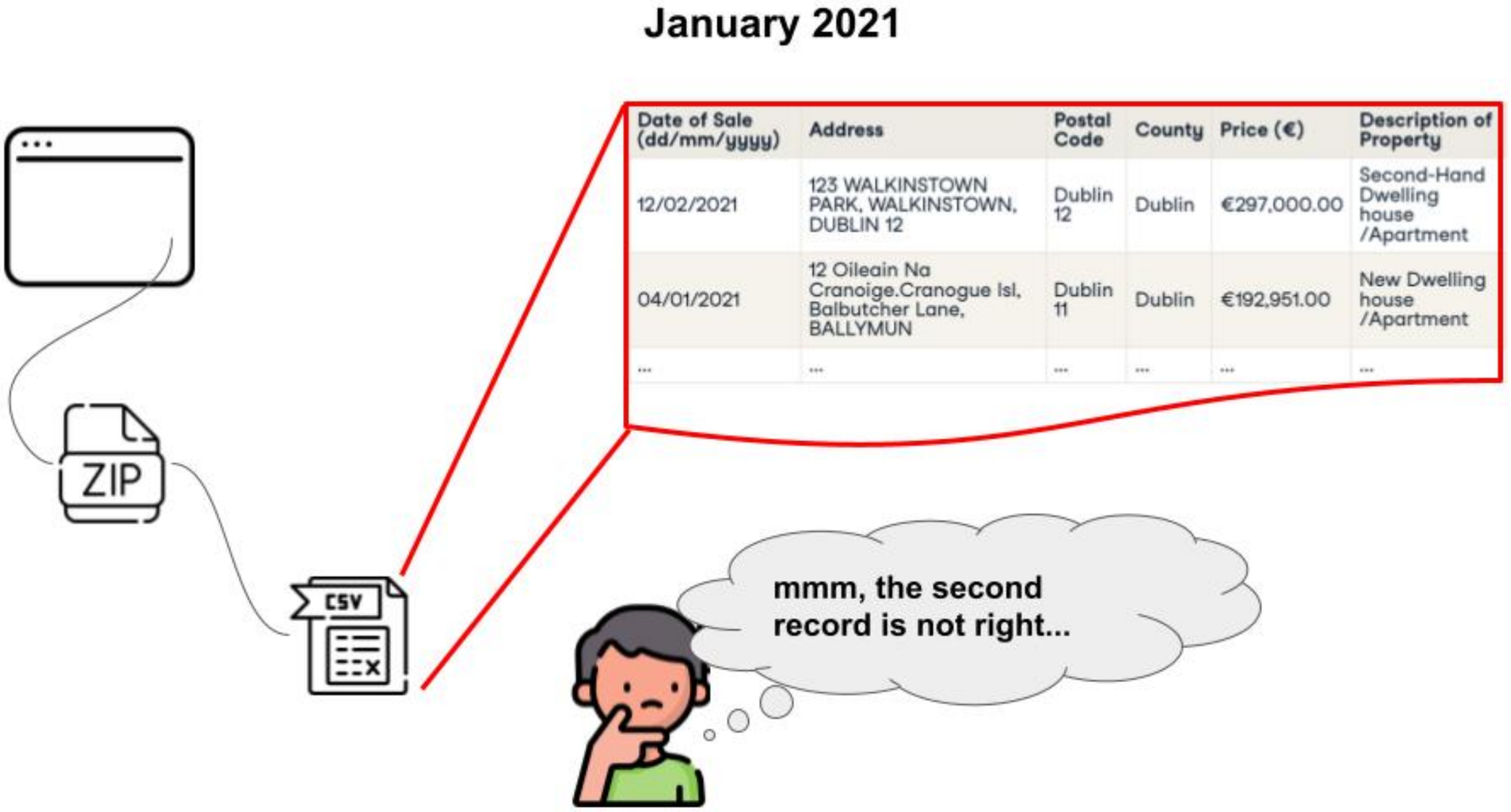
Python

```
clean_transaction_ids = session.query(PprCleanAll.date_of_sale,
                                      PprCleanAll.address,
                                      ...)
transactions_to_insert = session.query(PprRawAll)
                                .filter(
                                ~PprRawAll.transaction_id.in_(clean_transaction_ids)
                                )
stm = insert(PprCleanAll).from_select(['date_of_sale', 'address', ...],
                                      transactions_to_insert)
```

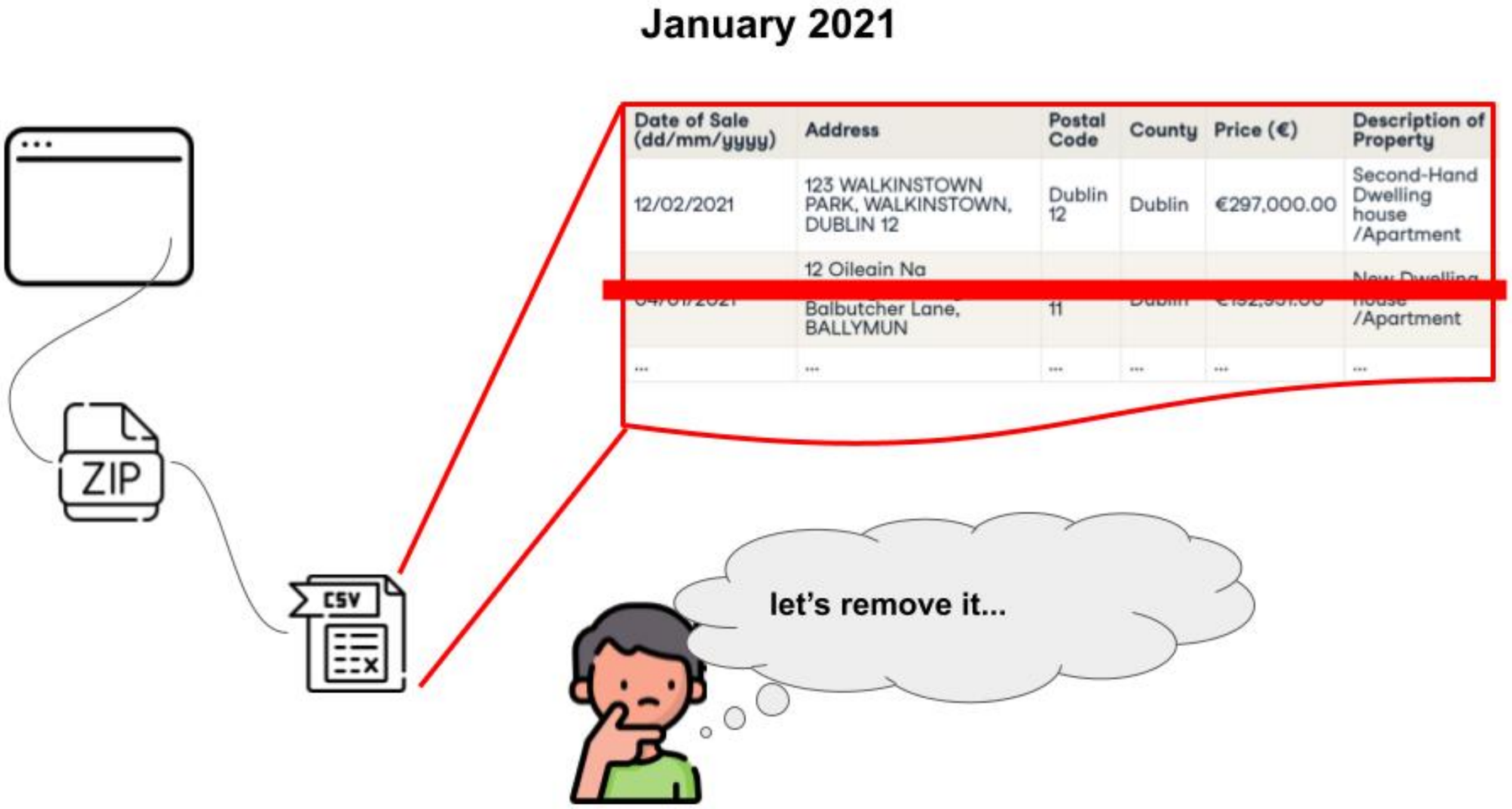
Delete


January 2021
ZIP
CSV
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
...
...
...
...
...
...

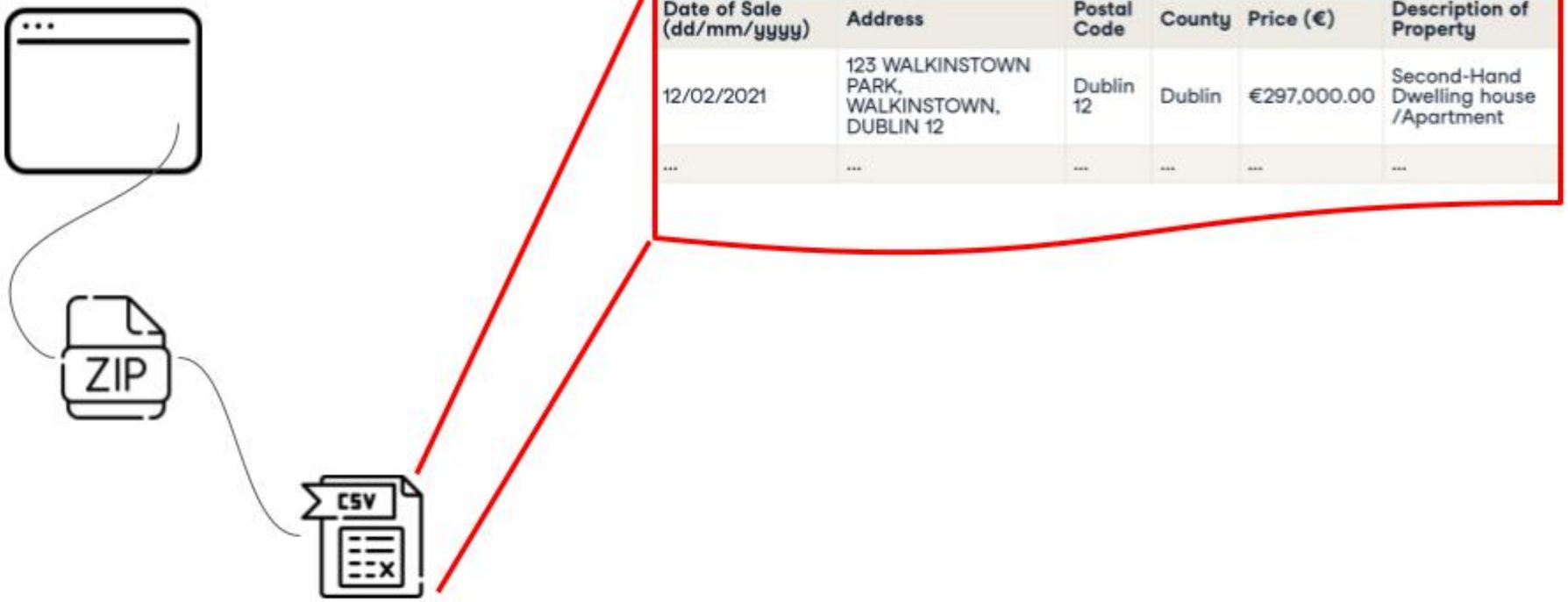
Delete


January 2021
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
...
ZIP
CSV
mmm, the second record is not right...

Delete


January 2021
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
12 Oileain Na
Balbutcher Lane, BALLYMUN
11
/Apartment
...
ZIP
CSV
let's remove it...

Delete


February 2021
ZIP
CSV
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
...
...
...
...
...
...

Delete

- `delete(PprCleanAll).filter()`
- SQL `NOT IN`

Delete: an example

SQL

```
DELETE FROM ppr_clean_all
WHERE transaction_id
NOT IN ("transaction_1", "transaction_2", "transaction_3")
```

Python

```
raw_transaction_ids = session.query(PprRawAll.transaction_id)
session.query(PprCleanAll)
       .filter(~PprCleanAll.transaction_id.in_(raw_transaction_ids))
       .delete()
```

Let's practice!

ETL IN PYTHON

Operators

ETL IN PYTHON

Stefano Francavilla
CEO - Geowox

Operators

- **In** / **not in** comparison
- **Basic** comparison
- **Identity** comparison
- **String** comparison
- **Conjunctions** and **negations** comparison

```
in_, ~in_
==, !=, >, >=, <, <=
is_, is_not
like, notlike
and_, or_
```

and_() operator

- **Conjunction** of expressions in `WHERE` clause
- `from sqlalchemy import and_`
- Use:
  - `and_(expression1, expression2, ..., expressionN)`
  - `(expression1 & expression2 & ... & expressionN)`

and_() operator: an example

SQL

```
SELECT * FROM ppr_clean_all
WHERE date_of_sale >= '2021-01-01' AND date_of_sale <= '2021-01-10'
```

Python

```
from sqlalchemy import and_

result = session.query(PprCleanAll)
                .filter(and_(PprCleanAll.date_of_sale >= '2021-01-01',
                             PprCleanAll.date_of_sale <= '2021-01-10'))
                .all()
```

or_() operator

- **Disjunction** of expressions in the `WHERE` clause
- `from sqlalchemy import or_`
- Use:
  - `or_(expression1, expression2, ..., expressionN)`
  - `(expression1 | expression2 ... | expressionN)`

or_() operator: an example

SQL

```
SELECT * FROM ppr_clean_all
WHERE price <= 50000 OR price >= 5000000
```

Python

```
from sqlalchemy import or_

result = session.query(PprCleanAll)
    .filter(or_(PprCleanAll.price <= 50000,
                PprCleanAll.price >= 5000000))
    .all()
```

Let's practice!

ETL IN PYTHON

Sqlalchemy func

ETL IN PYTHON

Stefano Francavilla
CEO - Geowox

SQL aggregate functions

- Aggregate functions perform calculation on rows
- `COUNT()` , number of rows
- `SUM()` , sum of all or distinct values
- `MAX()` , maximum value
- `MIN()` , minimum value
- `AVG()` , average value

The func attribute

- `from sqlalchemy import func`
- Generates SQL functions (like `COUNT(*)`)
  - `COUNT` --> `func.count()`
  - `SUM` --> `func.sum()`
  - `MAX` --> `func.max()`
  - `MIN` --> `func.min()`
  - `AVG` --> `func.avg()`

The func attribute: an example

Table name: **Products**

id	category	name	price
1	books	Sapiens	12
2	electronics	iPhone 12	900
3	books	Measure What Matters	10
4	books	Greenlights	14
5	electronics	Macbook Pro 13	1500

```
class Products(Base):
    __tablename__ = "products"

    id = Column(Integer,
                primary_key=True)
    category = Column(String(55))
    name = Column(String(55))
    price = Column(Integer)
```

The func attribute: COUNT(*)

SQL

```sql
SELECT COUNT(*) FROM products
```

Python

```python
result = session.query(func.count(Products.id))
                .all()

print("Result:", result)
```

- `COUNT(*)`
  - **counts** null values
- `COUNT([Column name])`
  - **doesn't count** null values

```
Result: [(5,)]
```

- `id` primary key and not null
- `.all()` retrieves the query result

The func attribute: SUM()

SQL

```sql
SELECT SUM(price)
FROM products
```

Python

```python
result = session.query(func.sum(Products.price)).all()

print("Result:", result)
```

- `func.sum([Column name])`

```
Result: [(2436,)]
```

The func attribute: MAX() and MIN()

SQL

```sql
SELECT MAX(price), MIN(price)
FROM products
```

Python

```python
result = session.query(func.max(Products.price),
                       func.min(Products.price)).all()

print("Result:", result)
```

- `func.max([Column name])`
- `func.min([Column name])`
- `session.query([Arg1], [Arg2])`

```
Result: [(1500, 10)]
```

The func attribute: AVG()

SQL

```
SELECT AVG(price) FROM products
```

Python

```
result = session.query(func.avg(Products.price)).all

print("Result:", result)
print("Result (int):", int(result[0][0]))
```

- `func.avg([Column name])`

```
Result: [(Decimal('487.2'),)]

Result (int): 487
```

Group by

- `SELECT column FROM table GROUP BY <column>`
- Divides results into groups
- Then applies the aggregate function
- `session.query().group_by([Column name]).all()`

Group by: an example

id	category	name	price
1	books	Sapiens	12
2	electronics	iPhone 12	900
3	books	Measure What Matters	10
4	books	Greenlights	14
5	electronics	Macbook Pro 13	1500

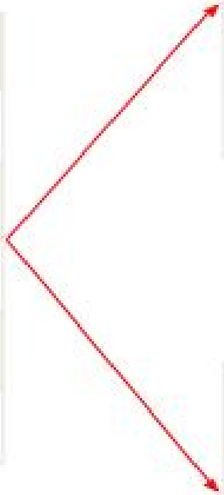
```
SELECT category, SUM(price)
FROM products
GROUP BY category
```

Group by: an example

id	category	name	price
1	books	Sapiens	12
2	electronics	iPhone 12	900
3	books	Measure What Matters	10
4	books	Greenlights	14
5	electronics	Macbook Pro 13	1500

Group by: an example

id	category	name	price
1	books	Sapiens	12
2	electronics	iPhone 12	900
3	books	Measure What Matters	10
4	books	Greenlights	14
5	electronics	Macbook Pro 13	1500

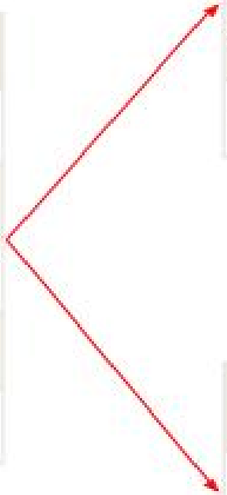


id	category	name	price
1	books	Sapiens	12
3	books	Measure What Matters	10
4	books	Greenlights	14

id	category	name	price
2	electronics	iPhone 12	900
5	electronics	Macbook Pro 13	1500

Group by: an example

id	category	name	price
1	books	Sapiens	12
2	electronics	iPhone 12	900
3	books	Measure What Matters	10
4	books	Greenlights	14
5	electronics	Macbook Pro 13	1500

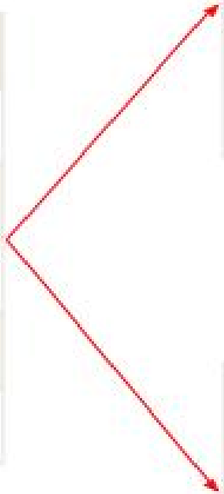


id	category	name	price
1	books	Sapiens	12
3	books	Measure What Matters	10
4	books	Greenlights	14

id	category	name	price
2	electronics	iPhone 12	900
5	electronics	Macbook Pro 13	1500

Group by: an example

id	category	name	price
1	books	Sapiens	12
2	electronics	iPhone 12	900
3	books	Measure What Matters	10
4	books	Greenlights	14
5	electronics	Macbook Pro 13	1500



id	category	name	price
1	books	Sapiens	12
3	books	Measure What Matters	10
4	books	Greenlights	14

id	category	name	price
2	electronics	iPhone 12	900
5	electronics	Macbook Pro 13	1500

Group by: an example

id	category	name	price
1	books	Sapiens	12
2	electronics	iPhone 12	900
3	books	Measure What Matters	10
4	books	Greenlights	14
5	electronics	Macbook Pro 13	1500

category	price
books	36
electronics	2400

Group by: an example

SQL

```sql
SELECT category, SUM(price)
FROM products
GROUP BY category
```

Python

```python
result = session.query(Products.category,
                       func.sum(Products.price))
                .group_by(Products.category).all()
print("Result:", result)
```

category character varying (55)	sum bigint
books	36
electronics	2400

```
Result: [('books', 36), ('electronics', 2400)]
```

Let's practice!

ETL IN PYTHON

Create the insights

ETL IN PYTHON

Stefano Francavilla
CEO - Geowox

Where we left

ppr_clean_all

Columns
id
date_of_sale
address
postal_code
county
price
description

- Derive insights for shareholders
- Work with `date_of_sale`, `county` and `price`
  - `county` is a string
  - `price` is numeric
  - `date_of_sale` is a date

What insights do we need?

- **Total** number sales
- **Sum** of all sales in euro
- **Highest** sold price
- **Lowest** sold price
- **Average** sold price
- Specific **date** range (e.g. first quarter, from 2021-01-01 to 2021-03-30)
- For each **county** (e.g. Dublin, Galway, Cork, etc.)

What insights do we need?

SQL

```
SELECT county,
       COUNT(*),
       SUM(*),
       MAX(price),
       MIN(price),
       AVG(price)
FROM ppr_clean_all
WHERE data_of_sale >= "2021-01-01" AND data_of_sale <= "2021-03-30"
GROUP BY county
```

Views

- Views are **not physically materialized**
- Create view with **pure SQL**
- Defined with `CREATE OR REPLACE VIEW <table_name> AS` query

```
CREATE OR REPLACE VIEW insights AS
SELECT * FROM ppr_clean_all
WHERE county='dublin'
```

Raw SQL on SQLAlchemy

- `session.execute("CREATE OR REPLACE VIEW AS ...")`
- `session.commit()`

Let's practice!

ETL IN PYTHON

Working with Excel files

ETL IN PYTHON

Stefano Francavilla
CEO - Geowox

xlsxwriter library

- `import xlsxwriter`
- `.xlsx` file format
- Can write:
  - text
  - numbers
  - formulas
  - images
  - charts
  - others

Workbook

```
import xlsxwriter
workbook = xlsxwriter.Workbook("Insights.xlsx")
# Do things with the workbook
workbook.close()
```

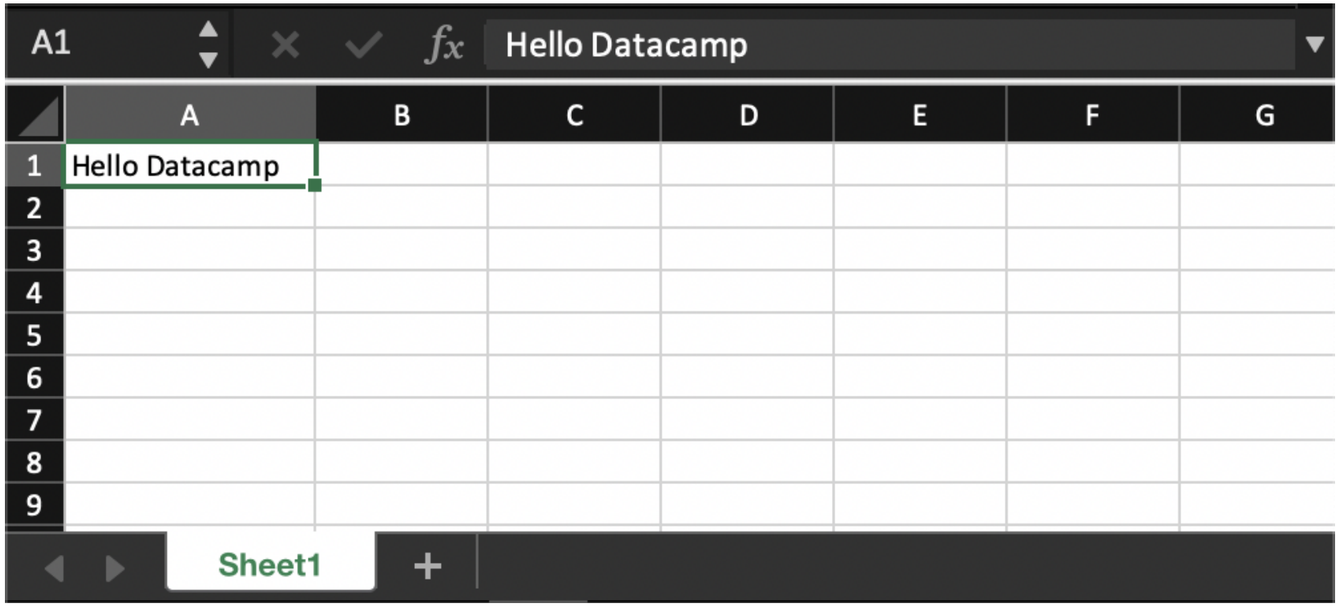
- `xlsxwriter.Workbook()` creates `.xlsx` files
- `workbook.close()` closes the file

Worksheet

- Manages Excel file sheets:
  - formulas
  - data
  - graph
- `worksheet = workbook.add_worksheet()`
- Default names: `Sheet1` , `Sheet2` , `...` , `SheetN`
- `worksheet = workbook.add_worksheet("Results")`
- `worksheet.write(row, column, data)`
  - A1 is `(0, 0)`

Worksheet: an example

```
import xlsxwriter
workbook = xlsxwriter.Workbook("Greetings.xlsx")
worksheet = workbook.add_worksheet()
worksheet.write(0, 0, "Hello Datacamp")
workbook.close()
```



Worksheet: add_table()

- `add_table()` adds table into the Excel file
- `add_table(<range (e.g. "B3:C5")>, {options})`
- `options`:
  - `data` specifies data to insert in cells
  - `columns` sets specific properties
- `"columns": [{"header": "Header 1"}, ..., {"header": "Header N"}]` overrides default column names (e.g. `Column1`, `...`, `ColumnN`)

```
{"columns": [
                {"header": "id"},
                {"header": "name"}
            ]
}
```

add_table(): an example

Books.xlsx

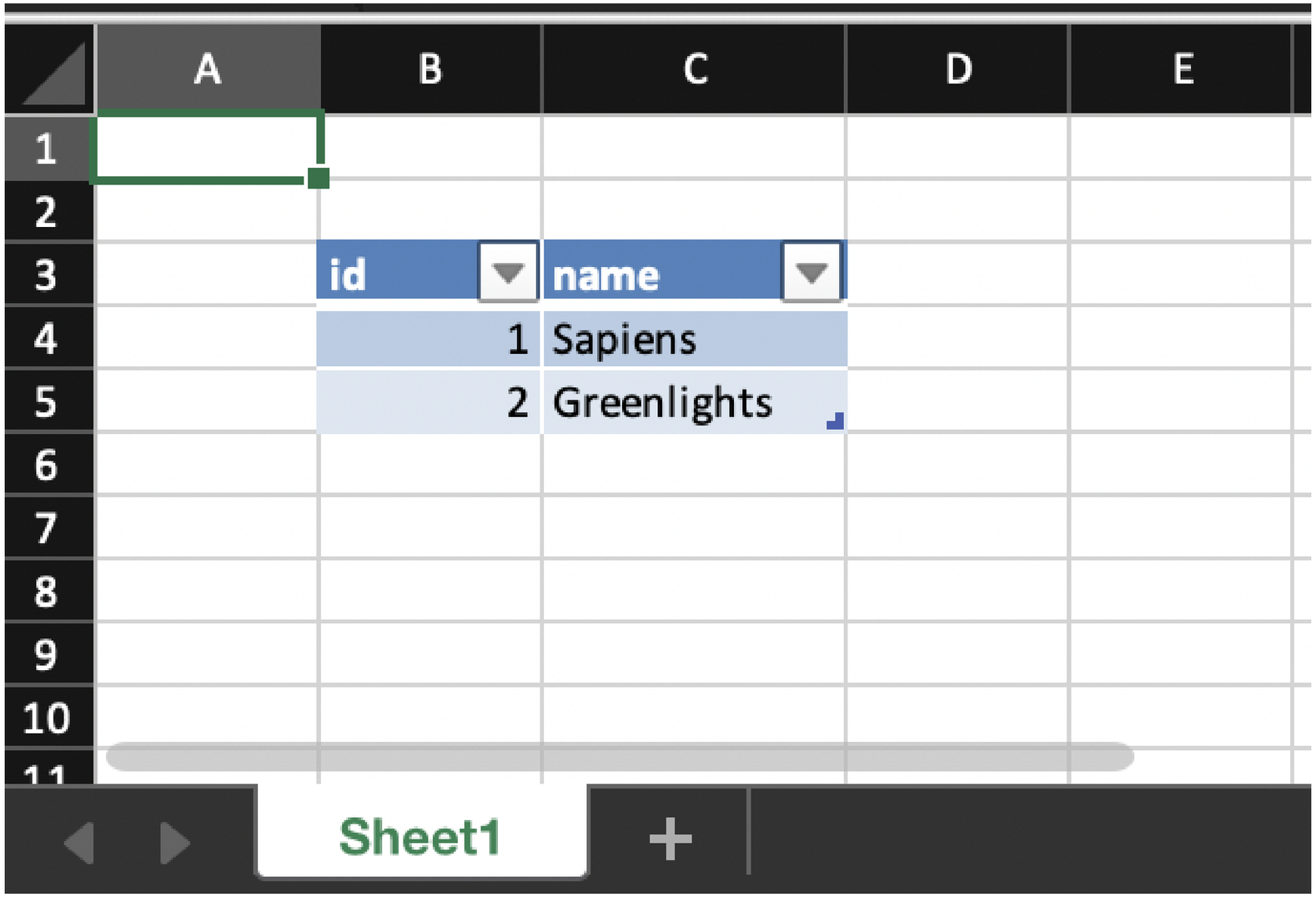
id	name
1	Sapiens
2	Greenlights

add_table(): an example

```
import xlsxwriter


workbook = xlsxwriter.Workbook("Books.xlsx")
worksheet = workbook.add_worksheet()
data = [[1, "Sapiens"],
        [2, "Greenlights"]]
worksheet.add_table(
    "B3:E6", {"data": data,
              "columns": [
                 {"header": "id"},
                 {"header": "name"}
              ]})
workbook.close()
```

add_table(): an example



Let's practice!

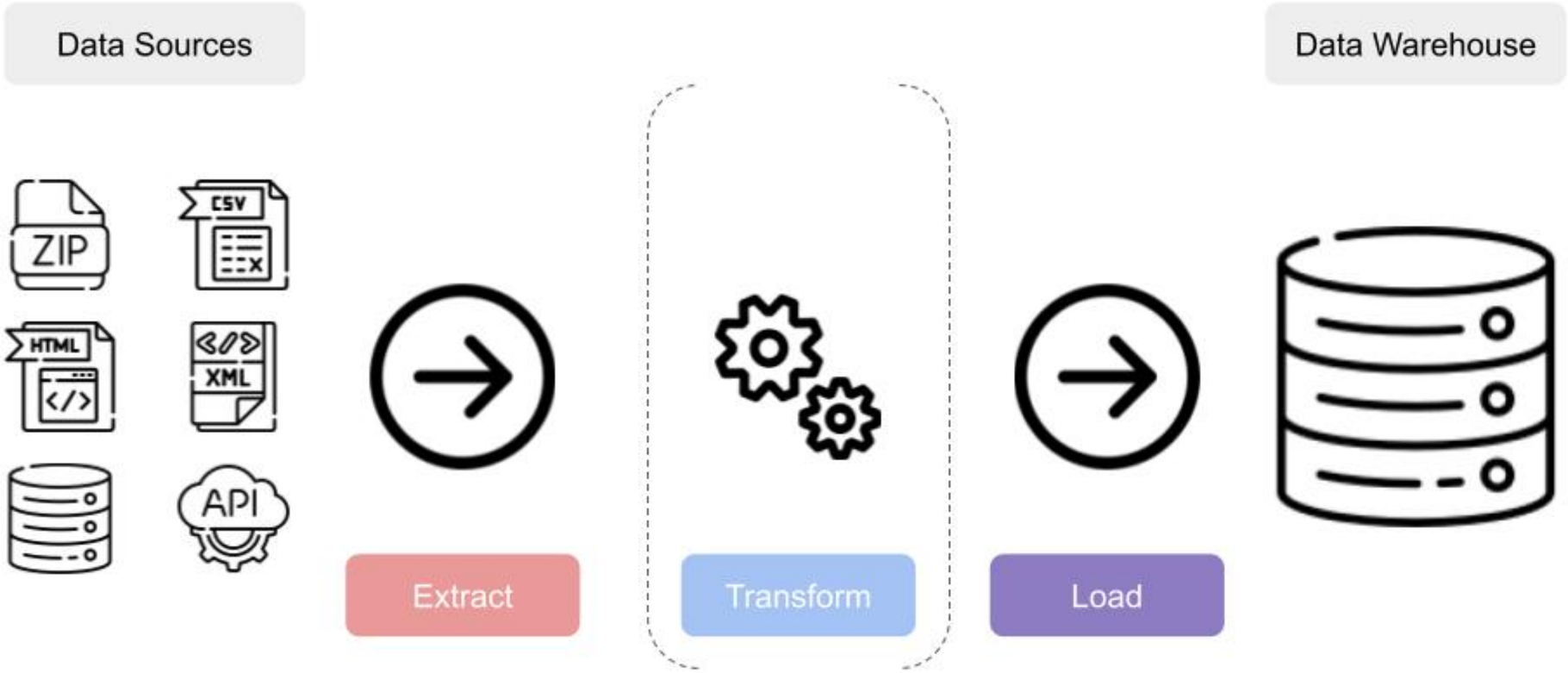
ETL IN PYTHON

Wrap-up

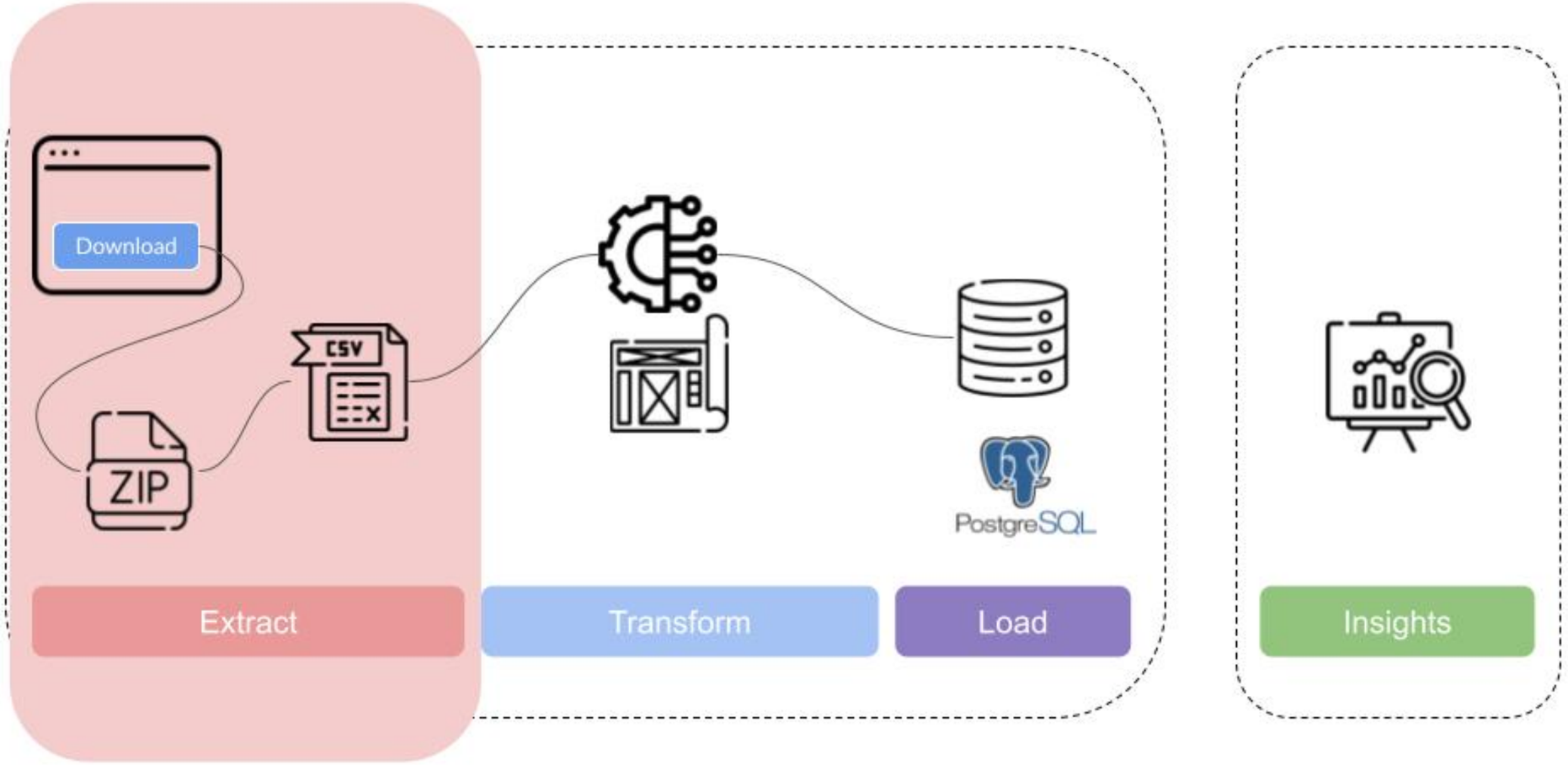
ETL IN PYTHON

Stefano Francavilla
CEO - Geowox

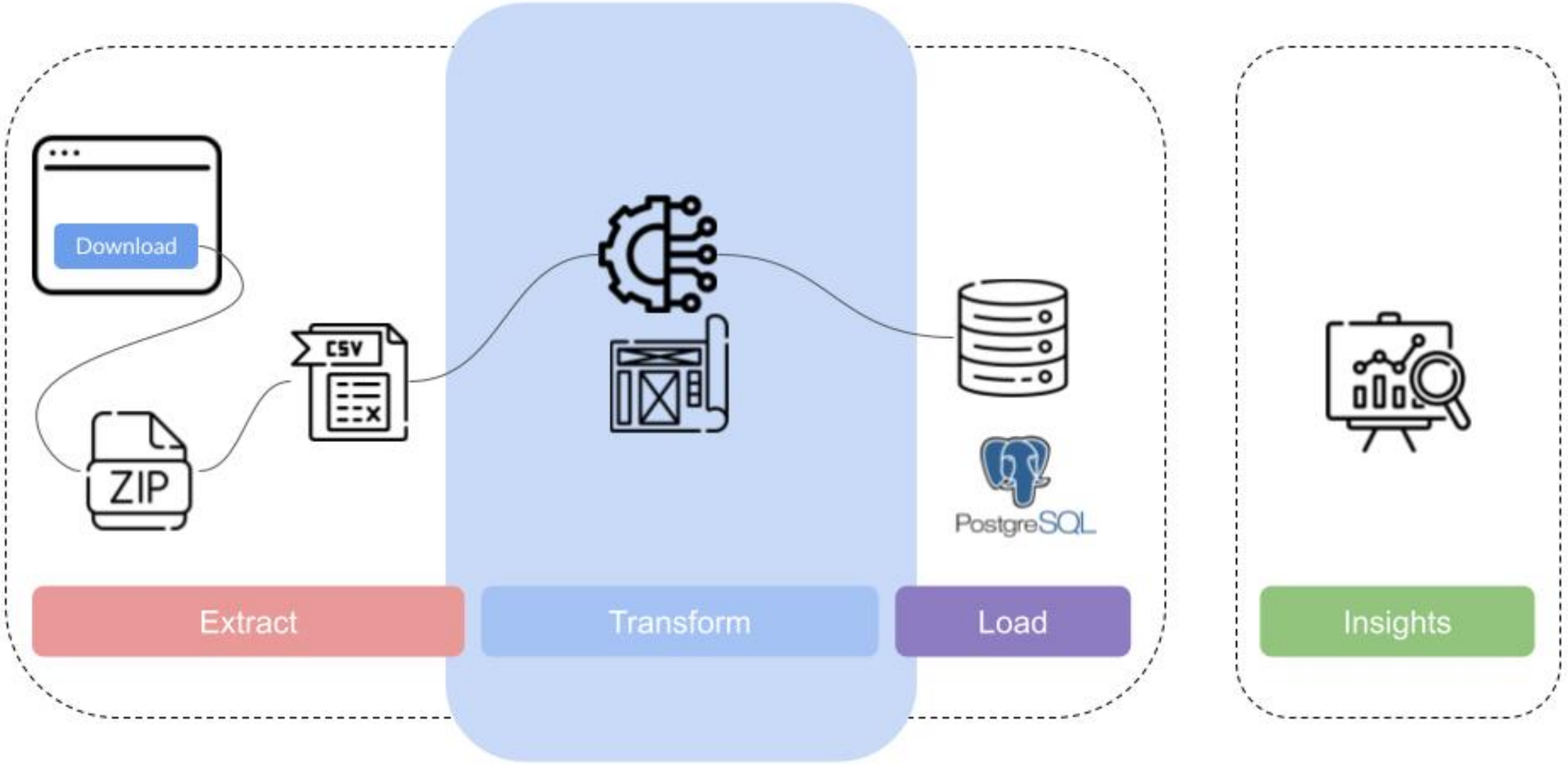
Explore the data and extract



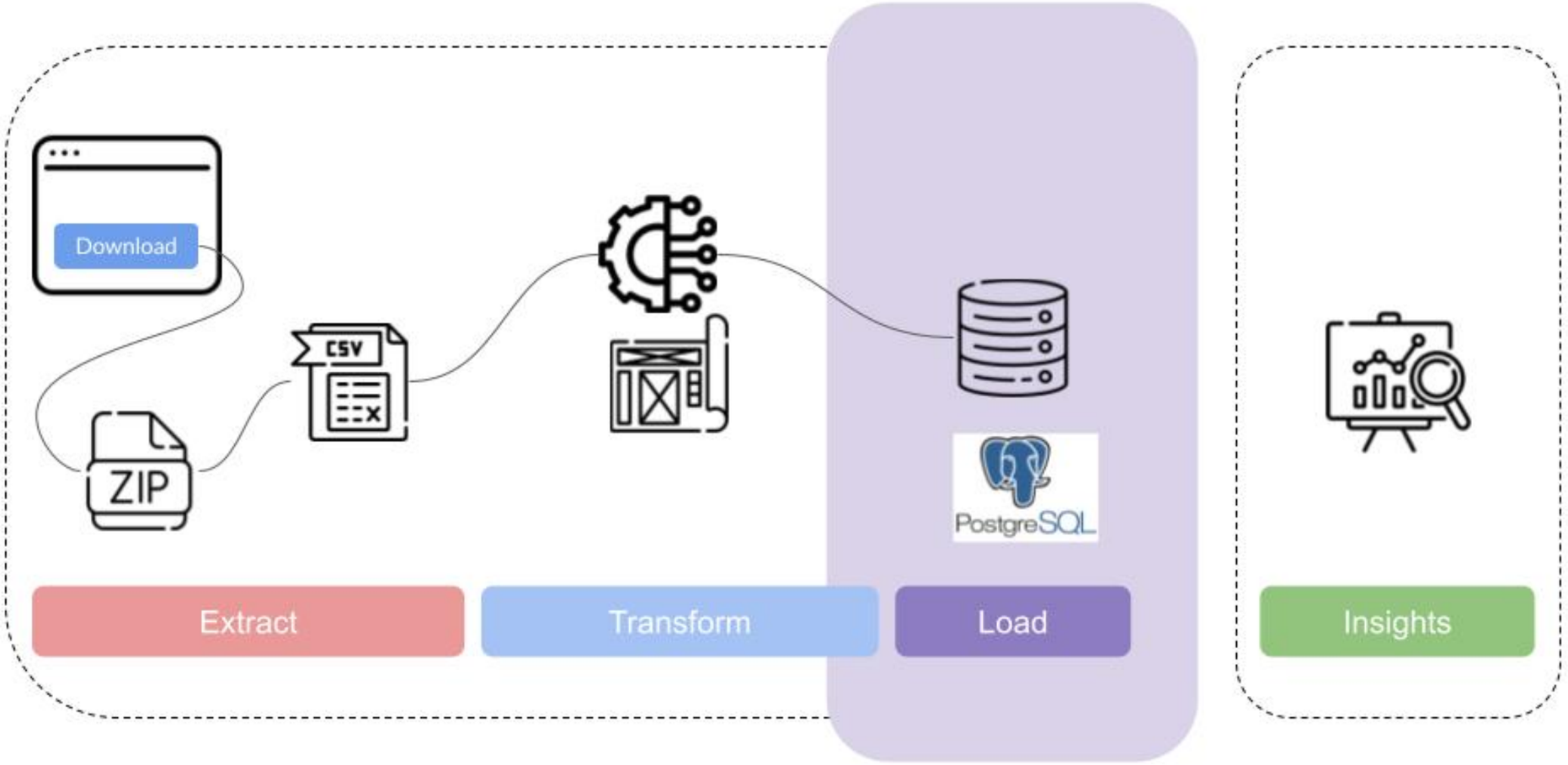
Explore the data and extract



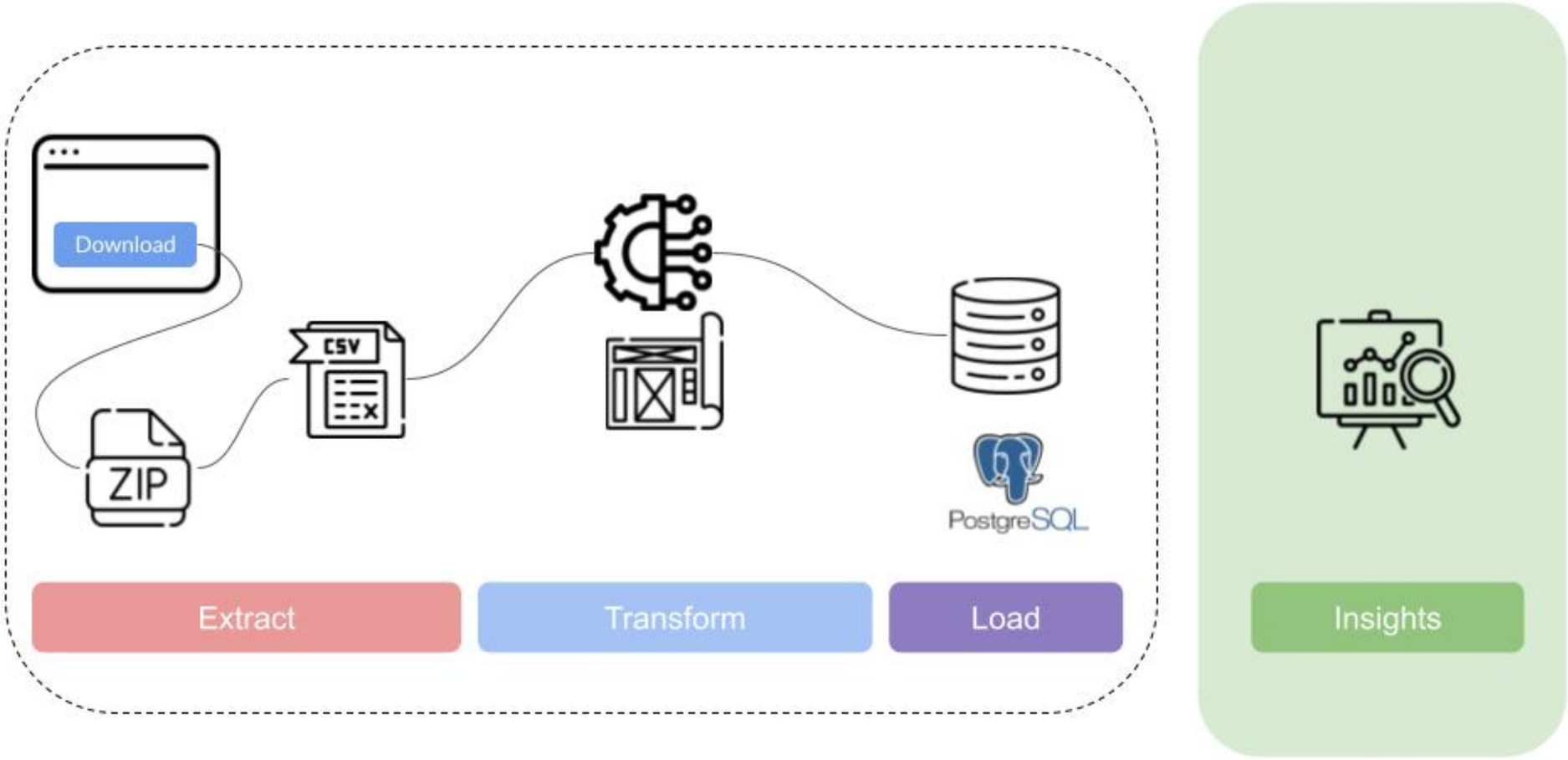
ETL foundations and tranform



From raw to clean: load



From clean data to meaningful insights



Thank you!

ETL IN PYTHON