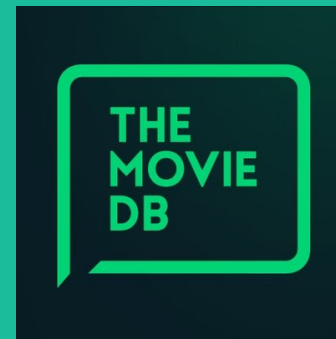


Construction d'une base de données de films et analyses



Gérald Bouget

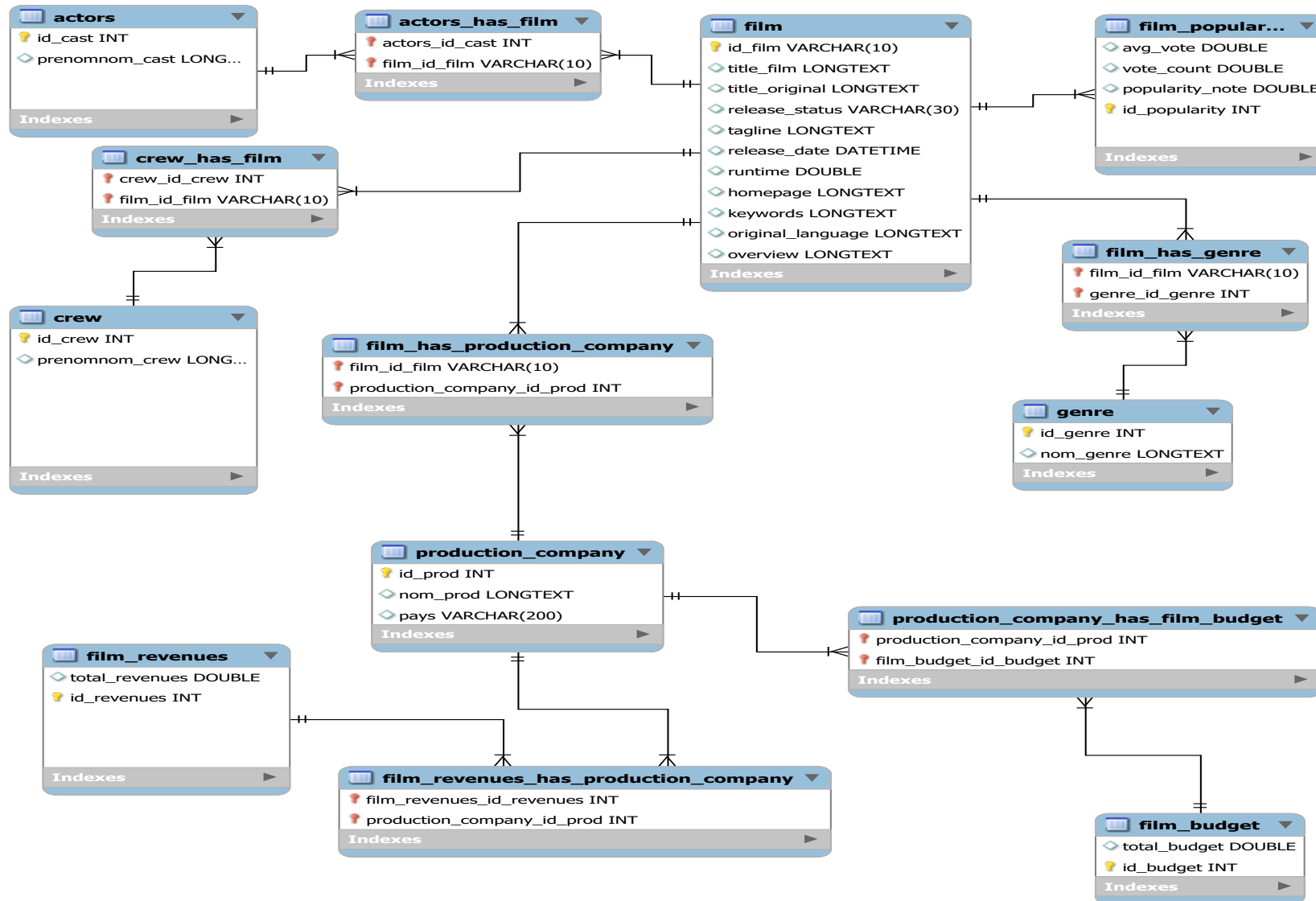
Le 18 décembre 2019

Simplon – Formation Développeur Data - Elancourt

Workflow

- 1) Aperçu et prise de connaissance des sources de données dans Excel (format, taille, noms des variables d'origine, type de contenu)
- 2) Conceptualisation de la base de données sur papier puis sous forme de diagramme UML et création des relations entre les tables.
- 3) Forward engineering : création des tables dans Mysql à partir du diagramme UML
- 4) À partir du fichier excel : récupération des colonnes par sous ensembles exportés chacun sous un fichier .csv.
- 5) « Nettoyage » rapide de chacun de ces fichiers en les important sous notebook – utilisation de la librairie Pandas et export en .csv
- 6) Dans mysql Workbench : import de chacun des fichiers dans les tables correspondantes
- 7) Import des tables dans notebook via Pymsql
- 8) Analyse des données avec la librairie Pandas et visualisation avec Matplotlib

Conceptualisation de la base de données



Conceptualisation de la base de données

- **Principes d'élaboration :**

pour créer cette base j'ai essayé de respecter les règles fondamentales de l'art de la modélisation :

- Pas de valeurs nulles
- Des données atomiques
- Pas de redondance
- La modification d'une donnée ne doit pas impacter plus d'un ligne

- **Ceci explique qu'il y ait de nombreuses tables**

- **Présence de relations « *many to many* » qui engendre la création de tables intermédiaires**

Conceptualisation de la base de données

- Exemple : la table 'film' a une relation « *many to many* » avec la table 'acteurs' :
 - Un acteur peut jouer dans plusieurs films et un film peut avoir plusieurs acteurs.
 - La table intermédiaire 'actors_has_films' permet de relier les deux tables par leur 'id' respectifs.

Conceptualisation de la base de données

- Création de fichiers .csv séparés pour chaque table :

exemple :

‘table ‘revenus’

	A	B
1	<u>movie_id</u>	film_revenues
2	19995	2787965087
3	285	961000000
4	206647	880674609
5	49026	1084939099
6	49529	284139100
7	559	890871626
8	38757	591794936
9	99861	1405403694
10	767	933959197
11	209112	873260194
12	1452	391081192
13	10764	586090727
14	58	1065659812
15	57201	80280010

Conceptualisation de la base de données

- Import des fichiers et nettoyage rapide des tables :

exemple : changement d'affichage de certaines colonnes, corrections d'incohérences flagrantes.

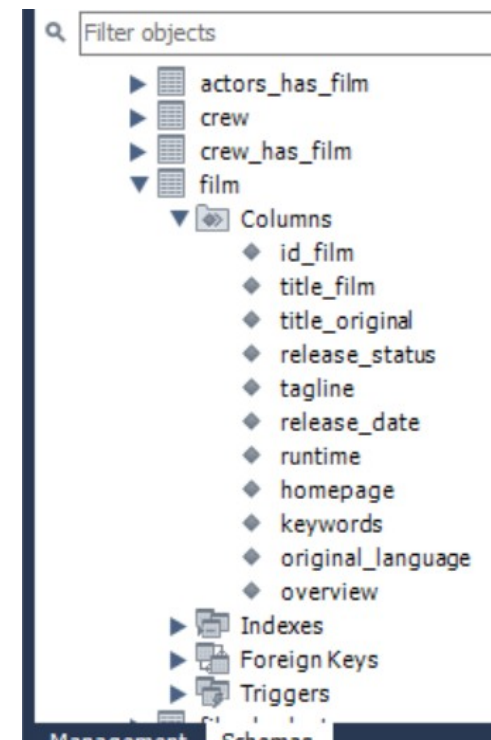
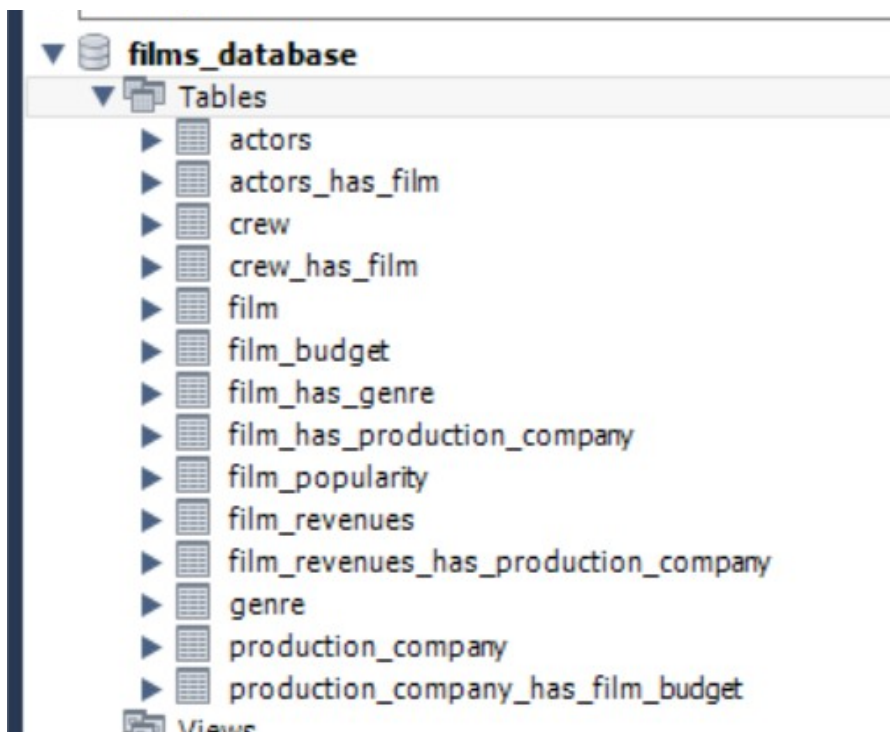
```
Entrée [884]: ### nan values dans runtime or nan = float >> pas possible de passer en int. Il faut passer par la fonction  
tb_film.runtime = pd.array(tb_film.runtime, dtype="Int64")
```

```
Entrée [885]: tb_film.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 4803 entries, 0 to 4802  
Data columns (total 11 columns):  
id_film          4803 non-null int64  
title           4803 non-null object  
title_original  4803 non-null object  
released_status 4803 non-null object  
tagline         3959 non-null object  
release_date    4802 non-null object  
runtime         4801 non-null Int64  
keywords        4389 non-null object  
original_language 4802 non-null object  
overview        4800 non-null object  
homepage        1715 non-null object  
dtypes: Int64(1), int64(1), object(9)  
memory usage: 417.6+ KB
```

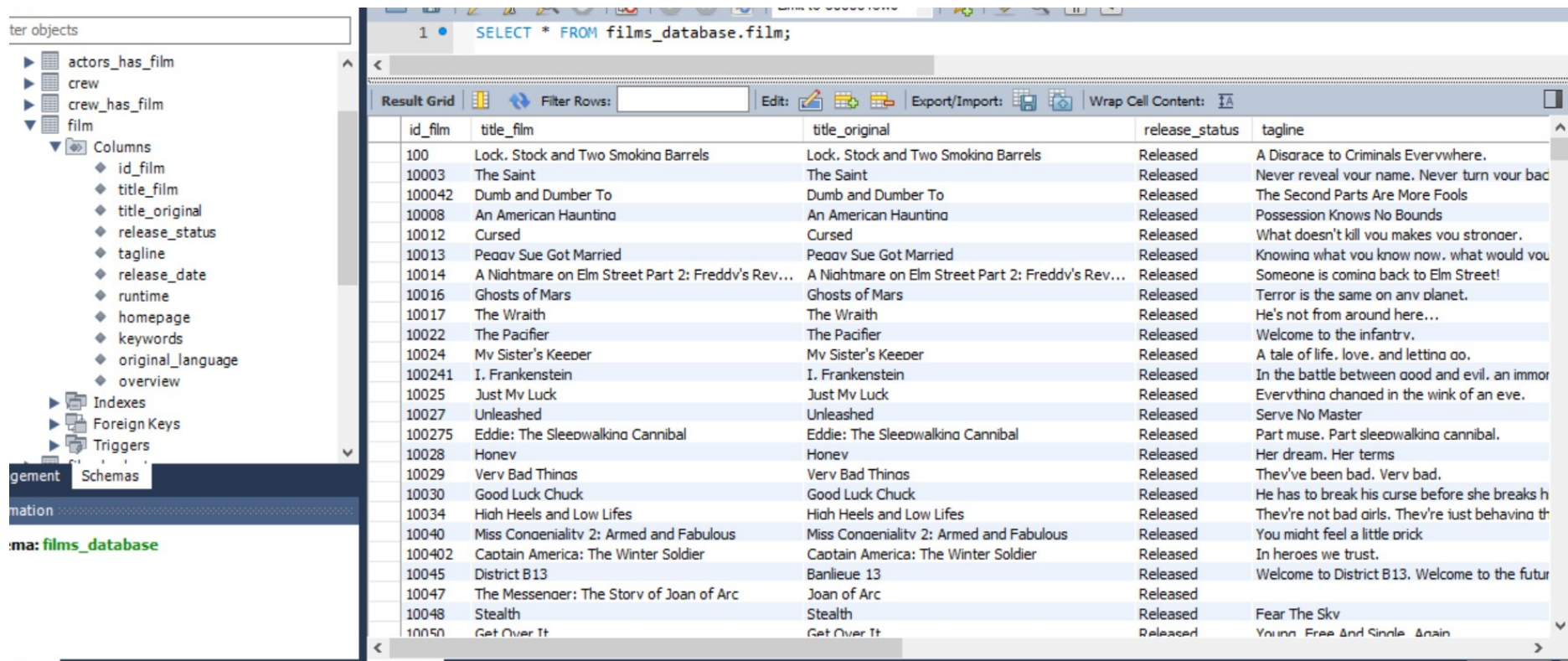
Conceptualisation de la base de données

- Création des tables dans mysql vial forward engineering et verification :



Conceptualisation de la base de données

- Import des fichiers .csv dans chaque table via « wizard import »



The screenshot displays a database management interface. On the left, a tree view shows the 'films_database' schema with tables like 'actors_has_film', 'crew', 'crew_has_film', and 'film'. The 'film' table is selected, showing its columns: id_film, title_film, title_original, release_status, and tagline. The main window shows a SQL query: `SELECT * FROM films_database.film;` and a result grid with 20 rows of film data.

id_film	title_film	title_original	release_status	tagline
100	Lock, Stock and Two Smoking Barrels	Lock, Stock and Two Smoking Barrels	Released	A Disgrace to Criminals Everywhere.
10003	The Saint	The Saint	Released	Never reveal your name. Never turn your back.
100042	Dumb and Dumber To	Dumb and Dumber To	Released	The Second Parts Are More Fools
10008	An American Haunting	An American Haunting	Released	Possession Knows No Bounds
10012	Cursed	Cursed	Released	What doesn't kill you makes you stronger.
10013	Peacov Sue Got Married	Peacov Sue Got Married	Released	Knowing what you know now, what would you do?
10014	A Nightmare on Elm Street Part 2: Freddy's Revenge	A Nightmare on Elm Street Part 2: Freddy's Revenge	Released	Someone is coming back to Elm Street!
10016	Ghosts of Mars	Ghosts of Mars	Released	Terror is the same on any planet.
10017	The Wraith	The Wraith	Released	He's not from around here...
10022	The Pacifier	The Pacifier	Released	Welcome to the infantry.
10024	Mv Sister's Keeper	Mv Sister's Keeper	Released	A tale of life, love, and letting go.
100241	I. Frankenstein	I. Frankenstein	Released	In the battle between good and evil, an immortal is born.
10025	Just Mv Luck	Just Mv Luck	Released	Everything changed in the wink of an eye.
10027	Unleashed	Unleashed	Released	Serve No Master
100275	Eddie: The Sleepwalking Cannibal	Eddie: The Sleepwalking Cannibal	Released	Part muse. Part sleepwalking cannibal.
10028	Honey	Honey	Released	Her dream. Her terms
10029	Verv Bad Things	Verv Bad Things	Released	They've been bad. Verv bad.
10030	Good Luck Chuck	Good Luck Chuck	Released	He has to break his curse before she breaks his.
10034	High Heels and Low Lifs	High Heels and Low Lifs	Released	They're not bad girls. They're just behaving badly.
10040	Miss Conoenality 2: Armed and Fabulous	Miss Conoenality 2: Armed and Fabulous	Released	You might feel a little prick
100402	Captain America: The Winter Soldier	Captain America: The Winter Soldier	Released	In heroes we trust.
10045	District B13	Banlieue 13	Released	Welcome to District B13. Welcome to the future.
10047	The Messenger: The Story of Joan of Arc	Joan of Arc	Released	
10048	Stealth	Stealth	Released	Fear The Sky
10050	Get Over It	Get Over It	Released	Young, Free And Single Again

Conceptualisation de la base de données

- Import des fichiers .csv dans chaque table via « wizard import »

The screenshot shows a database management interface. On the left, the 'SCHEMAS' panel displays a tree view of the database structure. The 'film_popularity' table is selected, and its columns are listed: avg_vote, vote_count, popularity_note, and id_popularity. The 'Columns' section shows the data types: avg_vote (double), vote_count (double), popularity_note (double), and id_popularity (int(11) AI PK). The 'Result Grid' on the right displays the data for the 'film_popularity' table, showing columns: avg_vote, vote_count, popularity_note, and id_popularity. The data is limited to 50,000 rows. The table contains 26 rows of data.

avg_vote	vote_count	popularity_note	id_popularity
7.2	11800	150.437577	1
6.9	4500	139.082615	2
6.3	4466	107.376788	3
7.6	9106	112.31295	4
6.1	2124	43.926995	5
5.9	3576	115.699814	6
7.4	3330	48.681969	7
7.3	6767	134.27922900000002	8
7.4	5293	98.885637	9
5.7	7004	155.790452	10
5.4	1400	57.925623	11
6.1	2965	107.928811	12
7	5246	145.847379000000005	13
5.9	2311	49.046956	14
6.5	6359	99.398009	15
6.3	1630	53.978602	16
7.4	11776	144.448633	17
6.4	4948	135.413856	18
6.2	4160	52.035179	19
7.1	4760	120.965743	20
6.5	6586	89.866276	21
6.2	1398	37.668301	22
7.6	4524	94.370564	23
5.8	1303	42.990906	24
6.6	2337	61.22601	25
7.5	7562	100.025899	26

The screenshot shows the 'Table Details' for the 'films_database.film_popularity' table. The details include:

- Engine: InnoDB
- Row format: Dynamic
- Column count: 4
- Table rows: 4803
- AVG row length: 57
- Data length: 272.0 KiB
- Index length: 0.0 bytes
- Max data length: 0.0 bytes
- Data free: 0.0 bytes
- Table size (estimate): 272.0 KiB
- Update time:
- Create time: 2019-12-18 11:10:37
- Auto increment: 4804
- Table collation: utf8_general_ci
- Create options:
- Comment:

Information on this page may be outdated. Click [Analyze Table](#)

Utilisation de Pymysql

- Connection entre notebook et Workbench pour importer les tables de la base :

```
1]: import pymysql.cursors
```

```
2]: pip install MySQLdb
```

Collecting MySQLdb

Note: you may need to restart the kernel to use updated packages.

ERROR: Could not find a version that satisfies the requirement MySQLdb (from versions: none)

ERROR: No matching distribution found for MySQLdb

```
5]: import pymysql.cursors
import pymysql
```

Connect to the database

```
con = pymysql.connect(host='localhost',
                      user='root',
                      password='root',
                      db='films_database',
                      charset='utf8mb4',
                      cursorclass=pymysql.cursors.DictCursor)
```

```
34]: with con:
```

```
    cur = con.cursor()
```

Utilisation de Pymysql

- Connection entre notebook et Workbench pour importer les tables de la base :

```
In [38]: actors=pd.DataFrame(actors)
         film=pd.DataFrame(film)
```

```
In [40]: actors.head()
```

```
Out[40]:
```

	id_cast	prenomnom_cast
0	1	SamWorthington
1	2	JohnnyDepp
2	3	DanielCraig
3	4	ChristianBale
4	5	TaylorKitsch

```
In [27]: film.head()
```

```
Out[27]:
```

	id_film	title_film	title_original	release_status	tagline	release_date	runtime	homepage
0	100	Lock, Stock and Two Smoking Barrels	Lock, Stock and Two Smoking Barrels	Released	A Disgrace to Criminals Everywhere.	1998-03-05	105.0	http://www.universalstudiosentertainment.com/l... ambush,alcohol,shotgun,tea,joint,r

Analyse des données

- Analyse limitée aux 10 plus importants résultats

```
ée [382]: #highest revenues  
hr = br.groupby('title').film_revenues.max().apply(lambda x: x/1e9).sort_values(ascending=False).head(10)
```

```
[382]: title  
Avatar                2.787965  
Titanic               1.845034  
The Avengers          1.519558  
Jurassic World        1.513529  
Furious 7             1.506249  
Avengers: Age of Ultron 1.405404  
Frozen                1.274219  
Iron Man 3            1.215440  
Minions               1.156731  
Captain America: Civil War 1.153304  
Name: film_revenues, dtype: float64
```

```
383]: #highest budget  
hb = br.groupby('title').film_budget.max().apply(lambda x: x/1e6).sort_values(ascending=False).head(10)  
hb
```

```
: title  
Pirates of the Caribbean: On Stranger Tides 380.0  
Pirates of the Caribbean: At World's End 300.0  
Avengers: Age of Ultron 280.0  
Superman Returns 270.0  
John Carter 260.0  
Tangled 260.0  
Spider-Man 3 258.0  
The Lone Ranger 255.0  
The Hobbit: The Desolation of Smaug 250.0  
The Hobbit: An Unexpected Journey 250.0  
Name: film_budget, dtype: float64
```

Analyse des données

- Analyse limitée aux 10 plus importants résultats

```
82]: hp = film.merge(pop)
```

```
86]: hp_pop = hp.groupby('title').popularity_note.max().round(2).sort_values(ascending=False).head(10)
```

```
87]: hp_pop
```

title	
Minions	875.58
Interstellar	724.25
Deadpool	514.57
Guardians of the Galaxy	481.10
Mad Max: Fury Road	434.28
Jurassic World	418.71
Pirates of the Caribbean: The Curse of the Black Pearl	271.97
Dawn of the Planet of the Apes	243.79
The Hunger Games: Mockingjay - Part 1	206.23
Big Hero 6	203.73
Name: popularity_note, dtype: float64	

Analyse des données

- Analyse limitée aux 10 plus importants résultats

Which year has seen maximum release of movies

```
Entrée [187]: #transformer la colonne année en valeur entière  
film.release_year = pd.array(film.release_year, dtype="Int64")
```

```
Entrée [202]: by = film.groupby('release_year').title.count().sort_values(ascending=False).head(10)
```

```
Entrée [203]: by
```

```
Out[203]: release_year  
2009      247  
2014      238  
2006      237  
2013      231  
2008      227  
2010      225  
2011      223  
2005      217  
2015      216  
2012      208  
Name: title, dtype: int64
```

Analyse des données

- Analyse limitée aux 10 plus importants résultats

Artist with most movies

```
Entrée [205]: am = film.merge(actors)
```

```
Entrée [320]: ma = am.groupby('actors').actors.count().sort_values(ascending=False).head(10)
ma
```

```
Out[320]: actors
BruceWillis      30
RobertDeNiro     30
NicolasCage      28
JohnnyDepp       27
DenzelWashington 25
TomHanks         24
MattDamon        23
AdamSandler      23
TomCruise        23
ArnoldSchwarzenegger 23
Name: actors, dtype: int64
```


Analyse des données

- Analyse limitée aux 10 plus importants résultats

```
...and then merge budget movies
```

```
e [211]: abm = am.merge(budget)
```

```
e [213]: abm_gp = abm.groupby(['actors', 'title']).film_budget.max().sort_values(ascending=False).head(10)
```

```
e [376]: abm_gp = abm_gp.reset_index()  
abm_gp
```

```
[76]:
```

	index	actors	title	film_budget
0	0	JohnnyDepp	Pirates of the Caribbean: On Stranger Tides	3800000000
1	1	JohnnyDepp	Pirates of the Caribbean: At World's End	3000000000
2	2	RobertDowneyJr.	Avengers: Age of Ultron	2800000000
3	3	BrandonRouth	Superman Returns	2700000000
4	4	TaylorKitsch	John Carter	2600000000
5	5	ZacharyLevi	Tangled	2600000000
6	6	TobeyMaguire	Spider-Man 3	2580000000

Visualisation des données

- Visualisation avec Matplotlib

Analyse graphique des résultats

```
Entrée [242]: #fig= plt.figure()
```

```
Entrée [351]: ax1 = plt.subplot(4,2,1)
x=hr.index
y=hr.values
plt.barh(x, y, color='y')
plt.title('10 Highest films revenues (MD$)')
plt.gca().invert_yaxis()

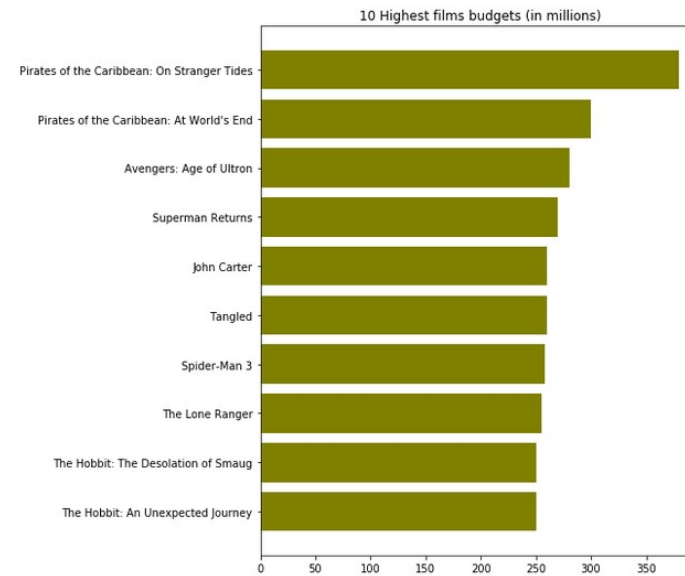
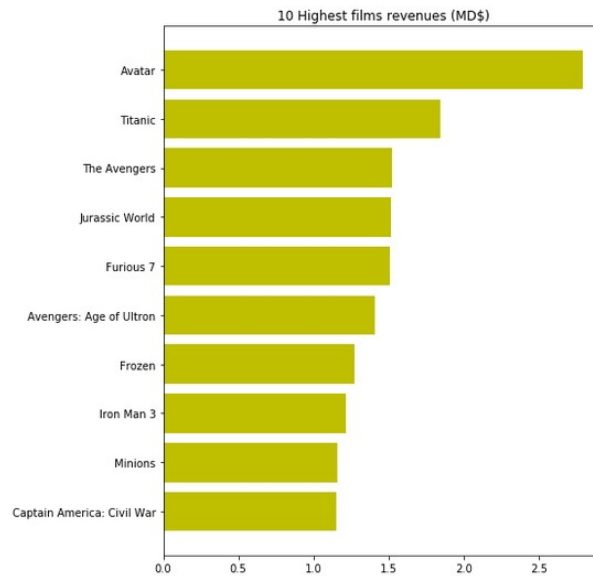
ax2 = plt.subplot(4,2,2)
x=hb.index
y=hb.values
plt.barh(x, y, color='olive')
plt.title('10 Highest films budgets (in millions)')
plt.gca().invert_yaxis()

ax3 = plt.subplot(4,2,3)
x=hp_pop.index
y=hp_pop.values
plt.barh(x, y, color='gold')
plt.title('Movies with the Highest popularity')
```

Visualisation des données

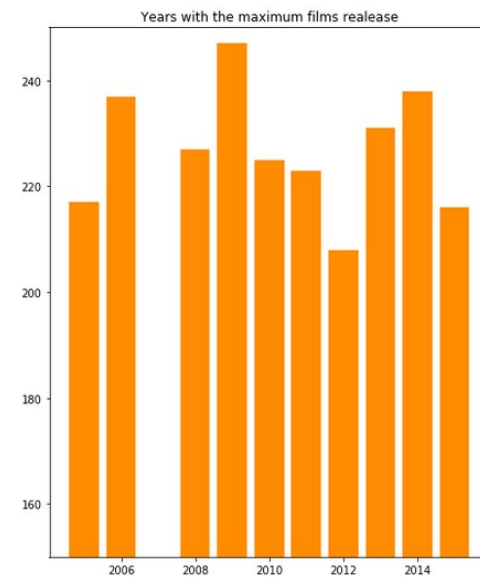
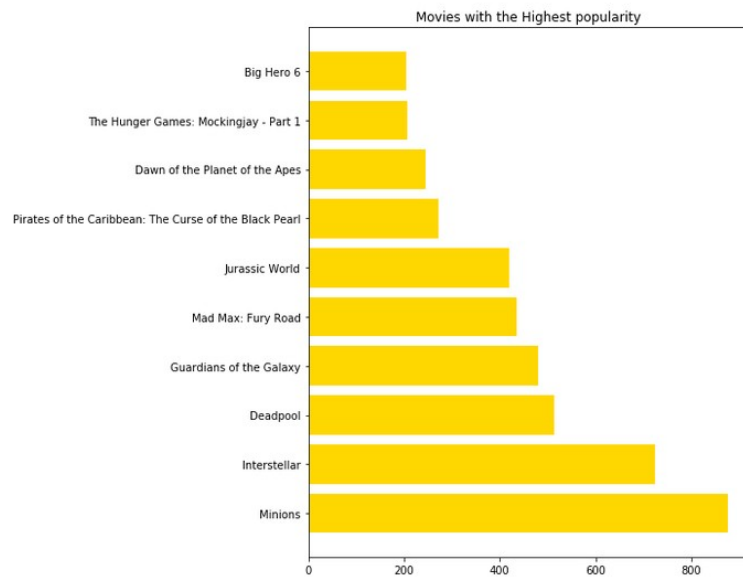
- Visualisation avec Matplotlib

```
plt.show()
```



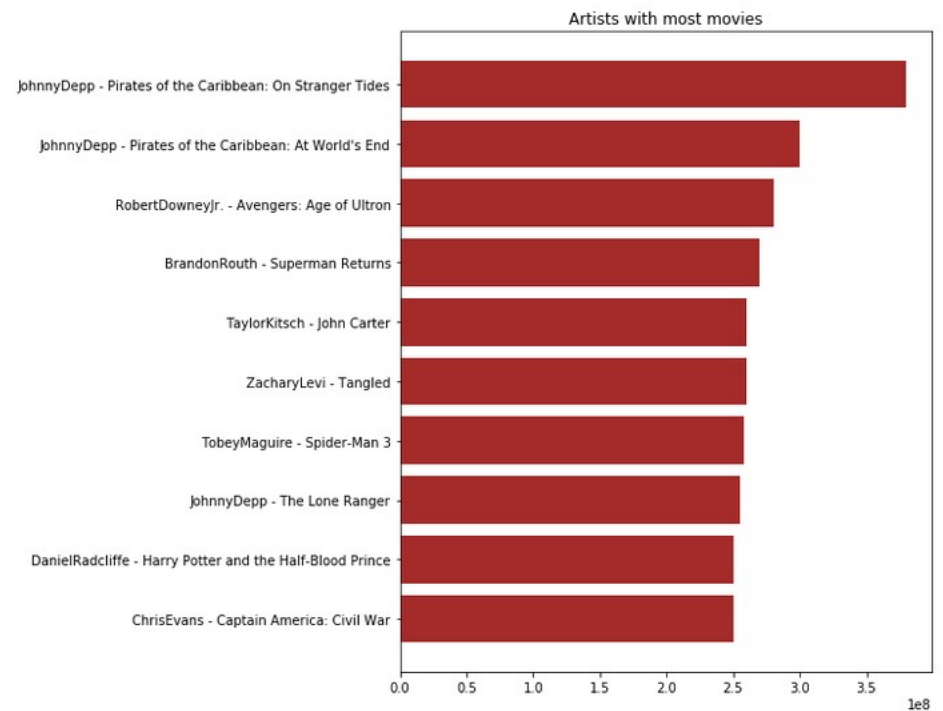
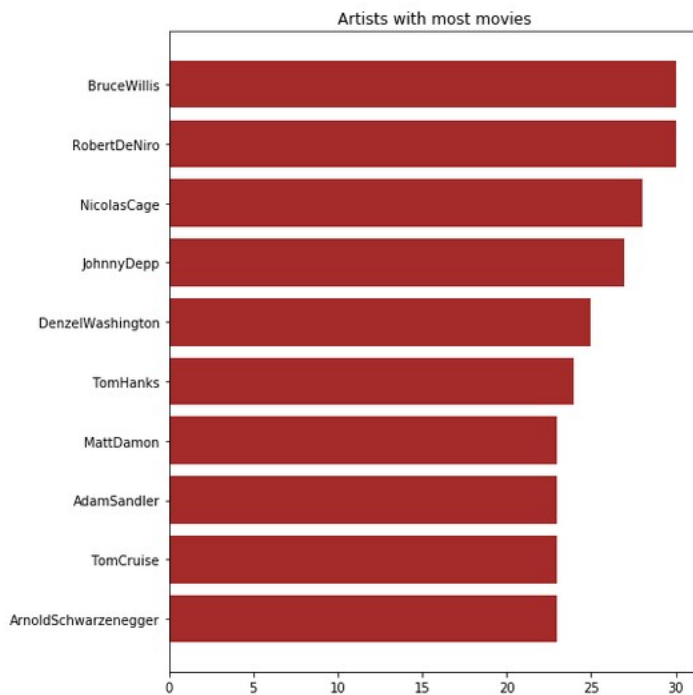
Visualisation des données

- Visualisation avec Matplotlib



Visualisation des données

- Visualisation avec Matplotlib



Si j'avais pu aller plus loin...

1) Pour la création de la base :

- j'aurais effectué un nettoyage approfondi de chaque fichier .csv avant de les importer dans mysql :
- j'aurais séparer nom et prénom, vérifier et corriger les valeurs placées au mauvais endroit,
- j'aurais créer une table pour chaque type de genre avec une relation *many to many* avec la table 'film', idem pour keywords de la table film

Si j'avais pu aller plus loin...

2) Pour l'analyse:

- j'aurais ajouté une analyse sur les réalisateurs
- j'aurais établi un lien à partir de chacun des graphiques visuellement mais aussi en créant un code pour récupérer les connexions entre chacun de ces graphiques et en sortir un nouveau tableau de synthèse et ainsi déterminer un lien possible entre réussite commerciale et les différents paramètres d'analyse d'un film.
- Il aurait été intéressant de pouvoir regrouper ces données avec les données sur le profil des spectateurs
- Des données sur la ventilation des recettes (video, vod, salle, tv...) aurait également permis d'enrichir l'analyse.

