Endava Data Engineer Project

1. Alegerea unui dataset care permite realizarea de rapoarte

Am ales ca dataset primar informații despre filmele de pe platforma MUBI, o platforma asemanatoare cu Netflixul: "Netflix for art movies".

Datele au fost preluate de pe Kaggle:

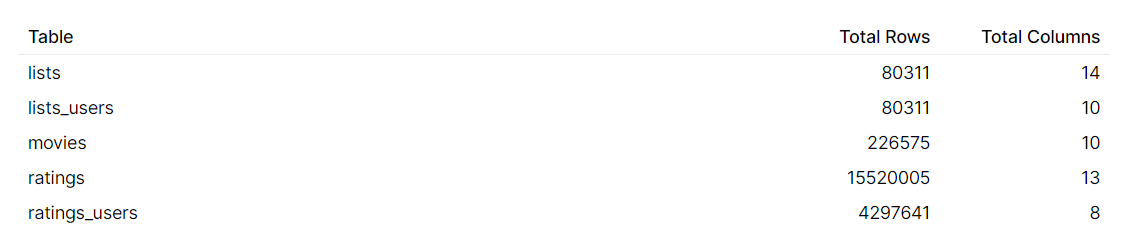
<https://www.kaggle.com/clementmsika/mubi-sqlite-database-for-movie-lovers>

Acestea au fost colectate de către utilizatorul [Clément Msika prin intermediul api-ului furnizat de către MUBI.](https://www.kaggle.com/clementmsika)

Datele au fost furnizate in 2 formate posibile:

* Baza de date sqlite
* Sub forma de fisiere csv

Continutul dataset-ului:



* “movies” reprezinta tabela care contine date despre toate filmele inregistrate pe platforma Mubi
* “ratings” contine informatiile despre rating-urile date de utilizatori pentru diferite filme
* “ratings\_users” este tabela care contine de 3 ori mai putine informatii decat “ratings” si include doar ultimul rating dat pe o anumita zi.
* “lists” contine grupuri de filme organizate de utlizatori care nu si au setat profilul privat.
* “lists\_users” sunt listele create de utilizatorii care nu si-au setat contul in modul privat.

Probleme aduse de acest dataset:

1. Datele din tabele nu sunt normalizate

* Tabela “movies” contine campurile “director\_id”, “director\_name” si “director\_url” cu elemente separate prin virgula.
* Tabelele “lists\_users” si “ratings” conțin date despre useri care trebuie colectate într-un singur tabel “users”.

1. Datorita faptului ca exista coloane care nu contin date atomice, ci separate prin virgula, formatul fisierelor de tip csv nu este corect (coloanele sunt separate prin virgula, dar si datele neatomice la fel)

* Am decis sa optez pentru extragerea de date direct din baza de date Sqlite.

1. Numarul de inregistrari din tabela “ratings” este destul de mare.

* Reducerea înregistrărilor in mod determinist (ex: delete from ratings where rating\_id > 2000000) duce la alterarea informatiilor atat despre utilizatori (deoarece trebuie extrasi utilizatorii din toate inregistrarile din ratings)

1. Alegerea engine-ului potrivit pentru manipularea datelor.

Stocarea datelor raw in Sqlite nu este o alegere buna, deoarece:

* aplicatia nu este una standalone, de dimensiune mica
* necesita scalabilitate
* citirea si scrierea se face direct de pe disc
* pentru basic development si testing

Diferente intre Sqlite si PostgreSQL:

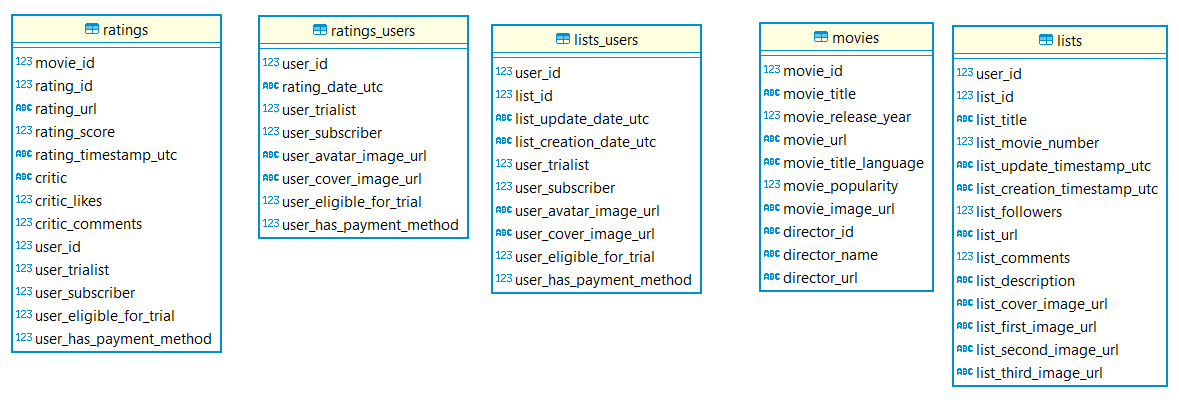
1. Sqlite este embedded, Serverless, pe cand PostgreSQL are nevoie de un server pe care sa ruleze.
2. Sqlite suporta doar 5 tipuri de date, pe cand PostgreSQL detine o multitudine de tipuri de date diferite.
3. Sqlite este o baza de date portabila (motiv pentru care am putut lua fisierul de pe internet cu usurinta), pe cand PostgreSQL necesita export.
4. **Sqlite nu suporta multiacces din partea a mai multor useri, pe cand PostgreSQl da.**
5. Sqlite suporta doar un subset de comenzi PL/SQL, daca utilizeaza StepSqlite product from Metatranz LLC, pe cand PostgreSQL suporta comenzi complexe de plpgsql.

Din toate aceste motive, am decis ca sa realizez o migrare din Sqlite in PostgreSQL utilizand DBeaver **(desi nu este ok d.p.d.v al productiei)**.

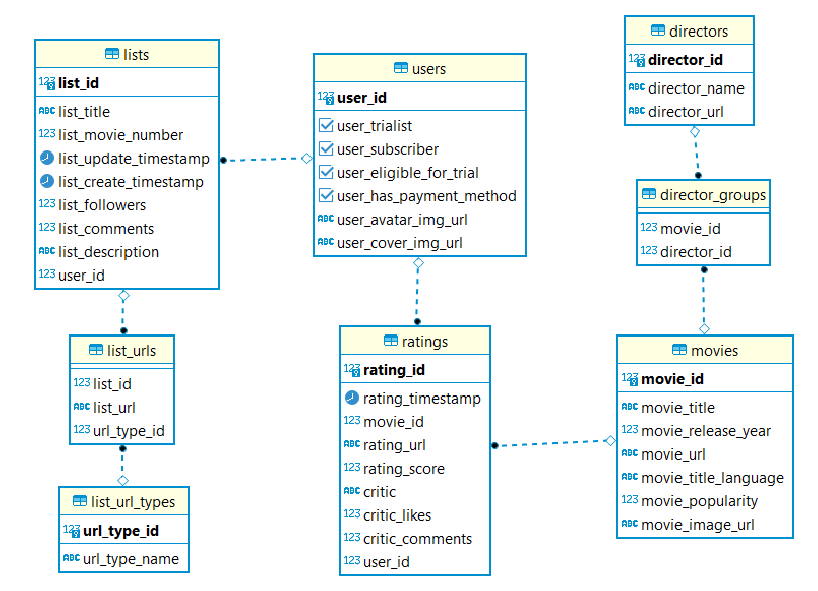
Un motiv in plus este acela ca Talend nu recunoaste schemele pentru Sqlite automat si mi-a fost mai usor sa mut intr-un PostgreSql si sa utilizez asa in Talend.

1. Realizarea structurii finale a tabelelor normalizate in PostgreSQL

Procesul de normalizare a tabelelor incepe cu crearea structurii de baza a tabelelor normalizate in postgres. Astfel, de la varianta cruda a datelor:



S-au creat tabelele normalizate:



Explicatie semnificatie tabele normalizate:

* “directors” contine date despre toti directorii de filme
* “director\_groups” contine relatia de multi la mai multi intre directori si filme
* “movies” contine date despre filmele de pe platforma MUBI
* “ratings” contine ratinguri date de catre useri pentru filme
* “users” sunt toti userii de pe website
* “lists” sunt listele de filme create si updatate de utilizatori
* “lists\_urls” sunt url-uri spre imagini sau continutul unei liste
* “list\_url\_types” sunt tipurile de url-uri care exista din lista:
  + url
  + cover\_url
  + first\_image\_url
  + second\_image\_url
  + third\_image\_url

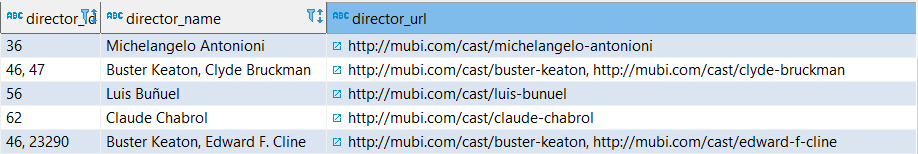
1. Alegerea unui tool potrivit de ETL

Din cauza faptului ca tabela “ratings” contine un volum mare de date (15000000 înregistrări), a fost necesar un tool rapid care accepta atat sa utilizeze date din surse cu PostgreSQL, cat și să fie eficient la operații.

În urma diferitelor încercări cu tool-uri precum Altova MapForce (nu acceptă Postgres), etc., am ales sa utilizez Talend pentru realizarea ETL-ului.

1. Preprocesare date înainte de a utiliza Talend

In tabela “movies”, daca un film are mai multi directori, acestia vor fi regasiti in acelasi camp (nu respecta forma normala 1 separati prin virgula.



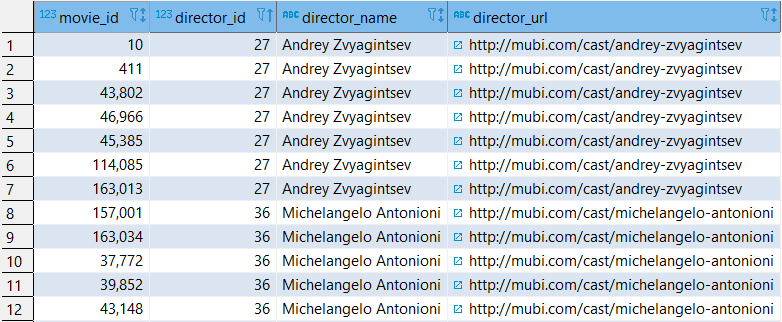
Desi problema poate fi rezolvata direct in Talend, din motive educationale, am ales sa utilizez plpgsql (feature care nu este suportat de sqlite) pentru a creea un materialized view cu campurile deja normalizate.

Totusi, materialized view am considerat ca nu se potriveste cu ceea ce am nevoie deoarece:

* Trebuie dat reload de fiecare data cand se porneste job-ul de migrare
* Talend nu recunoaste materialized views.

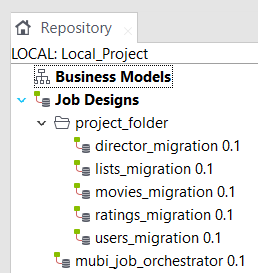
Solutie:

* Utilizez in proiect un view simplu în care salvez “movie\_id”, “director\_id”,”director\_name” si “director\_url” pentru a le transforma mai ușor în ETL:



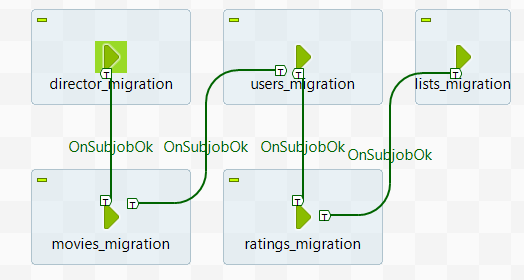
1. Job-uri Talend

Este importanta ordinea în care sunt transformate informațiile, deoarece acestea sunt influențate de constrângerile referențiale. De exemplu, dacă se trimit date in tabela “director\_groups”, exista șansa sa existe un director\_id care nu are corespondent in tabela “directors”. Acest lucru implica ca mai intai sa fie populata tabela “directors”.



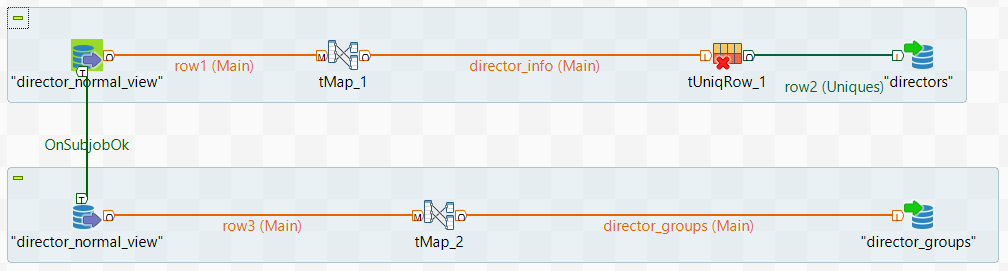
a) mubi\_job\_orchestrator

* un job care coordonează ordinea în care sa se efectueze restul subjob-urilor, astfel incat sa nu fie probleme legate de constrangeri referențiale



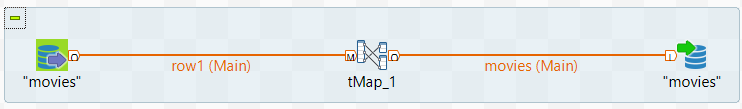
b) director\_migration

* utilizeaza “director\_normal\_view”
* view-ul incapsuleaza un sql care normalizeaza tabela “movies”
* extrage informațiile despre directori și le salvează în tabelele “directors” si “director\_groups”



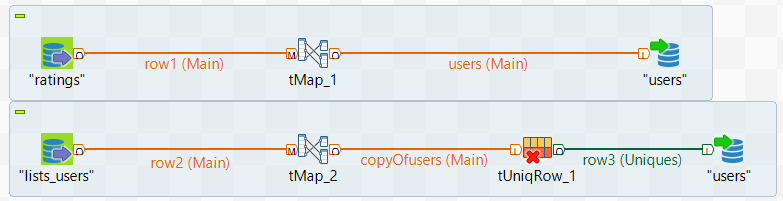
c) movie\_migration

* preia toate filmele și le stochează în tabela normalizată



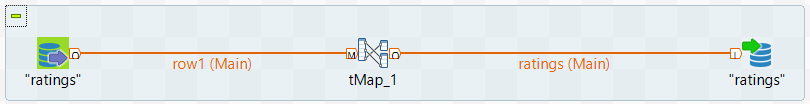
d) users\_migration

* extrage date din tabelele “ratings” și “list\_user” și le salvează în tabela users



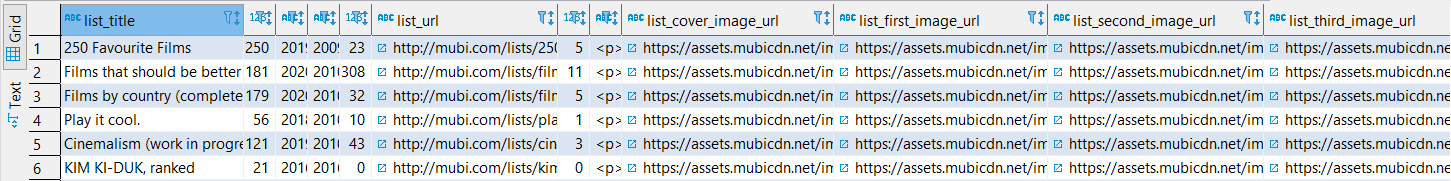
e) ratings\_migration

* deoarece reprezinta tabela cea mai populata, necesita cele mai multe resurse.
* fara o metoda de fetch pe segmente, programul va da eroare din cauza memoriei.

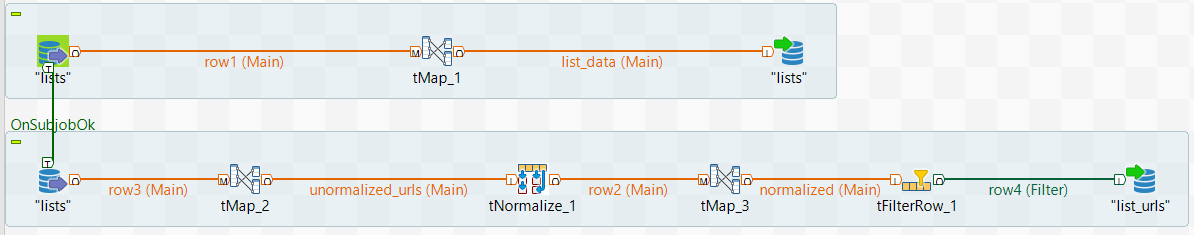


f) list\_migration

* aici problema care a intervenit a fost transformarea din coloane in randuri:
  + list\_url
  + list\_cover\_image\_url
  + list\_first\_image\_url
  + list\_second\_image\_url
  + list\_third\_image\_url



* s-a creat tabela “list\_url\_types” care reține tipul unui rand (numele coloanei) și id-ul tipului, astfel incat pot fi adaugate oricate alte imagini ce pot fi asociate unei liste



* s-a procedat în felul următor:

1. se transpun toate coloanele în una singura care contine url-urile concatenate în felul următor (realizat de tMap\_2):

<url>;1,<cover>;2,<first>;3,<second>;4,<third>;5

1. se normalizeaza după delimitatorul ","
2. “tMap\_3” va împărți fiecare url de tipul <url>;<tip\_id> în cele 2 componente
3. rezultatul va fi salvat în baza de date
4. Alegerea unui tool de BI potrivit

Am ales Tableau ca tool pentru BI reporting, deoarece, spre deosebire de alte tool-uri de BI, oferă posibilitatea de a crea tabele din comenzi SQL, conferind in același timp o performanță sporită la realizarea diferitelor task-uri.

Necesitatea performanței provine de la tabela “ratings” care contine un volum foarte mare de date din motive de a nu altera rezultatele concrete prin selecția unui subset de înregistrări. De exemplu, PowerBi necesita foarte mult timp pentru a realiza un merge (join) între tabele.

Tableau este foarte interactiv, avand la dispoziție un UI friendly. De asemenea, poate prelua date dintr-o gama foarte mare de surse (engine-uri pentru baze de date diferite, Cloud, fișiere locale, etc.).

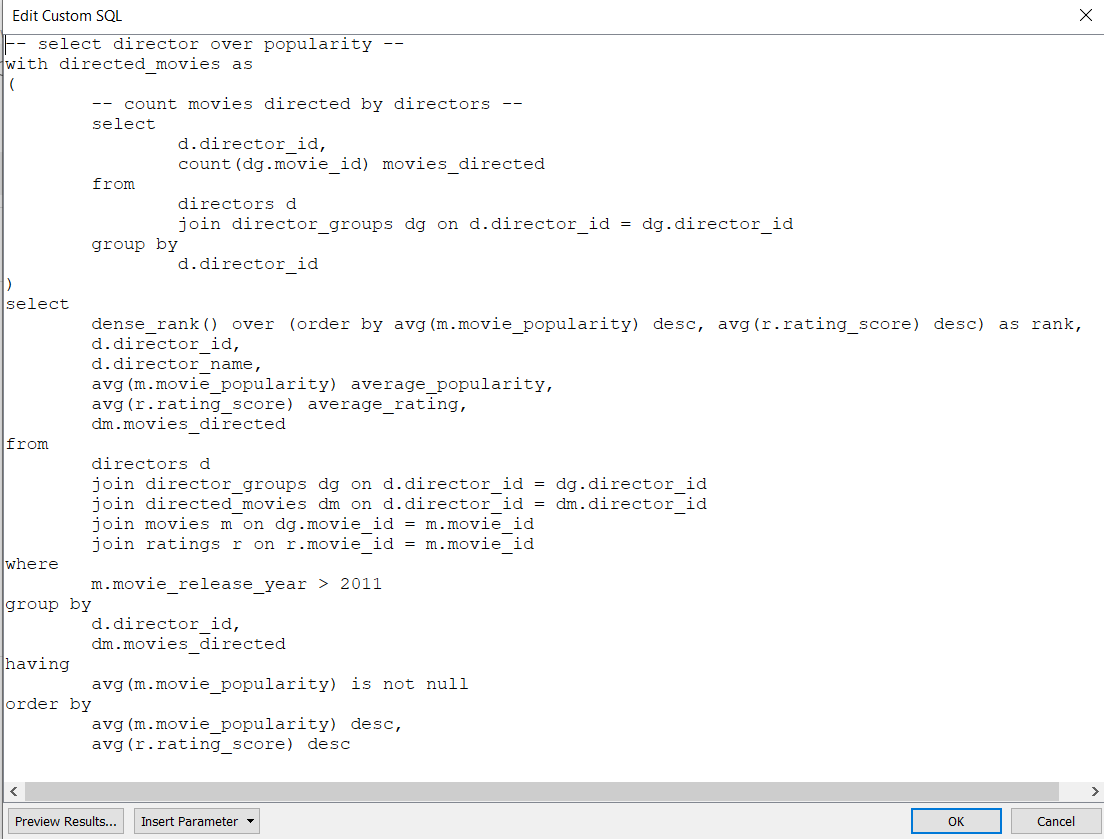
1. Premergator realizarii rapoartelor

Datele normalizate preluate din baza de date PostgreSQL au fost completate cu un Dataset ce poate fi gasit la link-ul: <https://zenodo.org/record/1240586#.YSS1Xo4zZPZ>

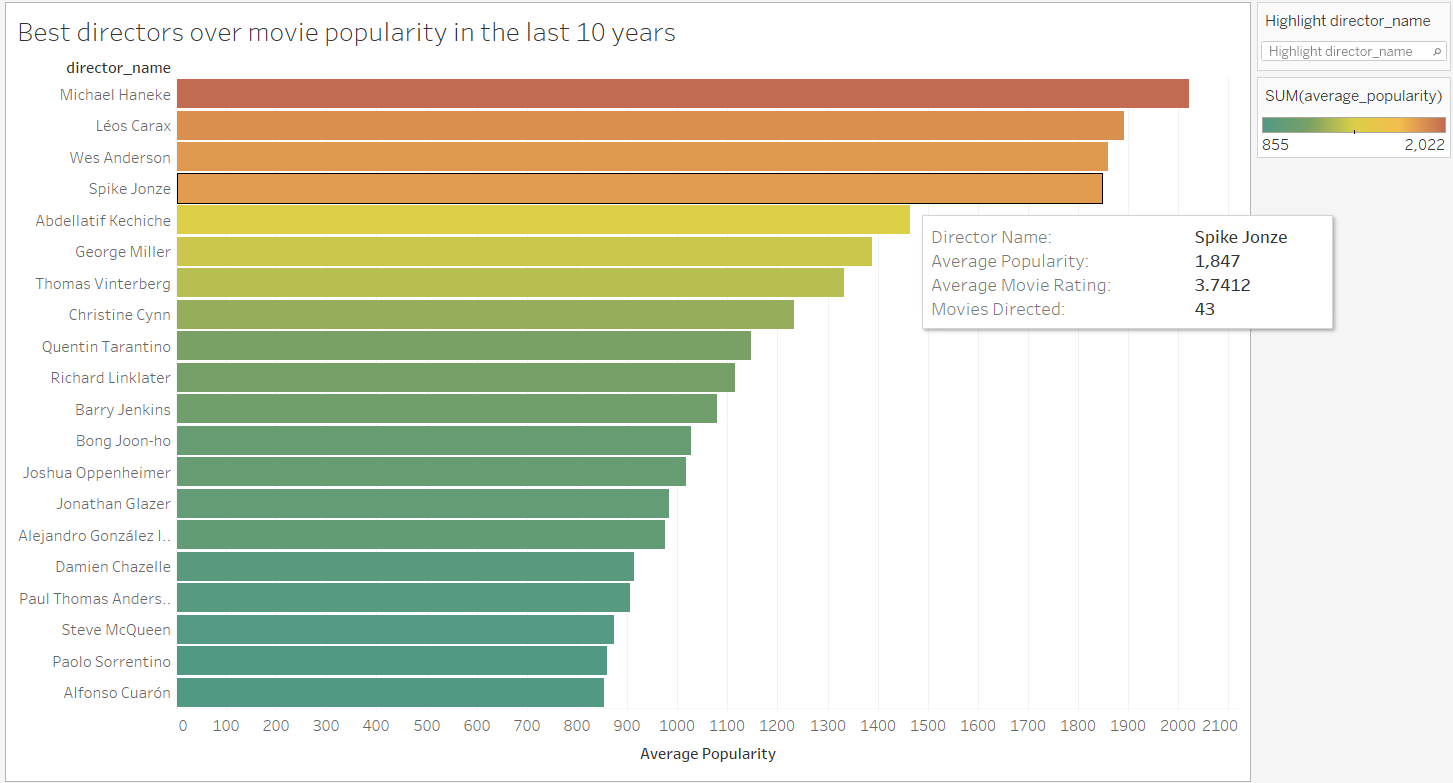
Acesta contine completari la filme legate de budget și revenue pentru fiecare în parte. Legatura dintre aceste date si tabelul “movies” a fost făcută prin intermediul titlului filmului.

1. Rapoarte realizate în Tableau
2. Best movie directors over movies popularity in the last 10 years

Pentru acest raport, am utilizat functionalitatea tool-ului de a genera tabele din cod SQL.

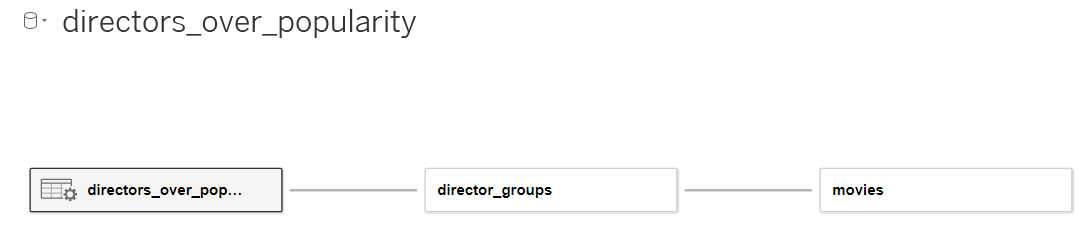


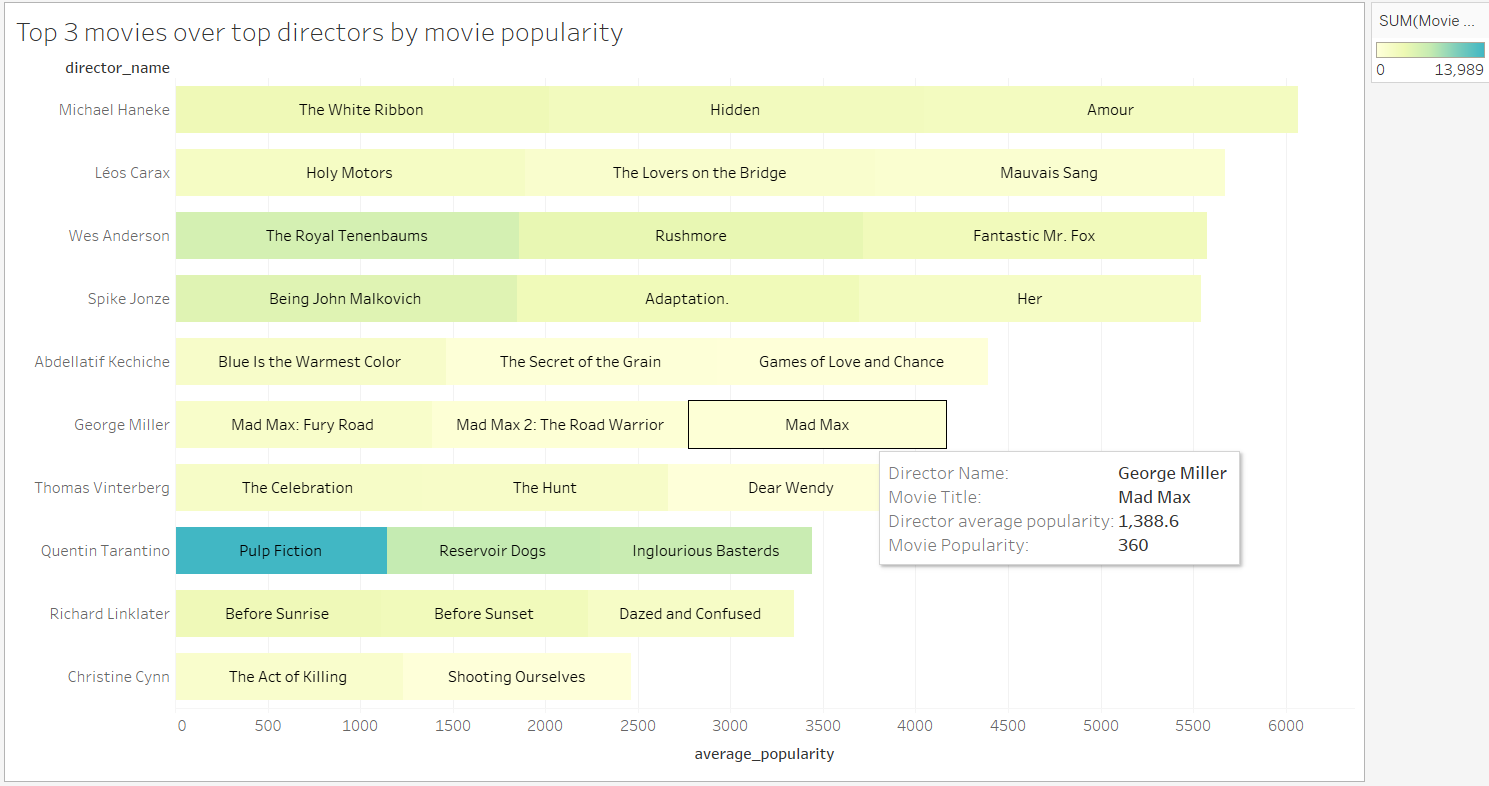
Utilizand campul rank, am pus un filtru ca sa afiseze doar pe primii 20 de directori (considerand doar filmele existente pe platforma MUBI).



1. Top 3 movies over popularity directed by this best directors

Am utilizat view-ul creat anterior si am facut join cu tabela movies pentru a identifica top 3 filme ale fiecarui director (create in toti anii).

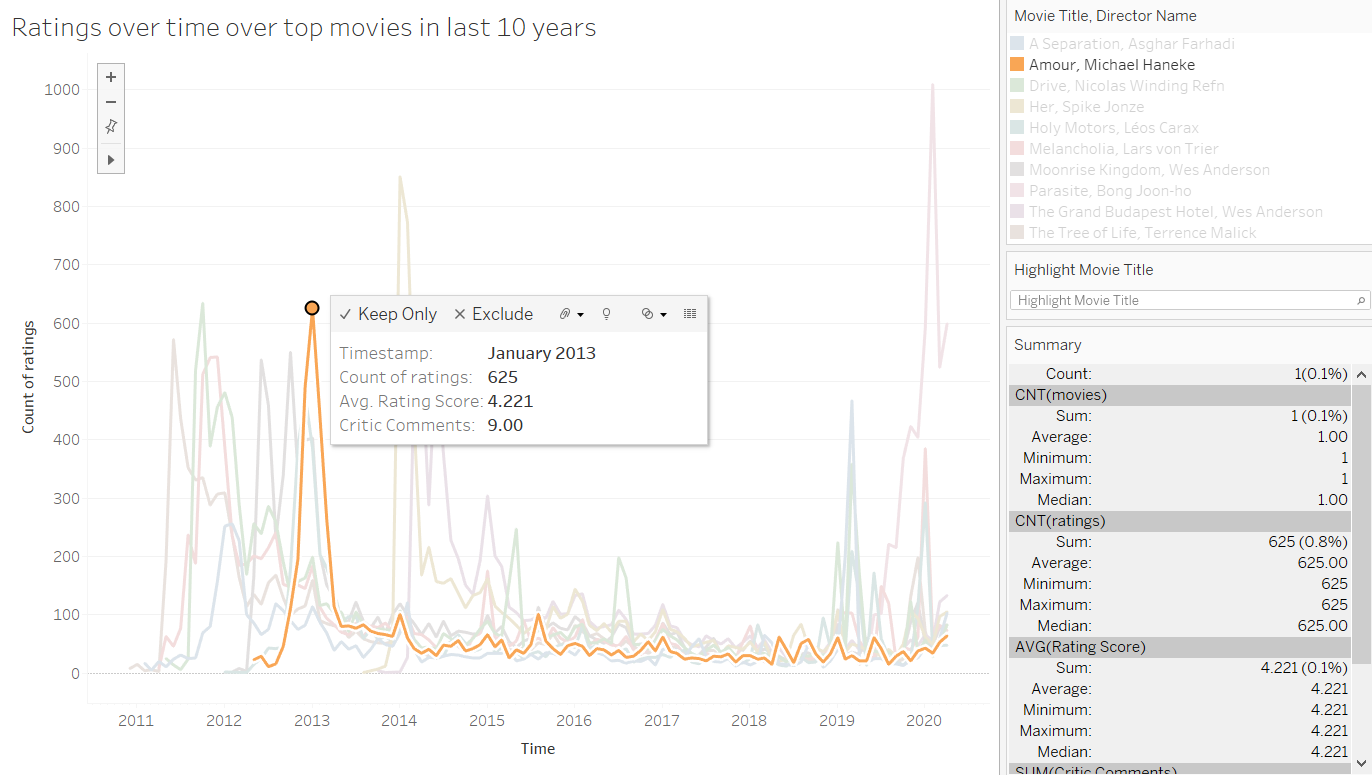


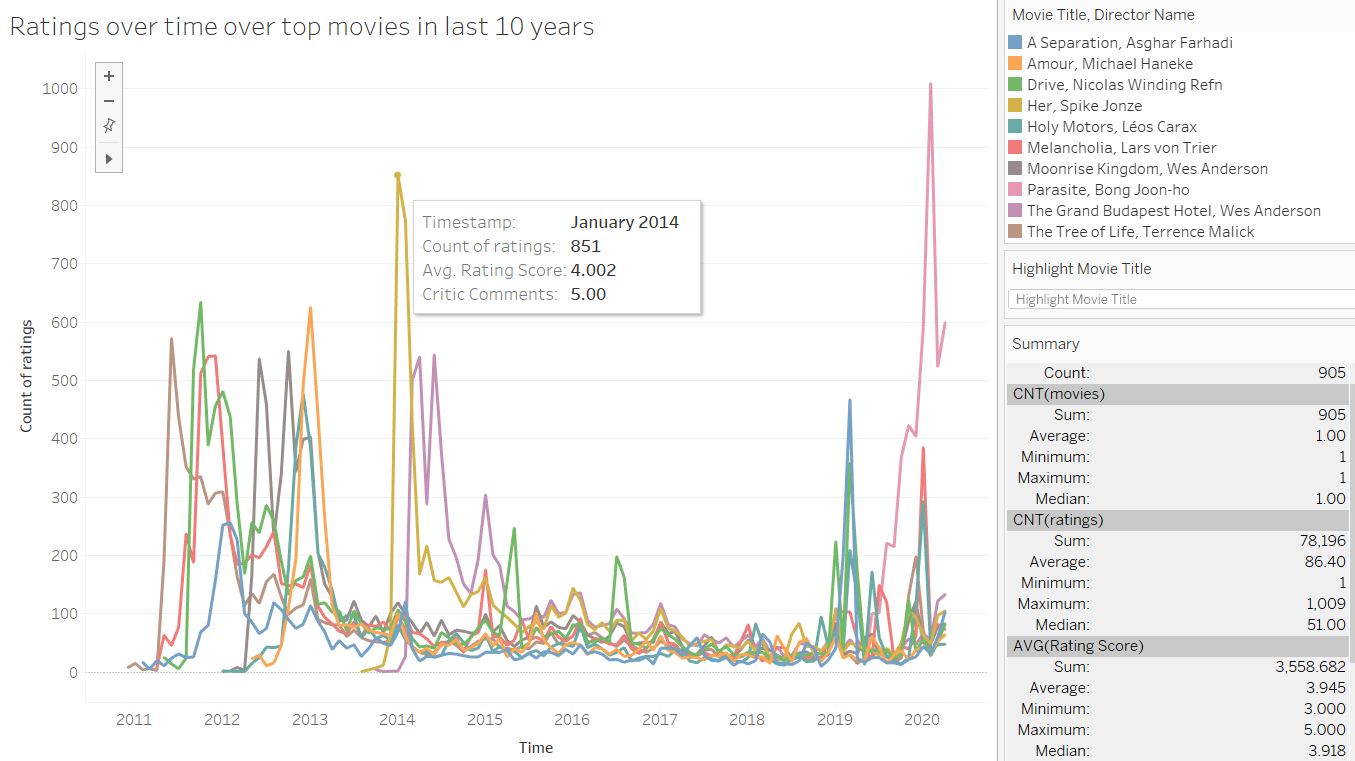


1. Ratings over time over top movies in the last 10 years

Acest raport permite vizualizarea evolutiei ratingurilor primite de-a lungul timpului pentru filmele de top din ultimii 10 ani.

De exemplu, se poate observa ca “Amour” regizat de Michael Haneke a obținut multe ratinguri din partea utilizatorilor în ianuarie 2013, iar după a scăzut la un nivel aproape constant.

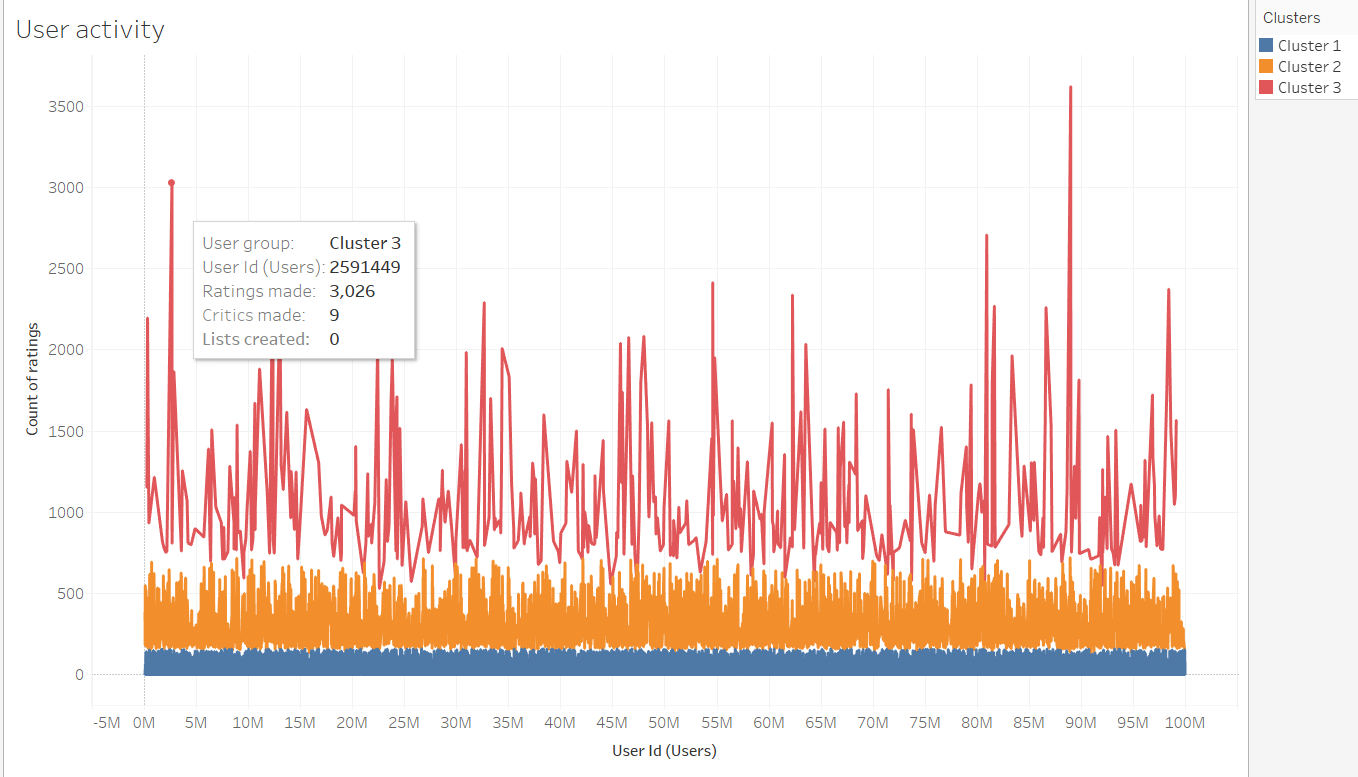




1. User activity

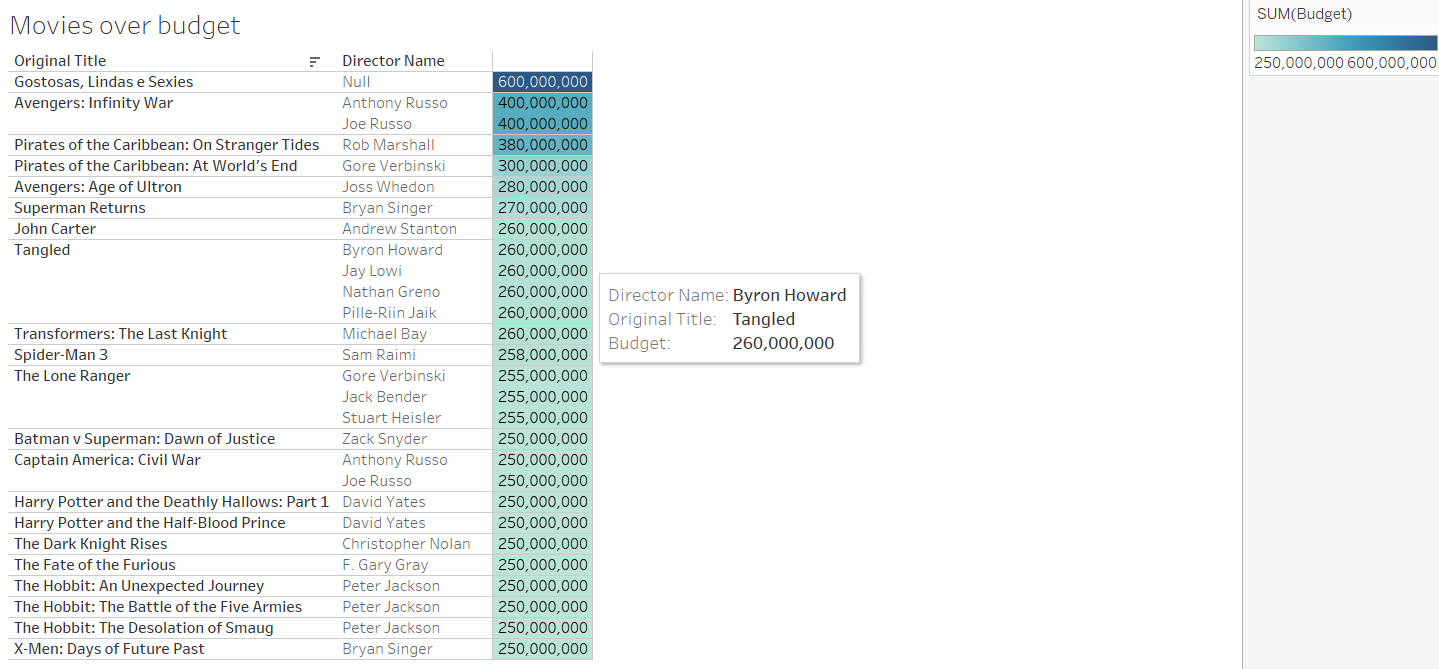
In acest raport se diferentiaza 3 grupuri diferite de useri in functie de activitatea acestora pe site-ul MUBI (numarul de recenzii date si numarul de critici date).

Pentru realizarea acesteia, s-au utilizat toate tabelele din baza de date normalizata legate prin join.

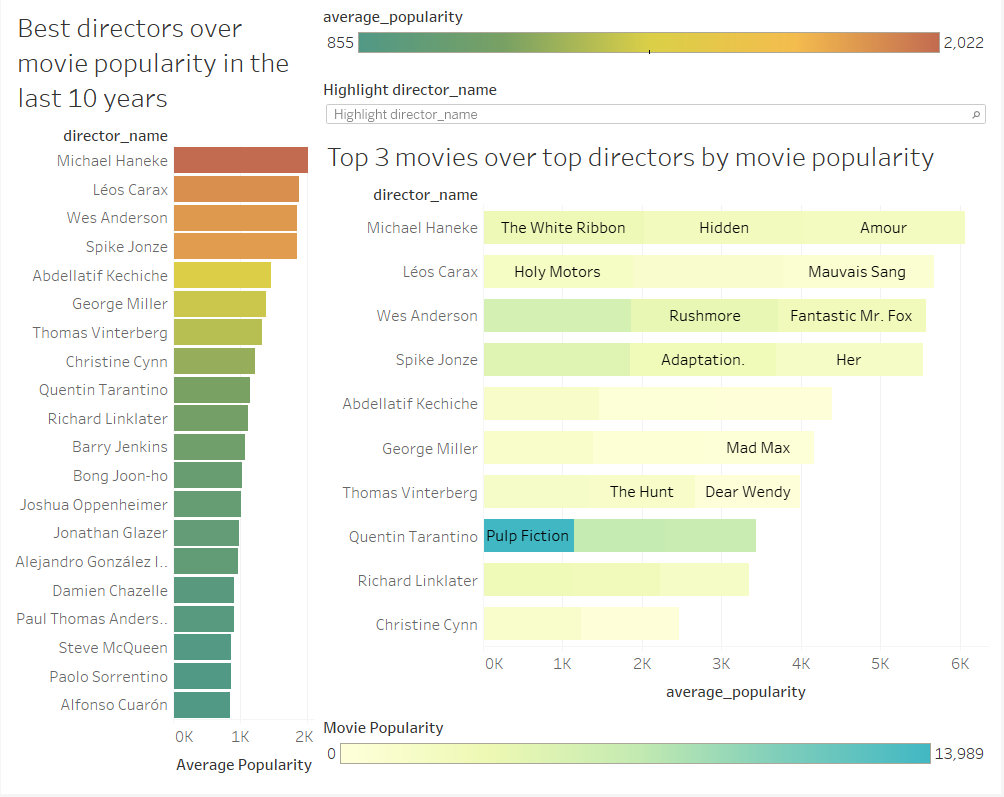


1. Movies over budget

Alipind datasets cu date despre bugetul filmelor de tabelele din baza de date normalizata, se obtine un top al filmelor după cel mai mare buget cheltuit.



1. Dashboard Best Directors



1. Dashboard User Activity

