Data engineering

Accusat'IF

★ CHAUDRON Félicie & MOULINAS Margaux ★

Topic and goals

- Actors and movie crew members legal problems are often talked about
- Often “Cancel culture” is brought up as well
- But are there real consequences ?

Our questions :

- Do the the most popular movies have “problematic” people in their cast ?
- Is there a genre that attracts problematic people ?
- Are actors the only ones getting in legal trouble ?

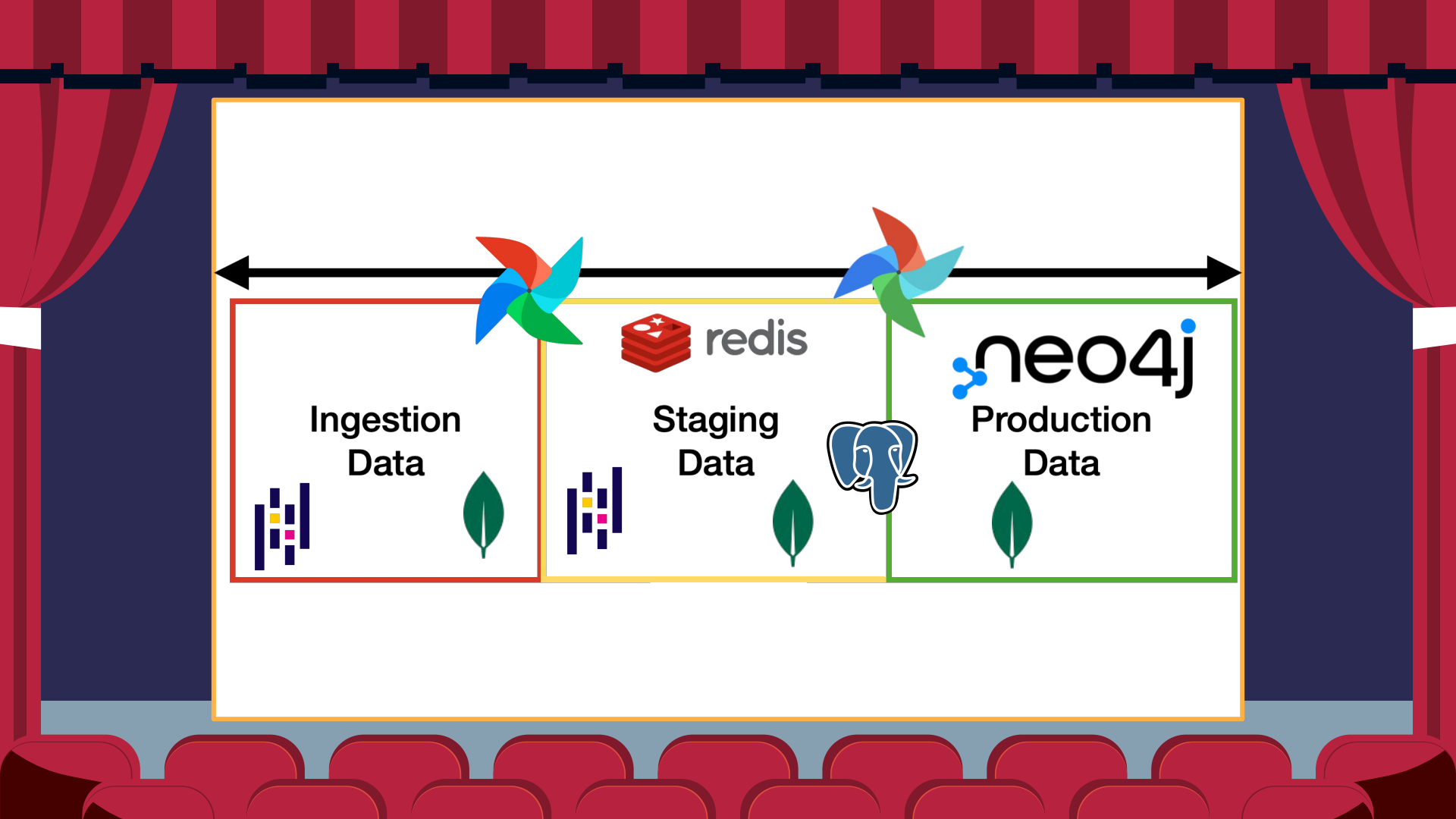
What are our sources

IMDb

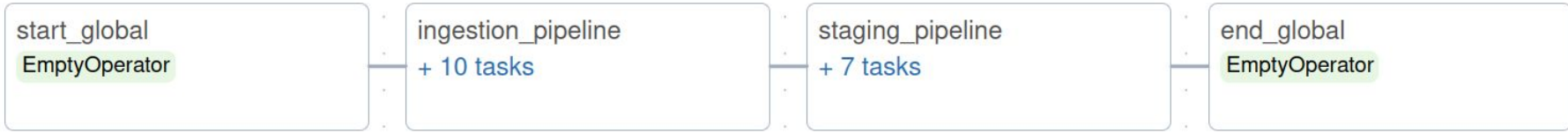
The **I**nternet **M**ovie **D**atabase
Data about movies in CSV format

Wikidata

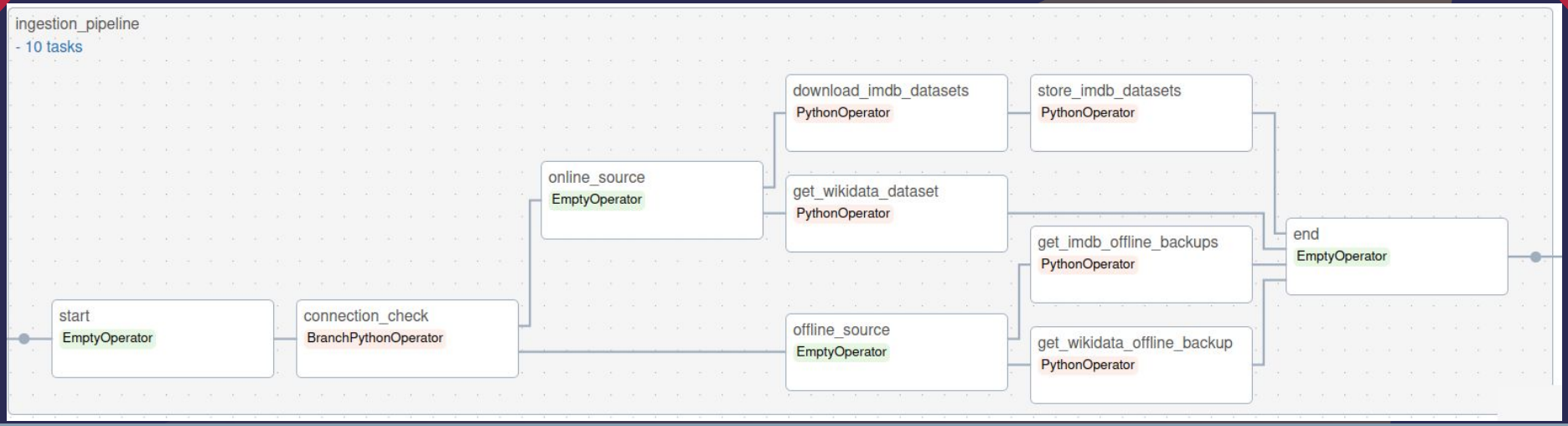
A graph database that normalizes part of the data available on wikipedia


redis
neo4j
Ingestion Data
Staging Data
Production Data

Our Pipelines



Data ingestion



Check connection

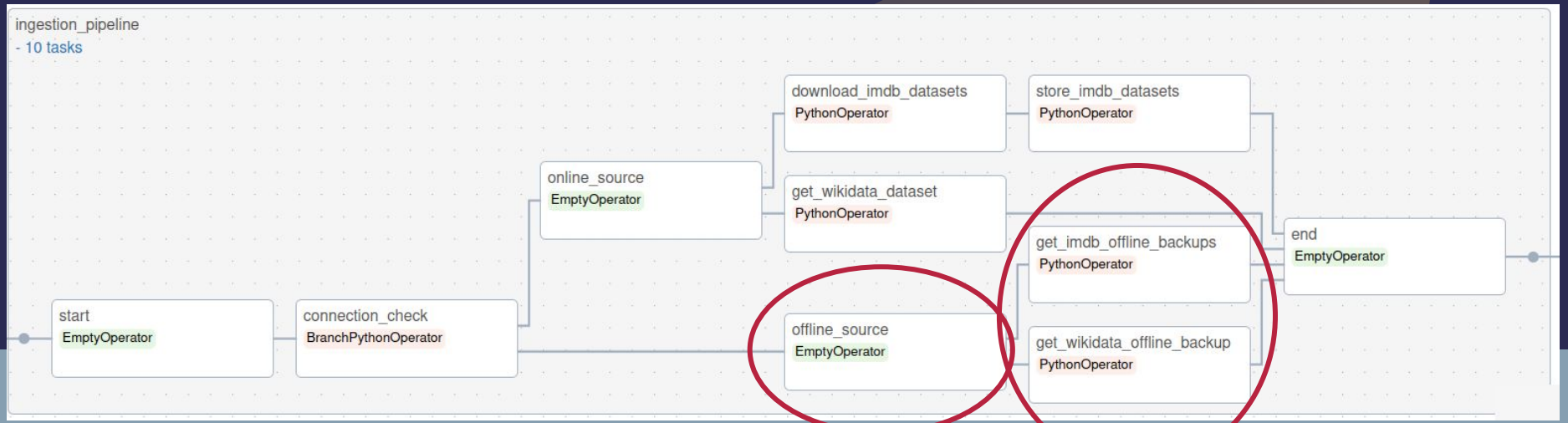
We use a BranchPythonOperator

- Online source: get our data from the internet
- Offline source: get our data from saved samples

Offline source

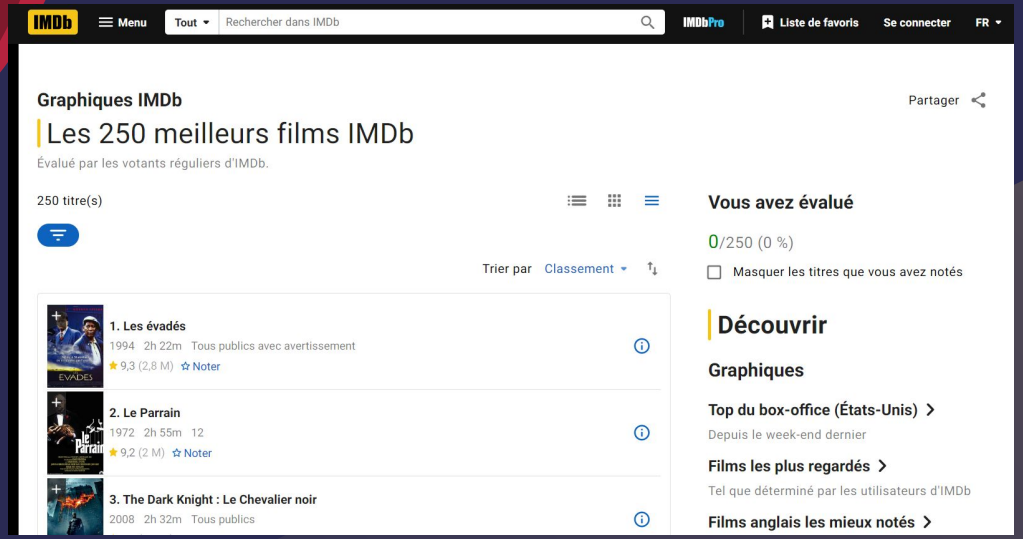
We have saves samples data in csv

- Open them in pandas
- Store them in mongoDB



IMDb dataset ingestion

IMDb dataset ingestion

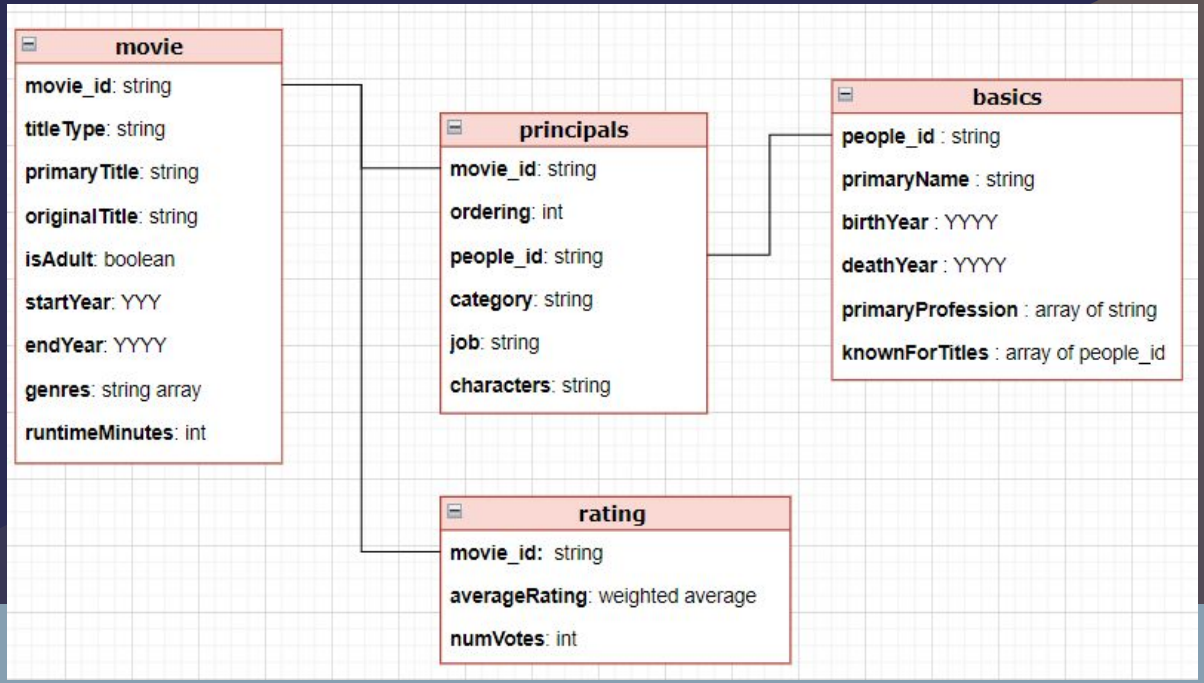


- ★ Get our data from IMDB website which provides an online database of informations related to movies, actors, producers...
- ★ Download 4 tables in zip format
- ★ Open them in pandas and store them in MongoDB

pandas

MongoDB.

IMDb dataset ingestion



Offence data ingestion

Offence data ingestion

Problems :

- ★ Legal troubles aren't a black and white thing
- ★ Can be hard to find extensive, reliable, structured data on the matter

Ideas :

- ★ Exploring social media data
  - ⇒ Large amount of unstructured data, requires a lot of processing
- ★ Exploring news articles data
  - ⇒ Could be more structured but couldn't find a usable source in time
- ★ Using the data available on wikidata
  - ⇒ Structured and easy access but data isn't extensive

Offence data ingestion

```
SELECT DISTINCT ?personLabel ?offenceLabel ?offence WHERE {
  ?person p:P106 _:b168.
  _:b168 ps:P106 ?metier.
  ?metier p:P31 _:b169.
  _:b169 ps:P31 wd:Q4220920.
  {
    ?person p:P1399 _:b171.
    _:b171 ps:P1399 ?offence.
  }
  UNION
  {
    ?person p:P1344 ?statement0.
    ?statement0 ps:P1344 ?offence.
    ?offence p:P31 _:b167.
    _:b167 (ps:P31/(wdt:P279*)) wd:Q1456832.
  }
  UNION
  {
    ?person p:P793 ?statement0.
    ?statement0 ps:P793 ?offence.
    ?offence p:P31 _:b187.
    _:b187 (ps:P31/(wdt:P279*)) wd:Q8016240.
  }
  SERVICE wikibase:label {
    bd:serviceParam wikibase:language "en,fr".
    ?person rdfs:label ?personLabel.
    ?offence rdfs:label ?offenceLabel.
  }
}
ORDER BY (?personLabel)
```

?person has occupation ?metier
?metier is an instance of filmmaking occupation

?person was convicted of ?offence

?person has important event ?offence
?offence is an instance of legal case

?person participates in ?offence
?offense is an instance of legal offence(infraction)

WIKIDATA

Offence data ingestion

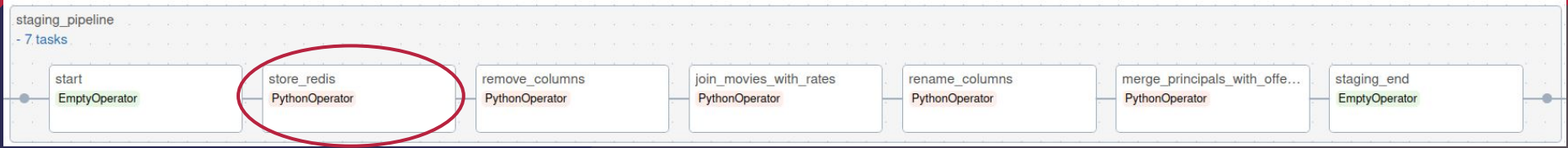
- ★ Only 233 lines
- ★ For each person/offence tuple so even less people (~190)

Data wrangling

- ★ Clean data
- ★ Remove unimportant data
- ★ Merge our sources
- ★ Save in MongoDB
- ★ ...

pandas redis MongoDB

Store in redis

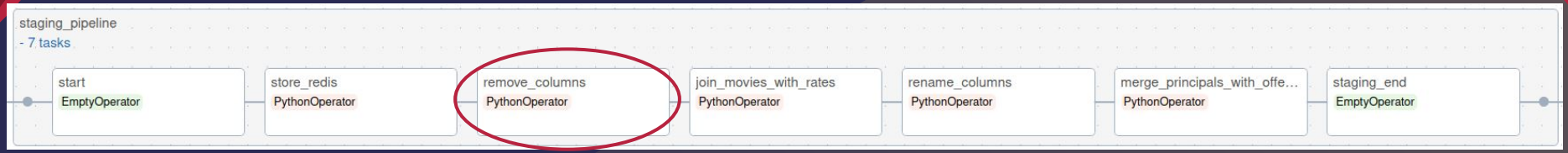


- ★ Use REDIS throughout the wrangling phase
- ★ Reduce data access time
- ★ Improve performance
- ★ Facilitate/Accelerate storage between each task in our wrangling pipeline

pandas

redis

Remove columns



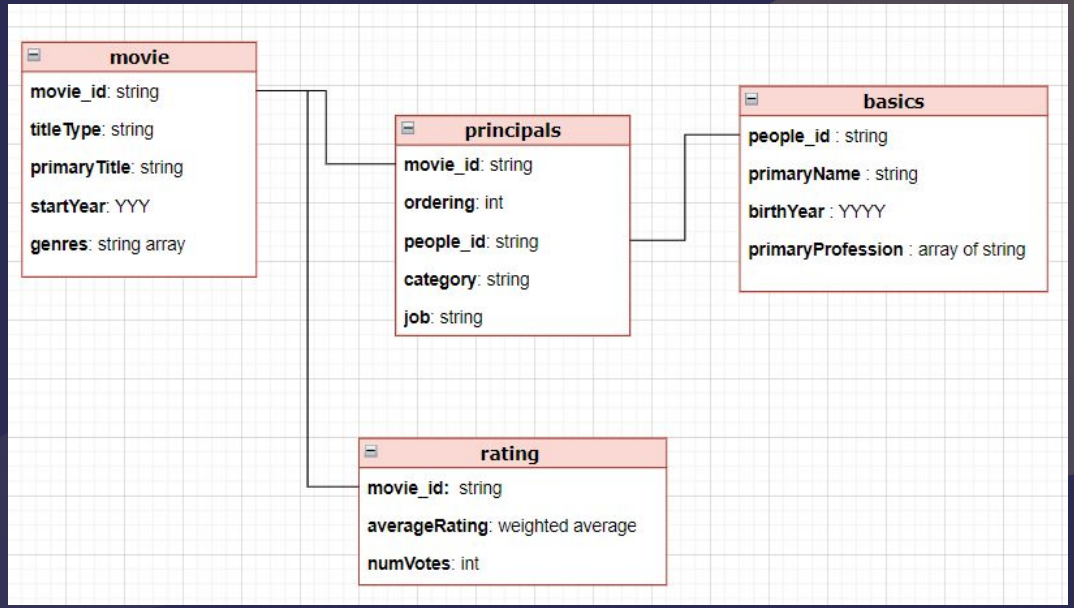
Remove unimportant columns :

- death Year,
- runtime of the movie
- isAdult field
- ...

pandas

redis

Remove IMDB columns

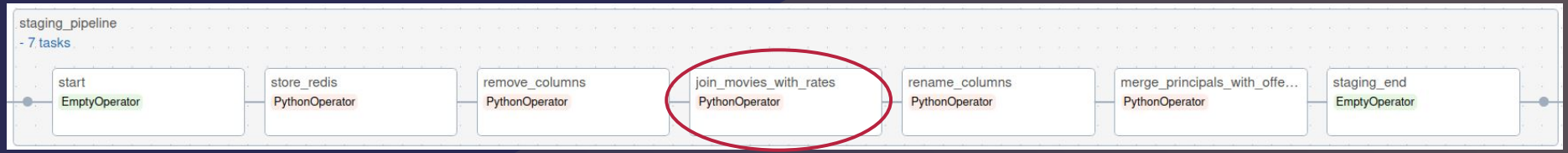


pandas

redis

Join movies with rates

We join our movie table with rate table on the id field to have a unique dataset

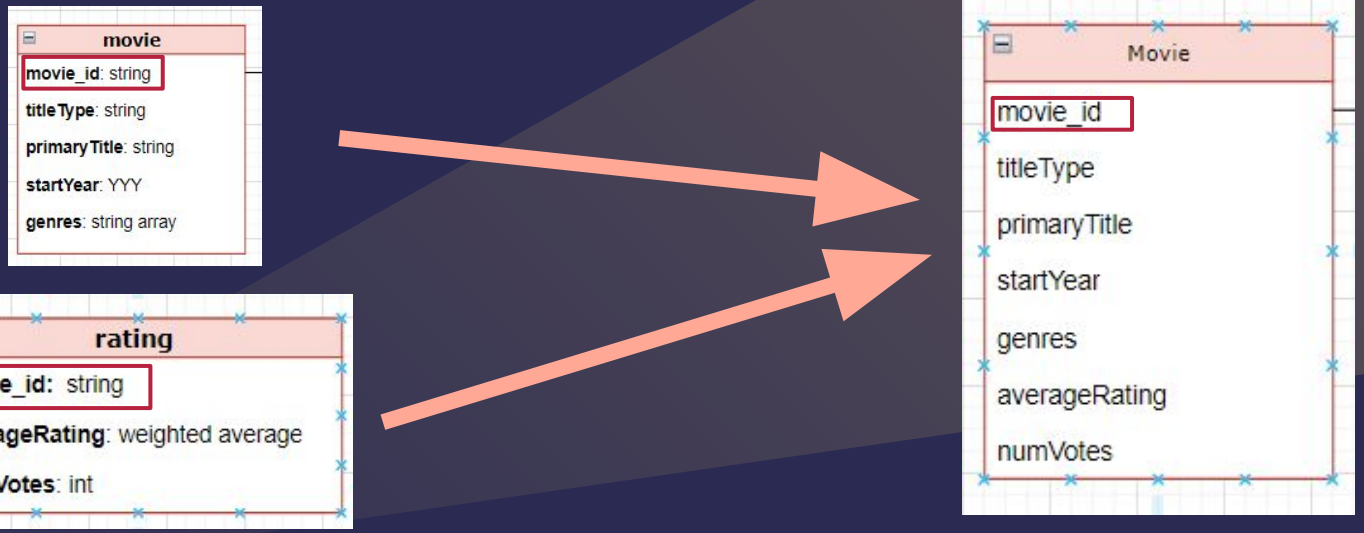



```
df_movie = df_movie.set_index('_id').join(df_rate.set_index('_id'))
```

redis

pandas

Join movies with rates

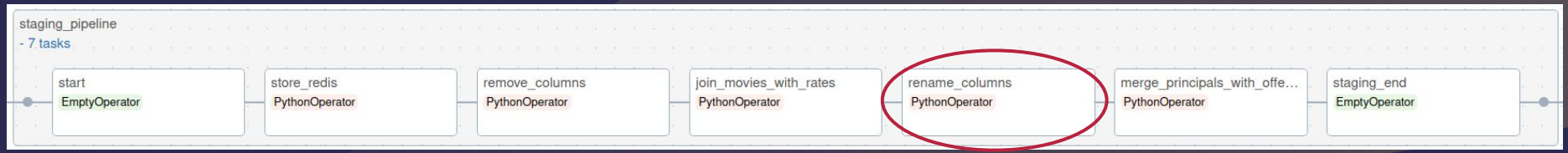


redis

pandas

Rename columns

We clean our data by renaming our columns field to have more understandable data

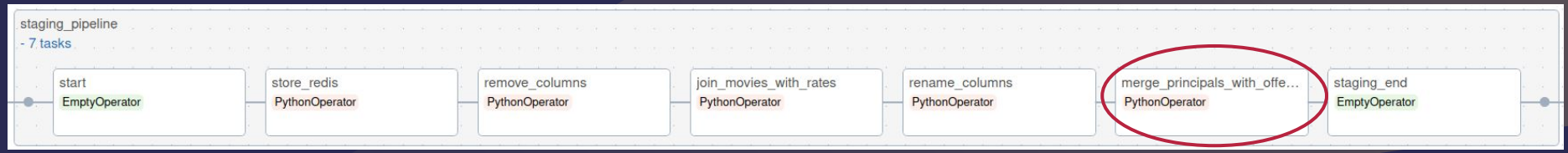


redis

Merge wikidata and IMDB dataset

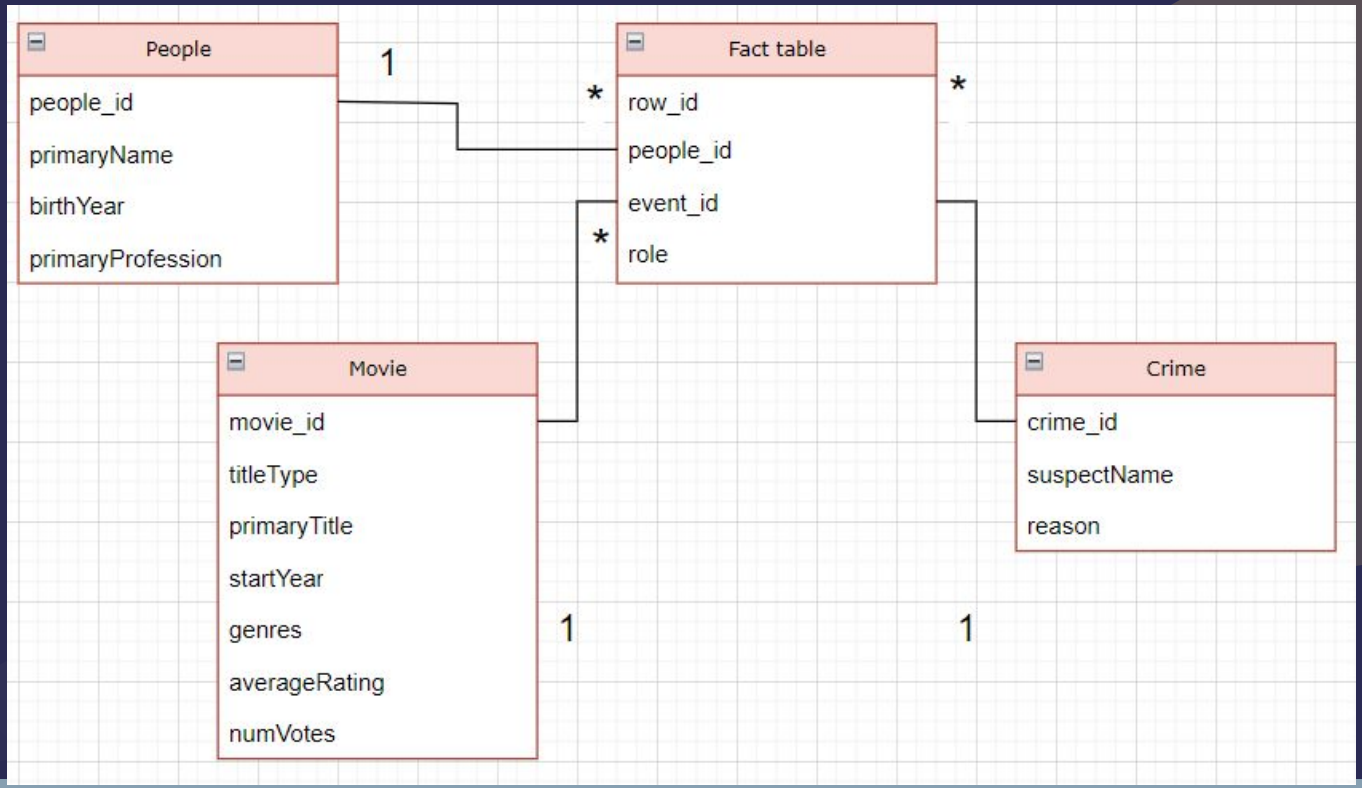
We merge our imdb datasets with offense dataset

```
df_fact = pd.merge(df_offense, df_people[['_id', 'primaryName']], how="left", left_on='suspectName', right_on='primaryName')
```



pandas

redis


People
people_id
primaryName
birthYear
primaryProfession
1
*
Fact table
row_id
people_id
event_id
role
*
*
Movie
movie_id
titleType
primaryTitle
startYear
genres
averageRating
numVotes
1
Crime
crime_id
suspectName
reason
1

Plans for the production data

- Store our data in Postgresql to make it persistent
- Use neo4j to create a queryable graph

Overview of the tools we used

- **MongoDB** : store and query the cleaned data
- **Pandas** : extract and stage the data
- **Redis** : better performances during staging phase
- **PostgreSQL : Store our data**
- **SPARQL : query Wikidata**
- **Neo4j : Production data in graph form**

Thank you!

Do you have any questions?