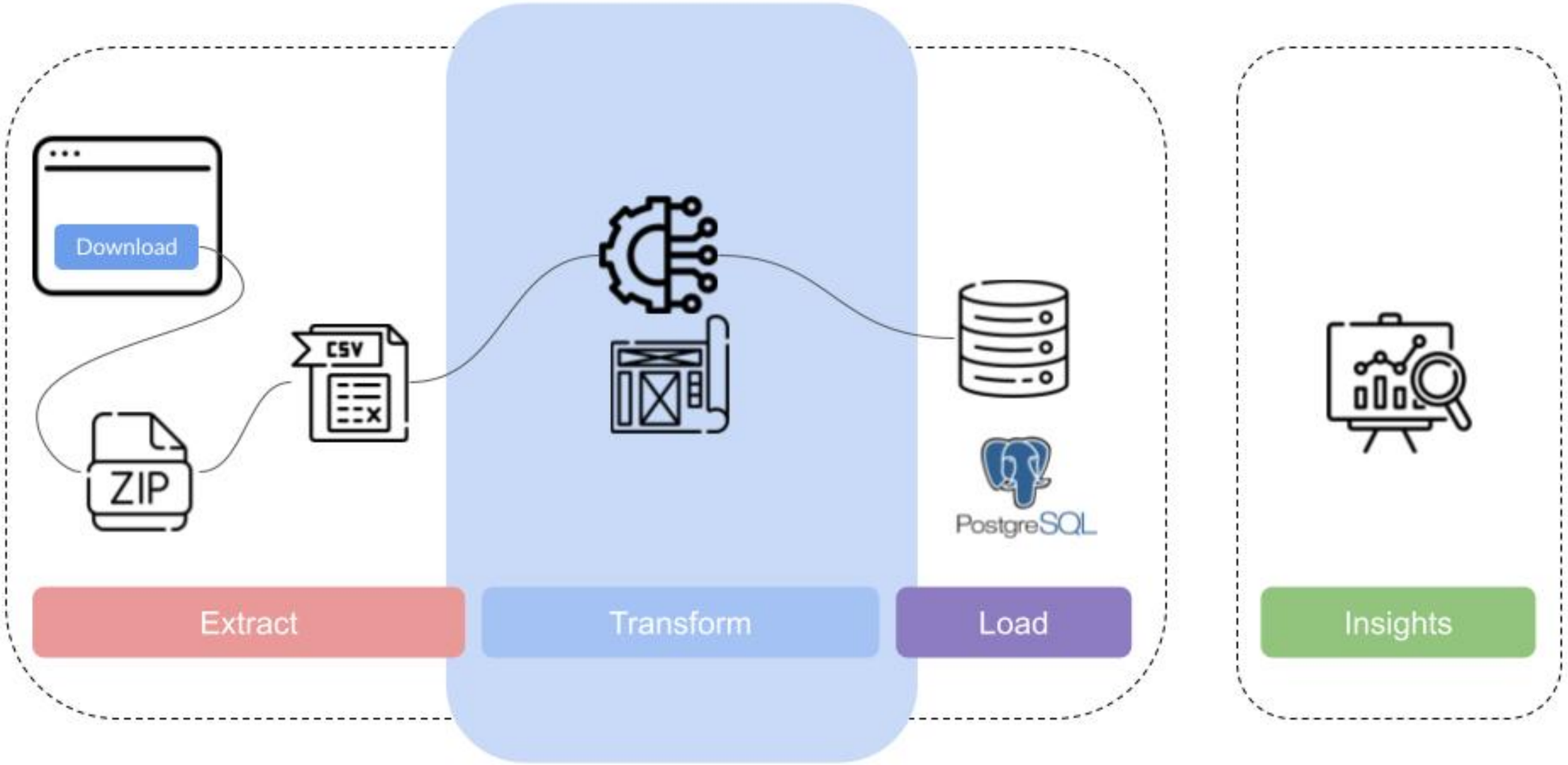
Let's talk with the database

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

database+dialect

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

SQLAlchemy engines

database+dialect

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

username:password

SQLAlchemy engines

database+dialect

database_url:port

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

username:password

SQLAlchemy engines

database+dialect

database_url:port

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

username:password

database_name

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

datacamp

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

```python
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(255)

```python
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

datacamp

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

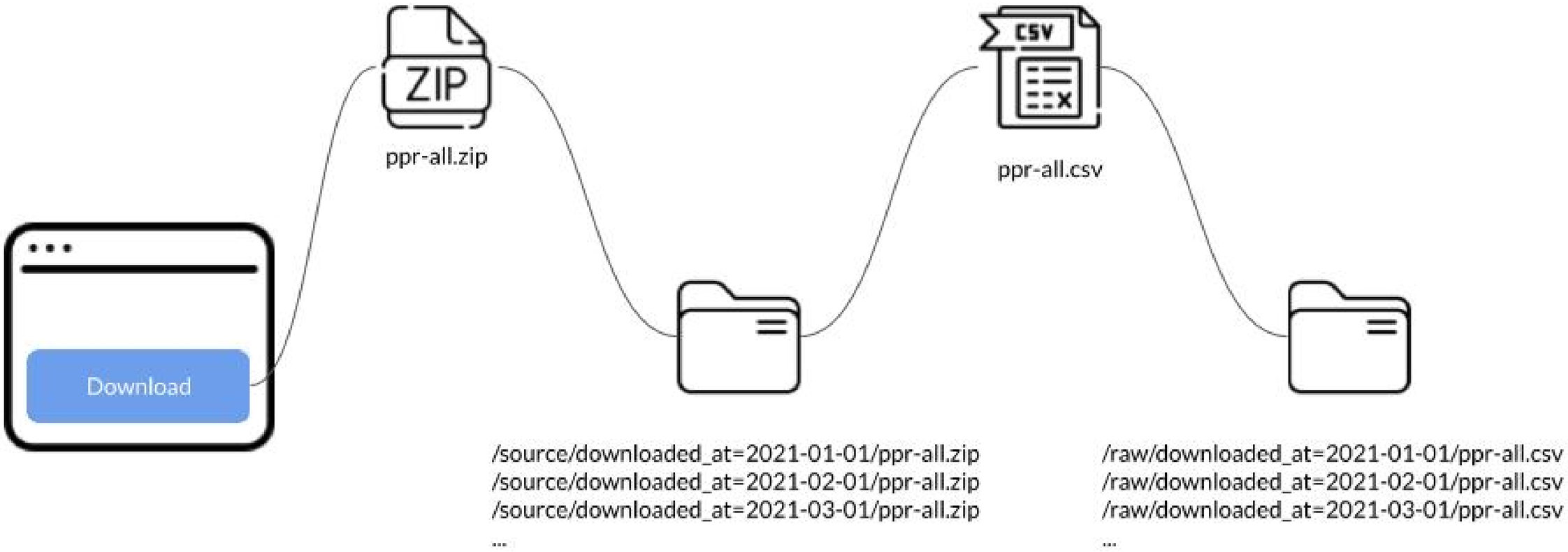
Put transform operations together

ETL IN PYTHON

Stefano Francavilla
CEO - Geowox

datacamp

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

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

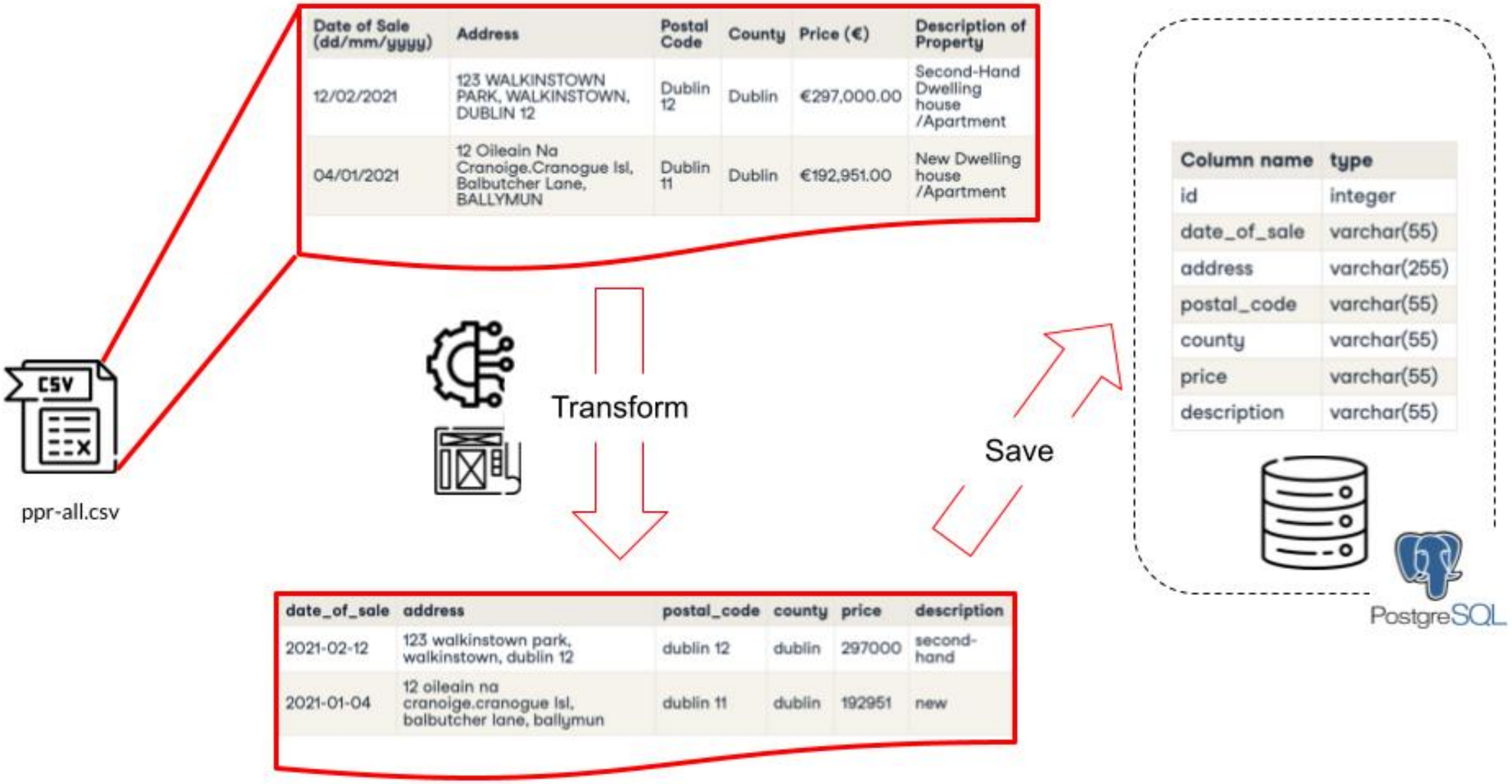
Date of Sale (dd/mm/yyyy)	Address	Postal Code	County	Price (€)	Description of Property
12/02/2021	123 WALKINSTOWN PARK, WALKINSTOWN, DUBLIN 12	Dublin 12	Dublin	€297,000.00	Second-Hand Dwelling house /Apartment
04/01/2021	12 Oileain Na Cranoige.Cranogue Isl, Balbutcher Lane, BALLYMUN	Dublin 11	Dublin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

ppr-all.csv

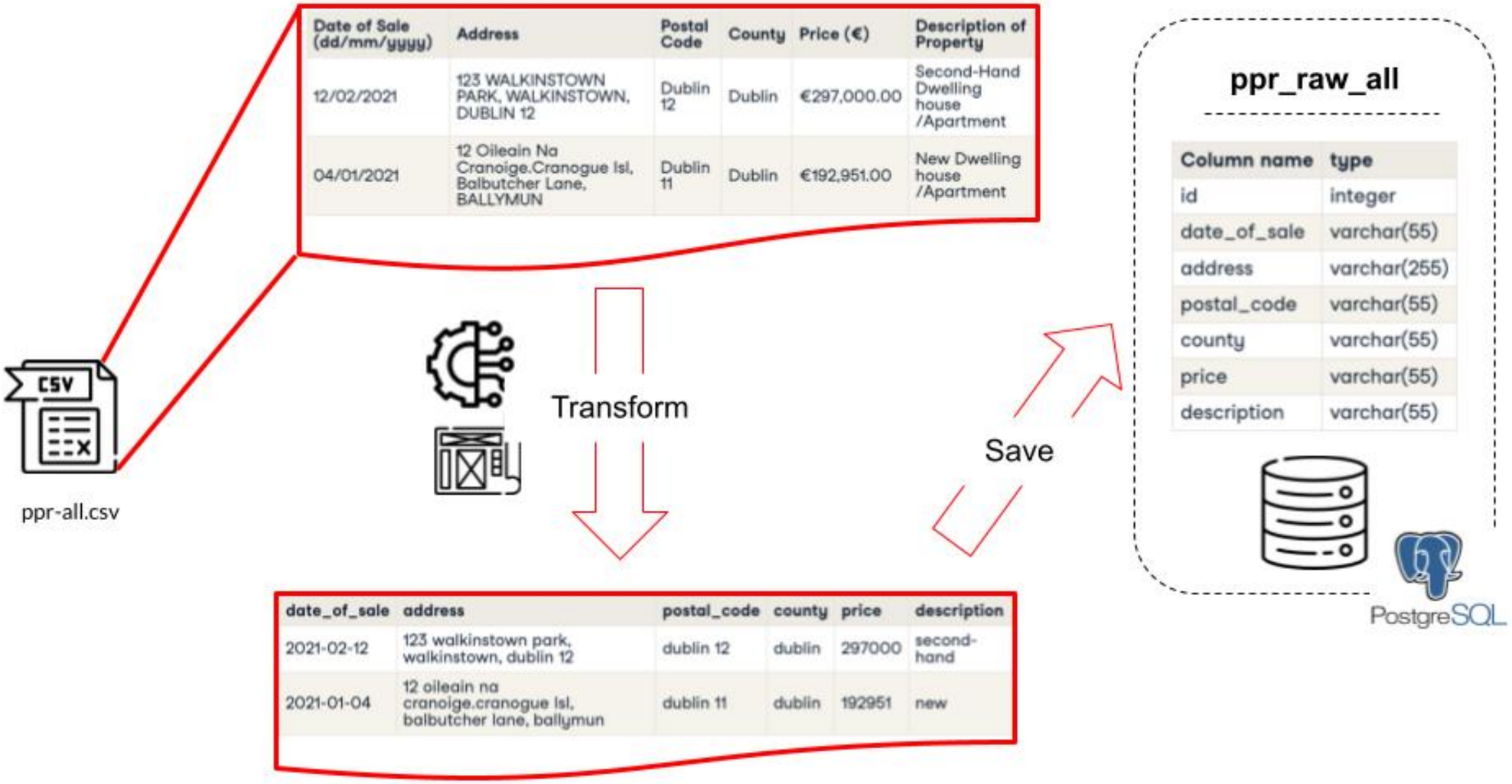
Transform

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

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