

Créer et utiliser  
une base de  
données sur les  
films



# Sommaire

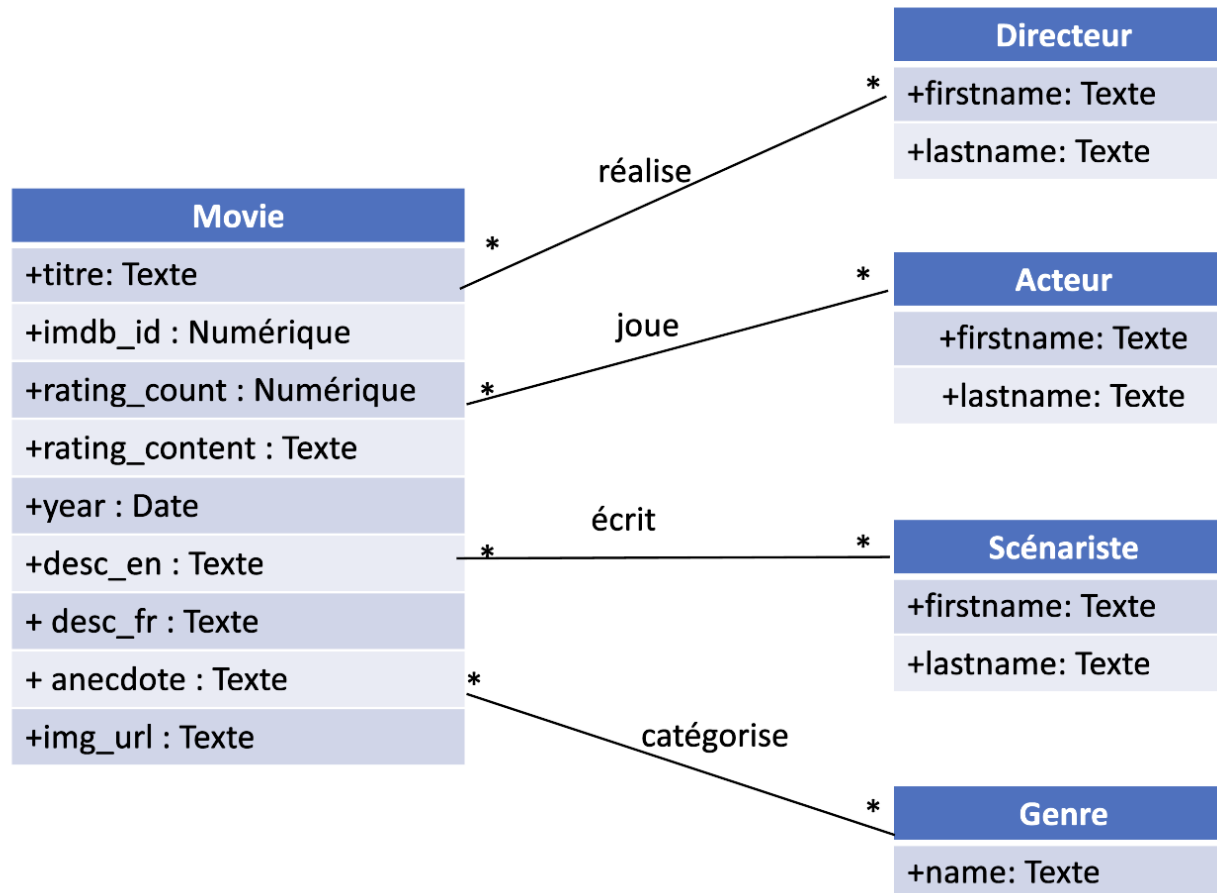
1. Dictionnaire des données
2. Modèle conceptuel (UML)
3. Schéma relationnel 3NF
4. Implémentation de la base de données
5. Réalisation de l'API
6. Requêtes et résultats



# 1. Dictionnaire des données

Numéro	Code propriété	Signification	Type	Observation
1	titre	titre du film	texte	Longueur : 128
2	act.firstname	prénom de l'acteur	texte	Longueur : 128
3	act.lastname	nom de famille de l'acteur	texte	Longueur : 128
4	sce.firstname	prénom du scénariste	texte	Longueur : 128
5	sce.lastname	nom de famille du scénariste	texte	Longueur : 128
6	dir.firstname	prénom du directeur	texte	Longueur : 128
7	dir.lastname	nom de famille du directeur	texte	Longueur : 128
8	imdb_id	identifiant du film sur le site imdb	numérique	Longueur max : 7
9	rating_count	nombre d'avis sur le film	numérique	Longueur : 30
10	rating_content	indication sur le public concerné par le film	choix	Longueur : 128
11	year	date de sortie du film	date	Longueur : 30
12	desc_en	description du film en anglais	texte	Longueur : 256
13	desc_fr	description du film en français	texte	Longueur : 256
14	anecdote	anecdote de tournage	texte	Longueur : 256
15	genre	genres du film	texte	Longueur : 128
16	img_url	url de l'image du poster du film	texte	Longueur : 128

## 2. Modèle Conceptuel

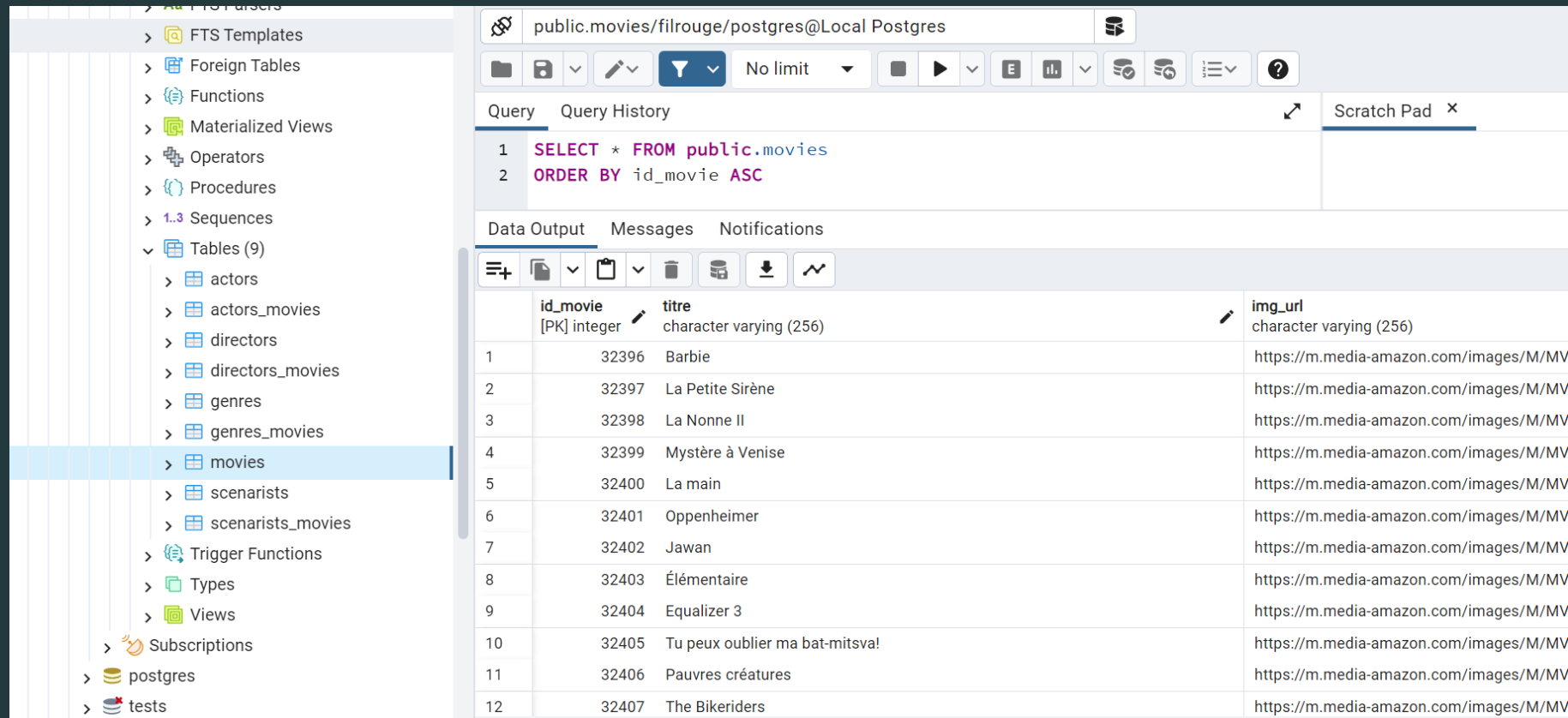


# 3. Schéma relationnel 3NF





# 4. Implémentation



The screenshot displays a PostgreSQL client interface. On the left, a sidebar shows a database schema with tables like `actors`, `directors`, `genres`, `movies`, `scenarists`, and `tests`. The `movies` table is selected. The main panel shows a query executed in the `public.movies/filrouge/postgres@Local Postgres` database:

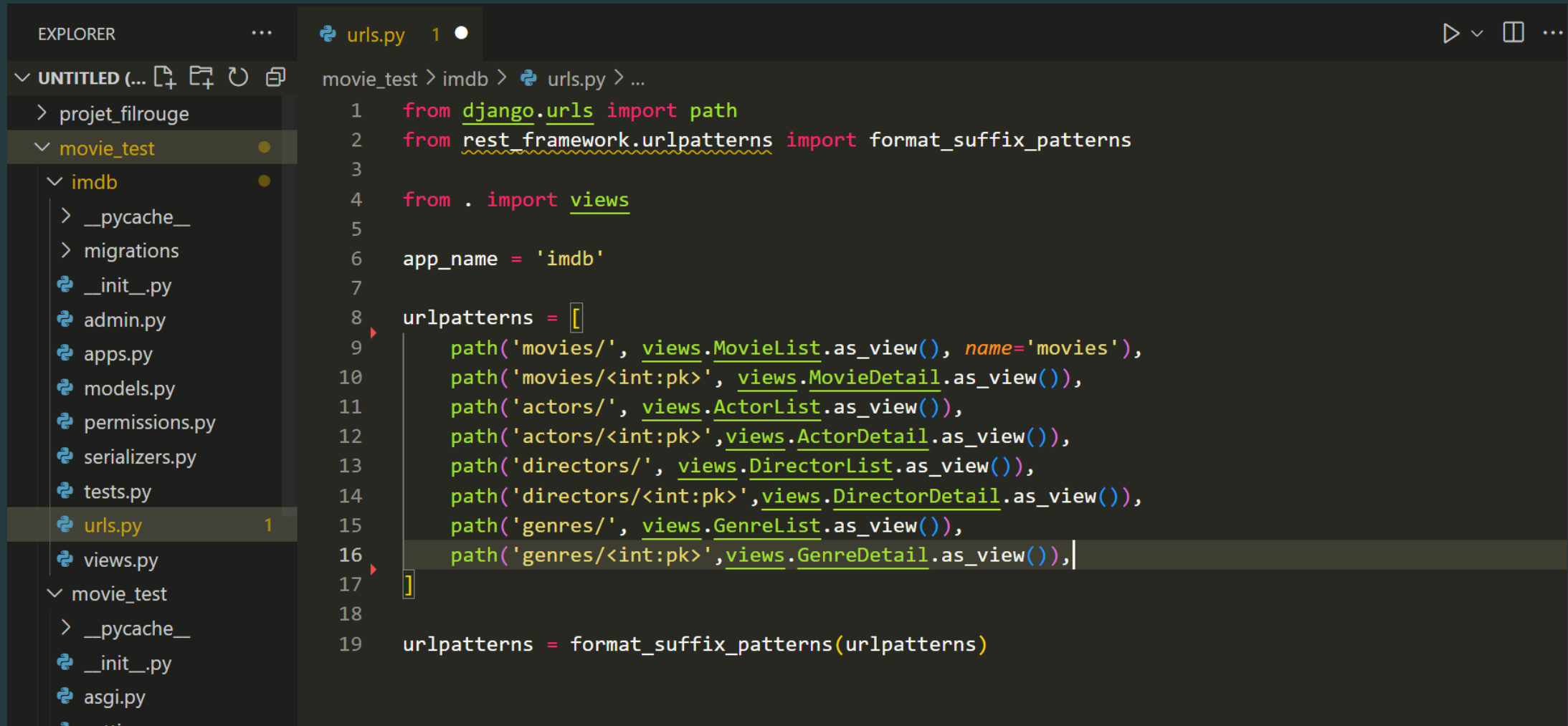
```
1 SELECT * FROM public.movies
2 ORDER BY id_movie ASC
```

The query results are displayed in a table with the following columns: `id_movie` (integer, primary key), `titre` (character varying (256)), and `img_url` (character varying (256)). The results list 12 movies, ordered by `id_movie` ascending.

	id_movie [PK] integer	titre character varying (256)	img_url character varying (256)
1	32396	Barbie	<a href="https://m.media-amazon.com/images/M/MV...">https://m.media-amazon.com/images/M/MV...</a>
2	32397	La Petite Sirène	<a href="https://m.media-amazon.com/images/M/MV...">https://m.media-amazon.com/images/M/MV...</a>
3	32398	La Nonne II	<a href="https://m.media-amazon.com/images/M/MV...">https://m.media-amazon.com/images/M/MV...</a>
4	32399	Mystère à Venise	<a href="https://m.media-amazon.com/images/M/MV...">https://m.media-amazon.com/images/M/MV...</a>
5	32400	La main	<a href="https://m.media-amazon.com/images/M/MV...">https://m.media-amazon.com/images/M/MV...</a>
6	32401	Oppenheimer	<a href="https://m.media-amazon.com/images/M/MV...">https://m.media-amazon.com/images/M/MV...</a>
7	32402	Jawan	<a href="https://m.media-amazon.com/images/M/MV...">https://m.media-amazon.com/images/M/MV...</a>
8	32403	Élémentaire	<a href="https://m.media-amazon.com/images/M/MV...">https://m.media-amazon.com/images/M/MV...</a>
9	32404	Equalizer 3	<a href="https://m.media-amazon.com/images/M/MV...">https://m.media-amazon.com/images/M/MV...</a>
10	32405	Tu peux oublier ma bat-mitsva!	<a href="https://m.media-amazon.com/images/M/MV...">https://m.media-amazon.com/images/M/MV...</a>
11	32406	Pauvres créatures	<a href="https://m.media-amazon.com/images/M/MV...">https://m.media-amazon.com/images/M/MV...</a>
12	32407	The Bikeriders	<a href="https://m.media-amazon.com/images/M/MV...">https://m.media-amazon.com/images/M/MV...</a>

# 5. Réalisation de l'API

- URLs dispos:



The screenshot shows a code editor with two panels. The left panel, titled 'EXPLORER', displays the project structure: 'UNTITLED (...)' containing 'projet\_filrouge', which contains 'movie\_test'. Inside 'movie\_test', there is an 'imdb' subdirectory. The 'imdb' directory contains files: '\_\_pycache\_\_', 'migrations', '\_\_init\_\_.py', 'admin.py', 'apps.py', 'models.py', 'permissions.py', 'serializers.py', 'tests.py', 'urls.py' (highlighted with a '1' icon), and 'views.py'. Below 'imdb' is another 'movie\_test' directory containing '\_\_pycache\_\_', '\_\_init\_\_.py', and 'asgi.py'. The right panel shows the content of 'urls.py' in the 'imdb' directory. The code defines the application name as 'imdb' and lists several URL patterns for movies, actors, directors, and genres, each mapped to a corresponding view function. The patterns are enclosed in a list, and the list is passed to 'format\_suffix\_patterns' at the end of the file.

```
movie_test > imdb > urls.py > ...
1  from django.urls import path
2  from rest_framework.urlpatterns import format_suffix_patterns
3
4  from . import views
5
6  app_name = 'imdb'
7
8  urlpatterns = [
9      path('movies/', views.MovieList.as_view(), name='movies'),
10     path('movies/<int:pk>', views.MovieDetail.as_view()),
11     path('actors/', views.ActorList.as_view()),
12     path('actors/<int:pk>', views.ActorDetail.as_view()),
13     path('directors/', views.DirectorList.as_view()),
14     path('directors/<int:pk>', views.DirectorDetail.as_view()),
15     path('genres/', views.GenreList.as_view()),
16     path('genres/<int:pk>', views.GenreDetail.as_view()),
17 ]
18
19 urlpatterns = format_suffix_patterns(urlpatterns)
```

# • Données sérialisées :

GET Get One by Id

GET Get data

+

...

No Environment

RESTful API basics: CRUD, test & variable / movies / Get One by Id

Save

GET

{{base\_url}}imdb/movies/34124

Send

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Cookies

Body

Cookies

Headers (10)

Test Results (1/1)

Status: 200 OK

Time: 1632 ms

Size: 1.13 KB

Save as example

...

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "id_movie": 34124,
3   "titre": "Margin Call",
4   "img_url": "https://m.media-amazon.com/images/M/MV5BOGI4YjdlyWMtNDUzMj00MWNhLTljNTAtZGEwOTQ3NDBiZjZmXkEyXkFqcGdeQXVyODIyOTYyMzY@._V1_.jpg",
5   "desc_fr": "Ce film suit les intervenants clés d'une banque d'investissement pendant 24 heures au tout début de la crise financière de 2008.",
6   "desc_en": "This film follows key players at an investment bank over 24 hours at the very beginning of the 2008 financial crisis.",
7   "anecdote": "The film was shot in 17 days.",
8   "rating_count": 139862,
9   "rating_content": "Tous publics",
10  "year": "2012-05-02",
11  "imdb_int": 1615147,
12  "actors": [
13    {
14      "id": 1016,
15      "name": "Zachary Quinto"
16    },
17    {
18      "id": 1971,
19      "name": "Stanley Tucci"
```

8



# 6. Requêtes et résultats

## 6.1 Top 10 des films les plus populaires

Requete :

Query	Query History
1	<b>SELECT</b> titre
2	<b>FROM</b> movies
3	<b>ORDER BY</b> rating_count <b>DESC</b>
4	<b>LIMIT</b> 10 ;

Output :

	titre character varying (256)	
1	Les évadés	
2	The Dark Knight : Le Chevalier noir	
3	Inception	
4	Fight Club	
5	Forrest Gump	
6	Matrix	
7	Interstellar	
8	Le Parrain	
9	Le Seigneur des anneaux : Le Retour du roi	
10	The Dark Knight Rises	

## 6.2 Flop 10 des films les moins populaires

Requete :

Query	Query History
1	<b>SELECT</b> titre
2	<b>FROM</b> movies
3	<b>ORDER BY</b> rating_count <b>ASC</b>
4	<b>LIMIT</b> 10 ;

Output :



	titre character varying (256) 🔒
1	Widow Clicquot
2	Irena's Vow
3	Wicked: Deuxième Partie
4	Talk to Me, Sweet Darling
5	Mascarpone
6	The Kill Room
7	Sables mortels
8	Tuesday
9	Sapio
10	Us or Them

## 6.3 Directeur ayant fait le plus de films

Requete :

```
Query  Query History
1  SELECT firstname, lastname
2  FROM directors dir
3  INNER JOIN (
4  SELECT dm.director_id, COUNT(dm.director_id) as count_movies
5  FROM directors_movies dm
6  GROUP BY dm.director_id
7  ORDER BY count_movies DESC
8  LIMIT 1) dm0
9  ON dir.id = dm0.director_id;
```

Output :

	firstname character varying 	lastname character varying 
1	Steven	Spielberg





## 6.4 Acteur qui a joué dans le plus de films

Requete :

```
Query  Query History
1  SELECT act.firstname, act.lastname
2  FROM actors act
3  INNER JOIN (
4      SELECT am.actor_id, COUNT(am.actor_id) as count_movies
5      FROM actors_movies am
6      GROUP BY am.actor_id
7      ORDER BY count_movies DESC
8      LIMIT 1) am0
9  ON act.id = am0.actor_id;
```

Output :


	firstname character varying 	lastname character varying 
1	Bruce	Willis

## 6.5 Nombre de films avec "love" dans la description

Requete :

Output :

Query	Query History
1	<code>SELECT COUNT(mv.id_movie) as count_love</code>
2	<code>FROM movies mv</code>
3	<code>WHERE mv.desc_en LIKE '%love%';</code>

	count_love bigint 
1	609







## 6.6 Le réalisateur ayant au moins 10 films ( présents dans la db) le plus populaire en moyenne sur ceux ci

Requete :

```
Query  Query History
1  SELECT
2      dir.firstname,
3      dir.lastname,
4      AVG(mv.rating_count) as average_popularity,
5      COUNT(mv.id_movie) as count_movie
6  FROM directors dir
7  INNER JOIN directors_movies dm
8  ON dir.id = dm.director_id
9  INNER JOIN movies mv
10 ON dm.movies_id = mv.id_movie
11 GROUP BY dir.id
12 ORDER BY count_movie DESC, average_popularity DESC
13 LIMIT 1;
```

Output :

	firstname character varying 	lastname character varying 	average_popularity numeric 	count_movie bigint 
1	Steven	Soderbergh	93372.615384615385	13