Store data

As calling API everytime may takes a while and we may hit the limit of API, we will store the data from API for now.

In [3]:
```
import tmdbsimple as tmdb
tmdb.API_KEY = "71e259894a515060876bab2a33d6bdc9"
```

In [4]:
```
import imdb as ib
from imdb import IMDb
import pandas as pd
from PIL import Image
from StringIO import StringIO
import requests
import os
import time
from shutil import copyfile
import types
import numpy as np
```

In [5]:
```
dir_python_notebook = os.getcwd()
dir_movie_project = os.path.abspath(os.path.join(dir_python_notebook, os.
dir_data = os.path.join(dir_movie_project, 'data')
```

IMDB

In [6]:
```
imdb = IMDb()
#get a movie by id
imdb_movie = imdb.get_movie('0325980')
#access attributes of the movie by dictionary keys
imdb.update(imdb_movie)
#available keys
imdb_info_column  =imdb_movie.keys()
imdb_info_column  = [str(c) for c in imdb_info_column]
imdb_info_column.sort()
```

In [7]:
```
print(len(imdb_info_column))
print(imdb_info_column)
```

```
61
['akas', 'animation department', 'art department', 'art direction', 'a
spect ratio', 'assistant director', 'camera and electrical department'
, 'canonical title', 'cast', 'casting department', 'casting director',
'certificates', 'cinematographer', 'color info', 'costume department',
'costume designer', 'countries', 'country codes', 'cover url', 'direct
or', 'distributors', 'editor', 'editorial department', 'full-size cove
r url', 'genres', 'kind', 'language codes', 'languages', 'location man
agement', 'long imdb canonical title', 'long imdb title', 'make up', '
miscellaneous companies', 'miscellaneous crew', 'mpaa', 'music departm
ent', 'original music', 'plot', 'plot outline', 'producer', 'productio
n companies', 'production design', 'production manager', 'rating', 'ru
ntimes', 'set decoration', 'smart canonical title', 'smart long imdb c
anonical title', 'sound crew', 'sound mix', 'special effects companies
', 'special effects department', 'stunt performer', 'thanks', 'title',
'top 250 rank', 'transportation department', 'visual effects', 'votes'
, 'writer', 'year']
```

In [8]:
```
# IMDB API shows that we can retrieve information from proprty of movies
imdb_info_column = imdb_movie.keys_alias.keys()
imdb_info_column.sort()
print(len(imdb_info_column))
print(imdb_info_column)
```

```
85
['actors', 'actresses', 'aka', 'also known as', 'amazon review', 'art
direction by', 'casting', 'casting by', 'certificate', 'certification'
, 'certifications', 'cinematography', 'cinematography by', 'color', 'c
ostume and wardrobe department', 'costume design', 'costume design by'
, 'country', 'cover', 'created by', 'crew members', 'crewmembers', 'di
rected by', 'distribution', 'distribution companies', 'distribution co
mpany', 'distributor', 'editing', 'episodes cast', 'episodes number',
'faq', 'film editing', 'film editing by', 'frequently asked questions'
, 'full-size cover', 'genre', 'guest', 'guest appearances', 'lang', 'l
anguage', 'make-up', 'makeup', 'makeup department', 'merchandise', 'me
rchandising', 'misc companies', 'misc company', 'misc crew', 'miscella
neous', 'miscellaneous company', 'miscellaneous links', 'miscellaneous
crew', 'music', 'non-original music by', 'notable tv guest appearances
', 'original music by', 'other companies', 'other company', 'other cre
w', 'parental guide', 'photographs', 'plot summaries', 'plot summary',
'produced by', 'production company', 'production countries', 'producti
on country', 'production management', 'runtime', 'sales', 'seasons', '
second unit director', 'second unit director or assistant director', '
set decoration by', 'sound department', 'soundclips', 'special effects
by', 'special effects company', 'stunts', 'tv guests', 'tv schedule',
'user rating', 'videoclips', 'visual effects by', 'writing credits']
```

In [9]:
```
for c in imdb_info_column:
    if imdb_movie.get(c) is None:
        print(c)
    #else:
    #    print (imdb_movie.get(c))
```

```
amazon review
created by
episodes cast
episodes number
faq
frequently asked questions
full-size cover
guest
guest appearances
merchandise
merchandising
miscellaneous
miscellaneous links
non-original music by
notable tv guest appearances
parental guide
photographs
sales
seasons
soundclips
special effects by
tv guests
tv schedule
videoclips
```

In [10]:
```
for c in imdb_info_column:
    print(c)
    print(imdb_movie.get(c))
```

```
actors
[<Person id:0000136[http] name:_Depp, Johnny_>, <Person id:0001691[htt
p] name:_Rush, Geoffrey_>, <Person id:0089217[http] name:_Bloom, Orlan
do_>, <Person id:0461136[http] name:_Knightley, Keira_>, <Person id:02
02603[http] name:_Davenport, Jack_>, <Person id:0000596[http] name:_Pr
yce, Jonathan_>, <Person id:0034305[http] name:_Arenberg, Lee_>, <Pers
on id:0188871[http] name:_Crook, Mackenzie_>, <Person id:1400933[http]
name:_O'Hare, Damian_>, <Person id:1099016[http] name:_New, Giles_>, <
Person id:0055845[http] name:_Barnett, Angus_>, <Person id:0047549[htt
p] name:_Bailie, David_>, <Person id:1400913[http] name:_Jr., Michael
Berry_>, <Person id:0802280[http] name:_Jr., Isaac C. Singleton_>, <Pe
rson id:0573618[http] name:_McNally, Kevin_>, <Person id:0262125[http]
name:_Etienne, Treva_>, <Person id:0757855[http] name:_Saldana, Zoe_>,
<Person id:0801838[http] name:_Siner, Guy_>, <Person id:0552924[http]
name:_Martin, Ralph P._>, <Person id:0628225[http] name:_Newman, Paula
J._>, <Person id:0445284[http] name:_Keith, Paul_>, <Person id:0808057
[http] name:_Smith, Dylan_>, <Person id:1096355[http] name:_Dryzek, Lu
cinda_>, <Person id:0212472[http] name:_de Woolfson, Luke_>, <Person i
d:0992126[http] name:_Tighe, Michael Sean_>, <Person id:0254862[http]
```

In [11]:
```
# IMDB API shows that we can retrieve information from proprty of movies
imdb_info_column2 = imdb_movie.keys_alias.values()
imdb_info_column2 = list(set(imdb_info_column2))
imdb_info_column2.sort()
print(len(imdb_info_column2))
print(imdb_info_column2)
```

```
49
['airing', 'akas', 'amazon reviews', 'art direction', 'assistant direc
tor', 'cast', 'casting director', 'certificates', 'cinematographer', '
color info', 'costume department', 'costume designer', 'countries', 'c
over url', 'creator', 'director', 'distributors', 'editor', 'faqs', 'f
ull-size cover url', 'genres', 'guests', 'languages', 'make up', 'merc
handising links', 'misc links', 'miscellaneous companies', 'miscellane
ous crew', 'non-original music', 'number of episodes', 'number of seas
ons', 'original music', 'parents guide', 'photo sites', 'plot', 'produ
cer', 'production companies', 'production manager', 'rating', 'runtime
s', 'set decoration', 'sound clips', 'sound crew', 'special effects',
'special effects companies', 'stunt performer', 'video clips', 'visual
effects', 'writer']
```

In [12]:
```
for c in imdb_info_column2:
    print(c)
    if (imdb_movie.has_key(c)):
        print(imdb_movie[c])
    else:
        print(None)
```

```
xe9rola Negra::Brazil (imdb display title)', u'Pirates des Cara\xefbes
- La mal\xe9diction de la Perle Noire::Canada (French title)', u'Pirat
es des Cara\xefbes - La mal\xe9diction du Black Pearl::France (imdb di
splay title)', u'Pirates of the Caribbean: Den sorte forbandelse::Denm
ark (imdb display title)', u'Pirates of the Caribbean: Mustan helmen k
irous::Finland', u'Pirates of the Caribbean: Svarta P\xe4rlans f\xf6rb
annelse::Sweden (imdb display title)', u'Pirati dei Caraibi: La maledi
zione della prima luna::Italy (alternative title)', u'Pirati s Kariba:
Prokletstvo Crnog bisera::Croatia (imdb display title)', u'Pirati s Ka
ribov: prekletstvo \u010drnega bisera::Slovenia', u'Pirati sa Kariba -
Prokletstvo Crnog bisera::Serbia (imdb display title)', u'Piratii din
Caraibe: Blestemul Perlei Negre::Romania (imdb display title)', u"Shod
eday Ha-Caribim: Klalat Ha-Pnina Ha-Sh'hora::Israel (Hebrew title)", u
'Sj\xf3r\xe6ningjar \xe1 Kar\xedbahafi: B\xf6lvun sv\xf6rtu perlunnar:
:Iceland (imdb display title)']
amazon reviews
None
art direction
[<Person id:0384192[http] name:_Hill, Derek R._>, <Person id:0694358[h
ttp] name: Powels, Michael >, <Person id:0865042[http] name: Tocci, Ja
```

In [14]:
```
tmdb_filename = str(dir_data)+'\\drv_tmdb_movie_details.json'
tmdb_movies = pd.read_json(tmdb_filename)
tmdb_movies.head(1)
```

Out[14]:

	adult	backdrop_path	belongs_to_collection	budget	genres
100	False	/kzeR7BA0htJ7Bel6QEUX3PVp39s.jpg	None	1350000	[Comedy Crime]

1 rows × 25 columns

In [15]:
```
# get IMDB movies that we already have data for TMDB
imdb_ids = tmdb_movies['imdb_id'].tolist()
tmdb_ids = tmdb_movies['id'].tolist()

imdb_ids = [ str(imdb_id.replace('tt','')) for imdb_id in imdb_ids]
imdb_ids = [ imdb_id for imdb_id in imdb_ids if imdb_id !='' and imdb_id

imdb_ids = list(set(imdb_ids))
```

In [16]:
```
tmdb_movies.head(1)
```

Out[16]:

	adult	backdrop_path	belongs_to_collection	budget	genres
100	False	/kzeR7BA0htJ7BeI6QEUX3PVp39s.jpg	None	1350000	[Comedy Crime]

1 rows × 25 columns

In [17]:
```
# verify that we get the same movie as TMDB

imdb = IMDb()
imdb_movie = imdb.get_movie(imdb_id)
#access attributes of the movie by dictionary keys
imdb.update(imdb_movie)
imdb_movie['title']
```

Out[17]:
```
u'Der freie Wille'
```

In [18]:
```
print(imdb_movie)
```

```
Der freie Wille
```

In [115]:
```
imdb_info_column = imdb_movie.keys_alias.values()
imdb_info_column = list(set(imdb_info_column))
if 'imdb_id' not in imdb_info_column:
    imdb_info_column.append('imdb_id')
imdb_info_column.sort()

movies = pd.DataFrame(columns=imdb_info_column)
invalid_imdb_ids = []

imdb_filename = str(dir_data)+'\\drv_imdb_movie_info.json'
imdb_filename_backup = str(dir_data)+'\\drv_imdb_movie_info_bkp.json'
imdb_invalid_id_filename = str(dir_data)+'\\drv_imdb_movie_invalid_id.jso
count = 0
```

```
count = 0
# if we already have a movie data file, we can just continue appending it
if os.path.isfile(imdb_filename):
    movies = pd.read_json(imdb_filename)
    if len(movies['imdb_id'].tolist()) >0:
        movie_ids = movies['imdb_id'].tolist()
        imdb_ids = [x for x in imdb_ids if x not in movie_ids]
        imdb_info_column = movies.columns
    else:
        movies = pd.DataFrame(columns=imdb_info_column)

# we don't want to waste our effort to load invalid imdb_id
if os.path.isfile(imdb_invalid_id_filename):
    invalid_imdb_id_df = pd.read_json(imdb_invalid_id_filename)
    if len(invalid_imdb_id_df['invalid_imdb_id'].tolist()) >0:
        invalid_imdb_ids = invalid_imdb_id_df['invalid_imdb_id'].tolist()
        imdb_ids = [x for x in imdb_ids if x not in invalid_imdb_ids]

for i in imdb_ids:
    count += 1
    if (count % 500 == 0):
        movies.to_json(path_or_buf= imdb_filename)
        if (len(invalid_imdb_ids)>0):
            invalid_imdb_id_df = pd.DataFrame({'invalid_imdb_id': invalid
            invalid_imdb_id_df.to_json(path_or_buf= imdb_invalid_id_filen
    #skip the non-existing movie ids
    #print(i)
    try:
        imdb_movie = imdb.get_movie(i)
        imdb.update(imdb_movie)
        if (len(imdb_movie.keys()) > 0) :
            movie_details = []
            for c in imdb_info_column:
                if imdb_movie.has_key(c):
                    info_field = imdb_movie[c]
                    if info_field is not None:
                        if type(info_field) is list:
                            if isinstance(info_field[0], ib.Person.Person
                                info_list = []
                                for item in info_field:
                                    info_list.append(item.getID())
                                movie_details.append(info_list)
                            elif isinstance(info_field[0], ib.Company.Com
                                info_list = []
                                for item in info_field:
                                    info_list.append(item.getID())
                                movie_details.append(info_list)
                            else:
                                movie_details.append(info_field)
                        else:
                            movie_details.append(info_field)
```

```
                    else:
                        if c == "imdb_id":
                            movie_details.append(i)
                        else:
                            movie_details.append(None)

                else:
                        movie_details.append(None)

            movies.loc[len(movies.index)] = movie_details

            movies.head(5)
    except Exception:
        invalid_imdb_ids.append(i)
        continue

movies.to_json(path_or_buf= imdb_filename)
if (len(invalid_imdb_ids)>0):
    invalid_imdb_id_df = pd.DataFrame({'invalid_imdb_id': invalid_imdb_id
    invalid_imdb_id_df.to_json(path_or_buf= imdb_invalid_id_filename)
```

```
2017-04-05 12:31:48,582 CRITICAL [imdbpy] C:\Users\Chrystal\Anaconda2\
envs\py27\lib\site-packages\imdb\_exceptions.py:35: IMDbDataAccessErro
r exception raised; args: ({'exception type': 'IOError', 'url': 'http:
//akas.imdb.com/title/tt0072996/combined', 'errcode': 'socket error',
'proxy': '', 'original exception': IOError('socket error', error(10060
, 'A connection attempt failed because the connected party did not pro
perly respond after a period of time, or established connection failed
because connected host has failed to respond')), 'errmsg': '[Errno 100
60] A connection attempt failed because the connected party did not pr
operly respond after a period of time, or established connection faile
d because connected host has failed to respond'},); kwds: {}
Traceback (most recent call last):
  File "C:\Users\Chrystal\Anaconda2\envs\py27\lib\site-packages\imdb\p
arser\http\__init__.py", line 202, in retrieve_unicode
    uopener = self.open(url)
  File "C:\Users\Chrystal\Anaconda2\envs\py27\lib\urllib.py", line 213
, in open
    return getattr(self, name)(url)
  File "C:\Users\Chrystal\Anaconda2\envs\py27\lib\urllib.py", line 350
  in open_http
```

In [116]:
```
#make a backup in case of corruption
copyfile(imdb_filename, imdb_filename_backup)
```

In [117]:
```
imdb_filename = str(dir_data)+'\\drv_imdb_movie_info.json'
imdb_movies = pd.read_json(imdb_filename)
imdb_movies.head(5)
```

Out[117]:

	airing	akas	amazon reviews	art direction	assistant director	cast	casting director	certificat
0	NaN	[Ultimate Avengers 2: Rise of the Panther::USA...	NaN	None	None	[1225431, 0217221, 0557219, 0941404, 0001882, ...	[0800493]	[Argentina (DVD ratir Russia:12 Russ...
1	NaN	[Bajo la piel::Argentina (imdb display title),...	NaN	None	[0116502, 0348808, 0601635, 0931271]	[0164809, 0891895, 0252961, 0862328, 0001026, ...	[0058089, 0338418]	[Argentina Australia: Netherlan
10	NaN	[Ultimate Avengers 2: Rise of the Panther::USA...	NaN	None	None	[1225431, 0217221, 0557219, 0941404, 0001882, ...	[0800493]	[Argentina (DVD ratir Russia:12 Russ...
100	NaN	[One Wild Night::, USA (working title), Возмож...	NaN	[0055618]	[0742835, 0815340]	[0001844, 0000124, 0000551, 0612487, 0413698, ...	[0228938]	[Australia Germany: Iceland:L Kor...
1000	NaN	[Wicked City::Australia (imdb display title), ...	NaN	[0644470]	None	[0946344, 0810987, 0589645, 0297847, 0448950, ...	[0532146, 0810987, 3108562]	[Australia Canada:1 (Alberta/E ...

5 rows × 50 columns

In [ ]: