Créer et utiliser une base de données

Movie Db

Pierre, Thierno, Bastien, Laurent

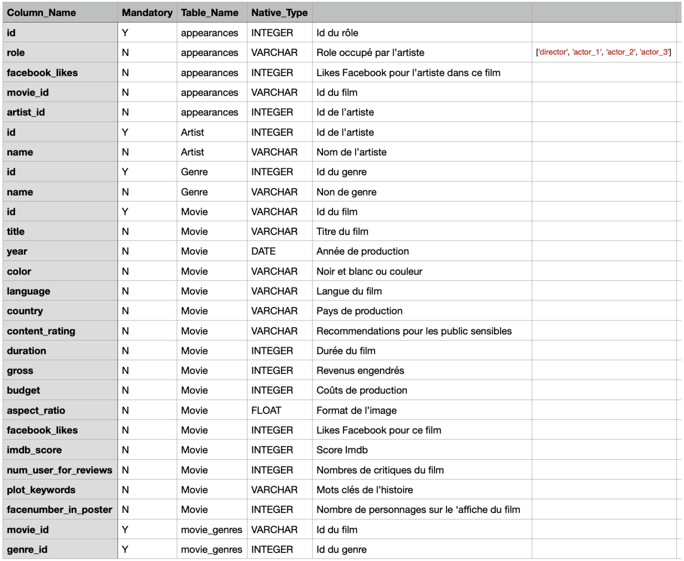
14 février 2022

Outline

Données en entrée

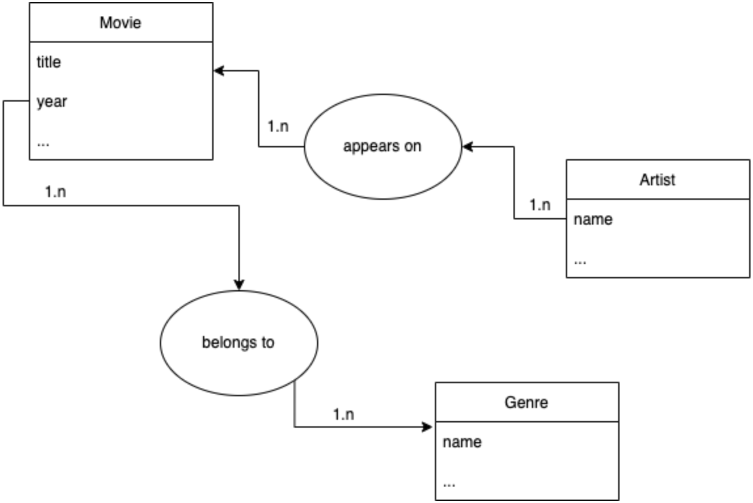
	count	unique	top	freq	mean
color	5024	2	Color	4815	nan
director_name	4939	2398	Steven Spielberg	26	nan
num_critic_for_reviews	4993	nan	nan	nan	140.194
duration	5028	nan	nan	nan	107.201
director_facebook_likes	4939	nan	nan	nan	686.509
actor_3_facebook_likes	5020	nan	nan	nan	645.01
actor_2_name	5030	3032	Morgan Freeman	20	nan
actor_1_facebook_likes	5036	nan	nan	nan	6560.05
gross	4159	nan	nan	nan	4.84684e+07
genres	5043	914	Drama	236	nan
actor_1_name	5036	2097	Robert De Niro	49	nan
movie_title	5043	4917	Ben-Hur	3	nan
num_voted_users	5043	nan	nan	nan	83668.2
cast_total_facebook_likes	5043	nan	nan	nan	9699.06
actor_3_name	5020	3521	John Heard	8	nan
facenumber_in_poster	5030	nan	nan	nan	1.37117
plot_keywords	4890	4760	based on novel	4	nan
movie_imdb_link	5043	4919	`http://www.imdb.com/title/tt0232500/?ref_=fn_tt_tt_1`	3	nan
num_user_for_reviews	5022	nan	nan	nan	272.771
language	5031	47	English	4704	nan
country	5038	65	USA	3807	nan
content_rating	4740	18	R	2118	nan
budget	4551	nan	nan	nan	3.97526e+07
title_year	4935	nan	nan	nan	2002.47
actor_2_facebook_likes	5030	nan	nan	nan	1651.75
imdb_score	5043	nan	nan	nan	6.44214
aspect_ratio	4714	nan	nan	nan	2.2204
movie_facebook_likes	5043	nan	nan	nan	7525.96

Dictionnaire des données

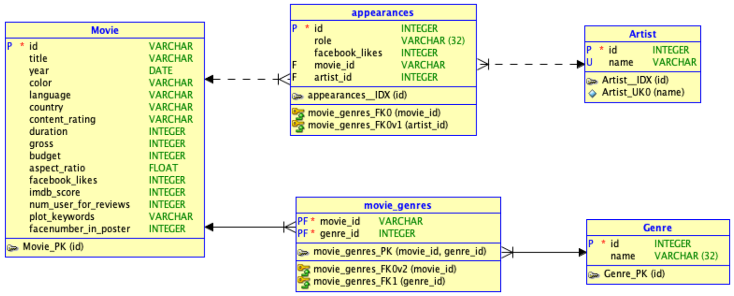


Column_Name	Mandatory	Table_Name	Native_Type		
id	Y	appearances	INTEGER	Id du rôle	
role	N	appearances	VARCHAR	Role occupé par l'artiste	['director', 'actor_1', 'actor_2', 'actor_3']
facebook_likes	N	appearances	INTEGER	Likes Facebook pour l'artiste dans ce film	
movie_id	N	appearances	VARCHAR	Id du film	
artist_id	N	appearances	INTEGER	Id de l'artiste	
id	Y	Artist	INTEGER	Id de l'artiste	
name	N	Artist	VARCHAR	Nom de l'artiste	
id	Y	Genre	INTEGER	Id du genre	
name	N	Genre	VARCHAR	Non de genre	
id	Y	Movie	VARCHAR	Id du film	
title	N	Movie	VARCHAR	Titre du film	
year	N	Movie	DATE	Année de production	
color	N	Movie	VARCHAR	Noir et blanc ou couleur	
language	N	Movie	VARCHAR	Langue du film	
country	N	Movie	VARCHAR	Pays de production	
content_rating	N	Movie	VARCHAR	Recommendations pour les public sensibles	
duration	N	Movie	INTEGER	Durée du film	
gross	N	Movie	INTEGER	Revenus engendrés	
budget	N	Movie	INTEGER	Coûts de production	
aspect_ratio	N	Movie	FLOAT	Format de l'image	
facebook_likes	N	Movie	INTEGER	Likes Facebook pour ce film	
imdb_score	N	Movie	INTEGER	Score Imdb	
num_user_for_reviews	N	Movie	INTEGER	Nombres de critiques du film	
plot_keywords	N	Movie	VARCHAR	Mots clés de l'histoire	
facenumber_in_poster	N	Movie	INTEGER	Nombre de personnages sur le 'affiche du film	
movie_id	Y	movie_genres	VARCHAR	Id du film	
genre_id	Y	movie_genres	INTEGER	Id du genre	

MCD


Movie
title
year
...
1.n
appears on
Artist
1.n
name
...
1.n
belongs to
1.n
Genre
name
...

Modèle logique de données



Movie

P	*	id	VARCHAR
		title	VARCHAR
		year	DATE
		color	VARCHAR
		language	VARCHAR
		country	VARCHAR
		content_rating	VARCHAR
		duration	INTEGER
		gross	INTEGER
		budget	INTEGER
		aspect_ratio	FLOAT
		facebook_likes	INTEGER
		imdb_score	INTEGER
		num_user_for_reviews	INTEGER
		plot_keywords	VARCHAR
		facenumber_in_poster	INTEGER

Movie_PK (id)

appearances

P	*	id	INTEGER
		role	VARCHAR (32)
		facebook_likes	INTEGER
F		movie_id	VARCHAR
F		artist_id	INTEGER

appearances__IDX (id)

movie_genres_FK0 (movie_id)
movie_genres_FK0v1 (artist_id)

Artist

P	*	id	INTEGER
U		name	VARCHAR

Artist__IDX (id)
Artist_UK0 (name)

movie_genres

PF	*	movie_id	VARCHAR
PF	*	genre_id	INTEGER

movie_genres_PK (movie_id, genre_id)

movie_genres_FK0v2 (movie_id)
movie_genres_FK1 (genre_id)

Genre

P	*	id	INTEGER
		name	VARCHAR (32)

Genre_PK (id)

Création de la base de données (SQLAlchemy)

```
class Movie(Base):
    __tablename__ = 'Movie'
    id = Column(String, primary_key=True, nullable=False)
    title = Column(String)
    year = Column(Date)
    color = Column(String)
    language = Column(String)
    country = Column(String)
    content_rating = Column(String)
    duration = Column(Integer)
    gross = Column(Integer)
    budget = Column(Integer)
    aspect_ratio = Column(Float)
    facebook_likes = Column(Integer)
    imdb_score = Column(Integer)
    num_user_for_reviews = Column(Integer)
    plot_keywords = Column(String)
    facenumber_in_poster = Column(Integer)
    genres = relationship('Genre', secondary = 'movie_genres', back_populates="movies")
    cast = relationship('Artist', secondary = 'appearances', back_populates="movies")
```

Création de la base de données (SQL)

```sql
CREATE TABLE appearances (
        id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        role VARCHAR(32),
        facebook_likes INTEGER,
        movie_id VARCHAR,
        artist_id INTEGER,
        FOREIGN KEY(movie_id) REFERENCES "Movie" (id),
        FOREIGN KEY(artist_id) REFERENCES "Artist" (id)
)
;
```

Chargement des donneés

- Extrait du script d'initialisation et de chargement de la base :

```python
roles = ['director', 'actor_1', 'actor_2', 'actor_3']

for i, row in df.iterrows():
    for role in roles:
        a = eval(f'row.{role}_name')
        q = s.query(Artist).filter(Artist.name==a)
        if s.query(q.exists()).scalar():
            artist_id = q.first().id
        else:
            artist = Artist(**{
                'name': eval(f'row.{role}_name')
            })
            s.add(artist)
            s.commit()
            artist_id = q.first().id
```

le top 10 des films les plus rentables

```
SELECT title,gross-budget FROM Movie ORDER BY gross-budget DESC LIMIT 10
```

title	gross-budget
Avatar	523505847
Jurassic World	502177271
Titanic	458672302
Star Wars : Episode IV - A New Hope	449935665
E.T. the Extra-Terrestrial	424449459
The Lion King	377783777
Star Wars : Episode I - The Phantom Menace	359544677
The Dark Knight	348316061
The Hunger Games	329999255
Deadpool	305024263

le top 10 des films les moins rentables

```
SELECT title,gross-budget FROM Movie ORDER BY gross-budget ASC LIMIT 10
```

The Host	-12213298588
Lady Vengeance	-4199788333
Fateless	-2499804112
Princess Mononoke	-2397701809
Steamboy	-2127109510
Akira	-1099560838
Godzilla 2000	-989962610
Tango	-698312689
Kabhi Alvida Naa Kehna	-696724557
Red Cliff	-553005191

les réalisateurs qui ont fait le plus de films

```
SELECT
a.name, COUNT(*)
FROM
Artist a
INNER JOIN
(Movie m INNER JOIN appearances p ON m.id = p.movie_id) ON a.id = p.artist_id
WHERE role='director'
GROUP BY a.name
ORDER BY COUNT(*) DESC LIMIT 10;
```

Steven Spielberg	26
Woody Allen	22
Martin Scorsese	20
Clint Eastwood	20
Spike Lee	16
Ridley Scott	15
Renny Harlin	15
Steven Soderbergh	14
Oliver Stone	14
Ron Howard	13

l'acteur qui a joué dans le plus de films

```sql
SELECT
a.name, COUNT(*)
FROM
Artist a
INNER JOIN
(Movie m INNER JOIN appearances p ON m.id = p.movie_id) ON a.id = p.artist_id
WHERE role='actor_1' OR role='actor_2' OR role='actor_3'
GROUP BY a.name
ORDER BY COUNT(*) DESC LIMIT 1;
```

Robert De Niro 51

le nombre de films avec "love" dans les mots clés

```
SELECT COUNT(*) From movie WHERE plot_keywords LIKE '% love,%'
```

```
156
```

le nombre de films français

```
SELECT COUNT(*) From movie WHERE language = 'French'
```

```
64
```

le réalisateur avec au moins 10 films qui obtient la meilleure moyenne (note imdb)

```
SELECT a.name, AVG(m.imdb_score), COUNT(*)
FROM movie m INNER JOIN appearances p
ON m.id = p.movie_id INNER JOIN artist a
ON p.artist_id = a.id WHERE p.role='director'
GROUP BY a.name HAVING COUNT(m.id) > 9
ORDER BY AVG(m.imdb_score) DESC LIMIT 1
```

David Fincher	7.75	10

References

- https://github.com/lsiksous/recsys