Team Project: ”The Movie Dataset”

Course: Databases and Big Data

Group 8:
Fabrizio Rocco, Davide Rosatelli, Davide Manniello, Roberto Colangelo

January 16, 2020

1 Abstract

This report contains a detailed analysis of the "Movies" dataset as requested by the assignment for the "Databases and Big Data" exam of the LUISS Management and Computer Science course.
The assignment asked for the aforementioned dataset to be imported and analyzed into three different DBMSs, namely: Oracle MySQL, MongoDB and Apache Spark.
The specific dataset presented several challenges, particularily bringing it into 3NF. After the design and import phase we developed four queries for each DBMS in order to retrieve:

i): The actor who acted in the most movies

ii): For each year, retrieve the best rated movie

iii): For each year, the best rated genre, the most revenued genre, and the best rated movie that revenued the most.

iv): For each year, the ranking of the top 10 european countries for movie revenues

We then analyzed the performance of each query in each DBMS by executing each query on the same hardware and by measuring the running time of each query.

2 Dataset choice

The assignment asks for a dataset to be chosen among two possible datasets:

A. https://www.kaggle.com/hugomathien/soccer

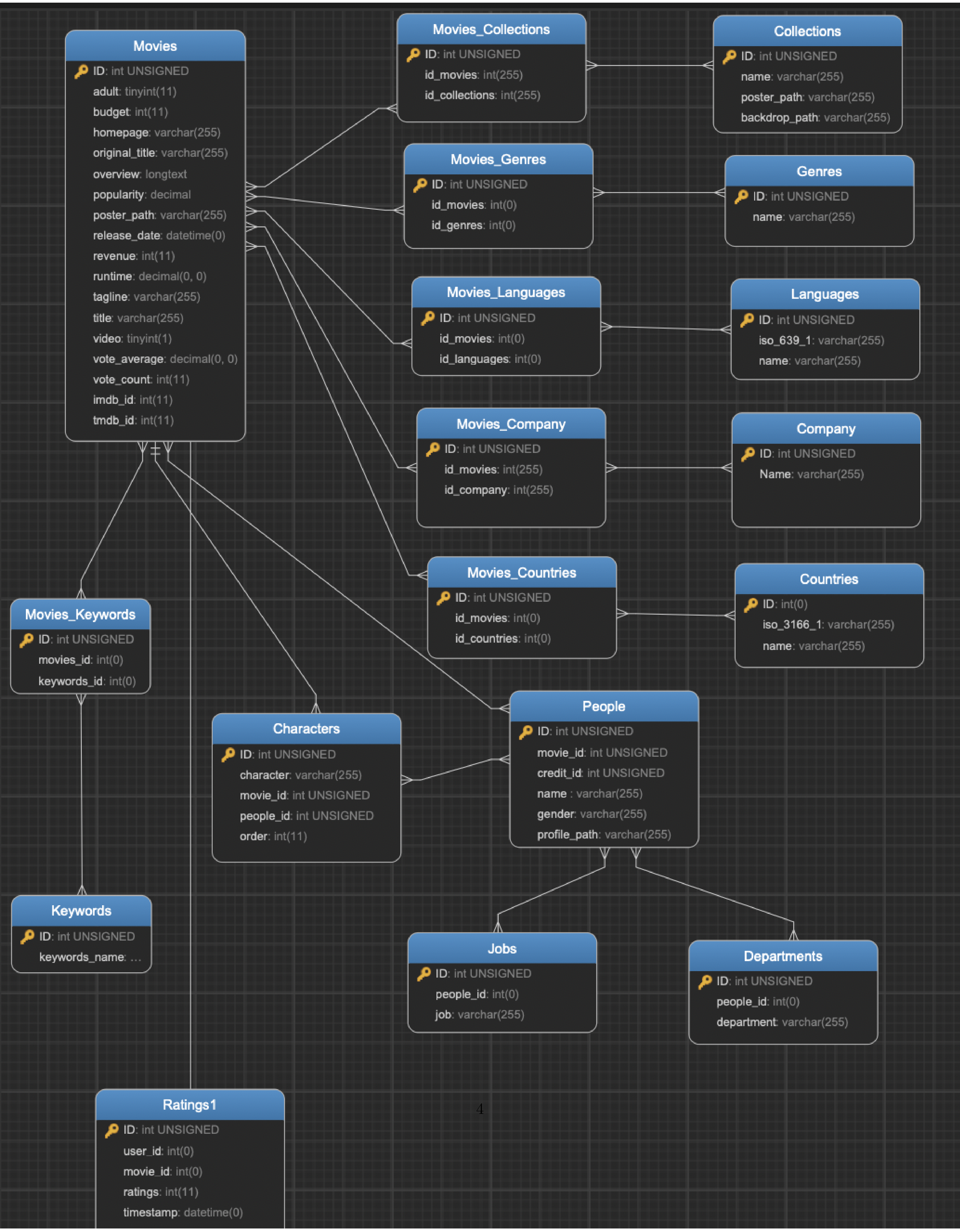
B. https://www.kaggle.com/rounakbanik/the-movies-dataset

We have chosen the second dataset since we found it more challenging and stimulating than the first one.
In particular we had to face the problem of the dataset not being in 1NF, with several fields holding serialized content, therefore we had to implement a JSON parser in the data import script.
Another challenge posed by this specific dataset is the design of a suitable ER schema, while designing it we took into account performance and the specific queries we had to run.

3 Database design and normalization

The dataset is provided in CSV format, with several JSON-serialized fields present.
We developed a Python script that normalizes the database and then imports it into MySQL.
The full Python script is detailed in the following section "Data import".
The script parses all JSON serialized fields and brings the dataset in 3NF.
Following we include the Entity-Relationship schema that we used to process the data in

Oracle MySQL and Apache Spark.


Movies
ID: int UNSIGNED
adult: tinyint(11)
budget: int(11)
homepage: varchar(255)
original_title: varchar(255)
overview: longtext
popularity: decimal
poster_path: varchar(255)
release_date: datetime(0)
revenue: int(11)
runtime: decimal(0, 0)
tagline: varchar(255)
title: varchar(255)
video: tinyint(1)
vote_average: decimal(0, 0)
vote_count: int(11)
imdb_id: int(11)
tmdb_id: int(11)
Movies_Collections
ID: int UNSIGNED
id_movies: int(255)
id_collections: int(255)
Collections
ID: int UNSIGNED
name: varchar(255)
poster_path: varchar(255)
backdrop_path: varchar(255)
Movies_Genres
ID: int UNSIGNED
id_movies: int(0)
id_genres: int(0)
Genres
ID: int UNSIGNED
name: varchar(255)
Movies_Languages
ID: int UNSIGNED
id_movies: int(0)
id_languages: int(0)
Languages
ID: int UNSIGNED
iso_639_1: varchar(255)
name: varchar(255)
Movies_Company
ID: int UNSIGNED
id_movies: int(255)
id_company: int(255)
Company
ID: int UNSIGNED
Name: varchar(255)
Movies_Countries
ID: int UNSIGNED
id_movies: int(0)
id_countries: int(0)
Countries
ID: int(0)
iso_3166_1: varchar(255)
name: varchar(255)
Movies_Keywords
ID: int UNSIGNED
movies_id: int(0)
keywords_id: int(0)
Characters
ID: int UNSIGNED
character: varchar(255)
movie_id: int UNSIGNED
people_id: int UNSIGNED
order: int(11)
People
ID: int UNSIGNED
movie_id: int UNSIGNED
credit_id: int UNSIGNED
name : varchar(255)
gender: varchar(255)
profile_path: varchar(255)
Keywords
ID: int UNSIGNED
keywords_name: ...
Jobs
ID: int UNSIGNED
people_id: int(0)
job: varchar(255)
Departments
ID: int UNSIGNED
people_id: int(0)
department: varchar(255)
Ratings1
ID: int UNSIGNED
user_id: int(0)
movie_id: int(0)
ratings: int(11)
timestamp: datetime(0)

4 Data import

Oracle MySQL import

Since the normalization script is written in Python we decided to use mysql-connector-python to directly import the data into MySQL as soon as each row is processed and normalized.

Following we include one of the import scripts that normalizes the dataset in 3NF and imports it into Oracle MySQL. All the other import scripts behave similarly and have been redacted for clarity.

```
import mysql.connector as mysql
import os
import csv
import json
import ast
import time
import datetime

db = mysql.connect(
    host = "localhost",
    user = "REDACTED",
    password = "REDACTED",
    database="movies"
)

cursor = db.cursor()

q1="CREATE TABLE IF NOT EXISTS movies (id INT(11) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
     _id INT(11), imdbId INT(11), adult BOOL, budget INT(11), homepage VARCHAR(255),
     original_title NVARCHAR(255), overview LONGTEXT, popularity DECIMAL, poster_path
     VARCHAR(255), release_date DATETIME, revenue BIGINT, runtime DECIMAL, status
     NVARCHAR(255), tagline NVARCHAR(255), title NVARCHAR(255), video BOOL, vote_average
     DECIMAL, vote_count INT(11), imdb_id INT(11), tmdb_id INT(11))"
cursor.execute(q1)
q2="CREATE TABLE IF NOT EXISTS collections (id INT(11) UNSIGNED AUTO_INCREMENT PRIMARY
     KEY, coll_id INT(11), name VARCHAR(255), poster_path VARCHAR(255), backdrop_path
     VARCHAR(255))"
cursor.execute(q2)
q3="CREATE TABLE IF NOT EXISTS movies_collection (id INT(11) UNSIGNED AUTO_INCREMENT
     PRIMARY KEY, id_movie INT(11), id_coll INT(11))"
cursor.execute(q3)
q4="CREATE TABLE IF NOT EXISTS genres (id INT(11) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
     genre_id INT(11), name VARCHAR(255))"
cursor.execute(q4)
q5="CREATE TABLE IF NOT EXISTS movies_genres (id INT(11) UNSIGNED AUTO_INCREMENT PRIMARY
     KEY, id_movie INT(11), id_genre INT(11))"
cursor.execute(q5)
q6="CREATE TABLE IF NOT EXISTS languages (id INT(11) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
      iso_639_1 VARCHAR(255), name VARCHAR (255))"
cursor.execute(q6)
q7="CREATE TABLE IF NOT EXISTS movies_languages (id INT(11) UNSIGNED AUTO_INCREMENT
     PRIMARY KEY, id_movie INT(11), id_lang VARCHAR(255))"
cursor.execute(q7)
q8="CREATE TABLE IF NOT EXISTS production_companies (id INT(11) UNSIGNED AUTO_INCREMENT
     PRIMARY KEY, comp_id INT(11), name VARCHAR(255))"
cursor.execute(q8)
q9="CREATE TABLE IF NOT EXISTS movies_companies (id INT(11) UNSIGNED AUTO_INCREMENT
     PRIMARY KEY, id_movie INT(11), id_company INT(11))"
cursor.execute(q9)
q10="CREATE TABLE IF NOT EXISTS countries (id INT(11) UNSIGNED AUTO_INCREMENT PRIMARY KEY
     , iso_3166_1 VARCHAR(255), name VARCHAR(255))"
cursor.execute(q10)
q11="CREATE TABLE IF NOT EXISTS movies_countries (id INT(11) UNSIGNED AUTO_INCREMENT
     PRIMARY KEY, id_movie INT(11), id_country VARCHAR(255))"
```

```
cursor.execute(q11)

file = "movies_metadata.csv"

dataFile = []

def process_data(dataFile):
    for data in dataFile:
        columns_1 = "(_id, adult, budget, homepage, original_title, overview, popularity,
             poster_path, release_date, revenue, runtime,status, tagline, title, video,
            vote_average, vote_count )"
        columns_2 = "(coll_id, name, poster_path, backdrop_path)"
        _id = str(data['id'])
        adult = str(data['adult'].lower())
        budget = str(data['budget'])
        homepage = str(data['homepage']).replace("'", "")
        original_title = str(data['original_title']).replace("'", "").replace("-", "").
            replace(",", "")
        overview = str(data['overview']).replace("'", "")
        if 'popularity' in data:
            popularity = str(data['popularity'])
        else:
            continue
        poster_path = str(data['poster_path']).replace("'", "''")
        release_date = str("STR_TO_DATE('" + data['release_date'] + "', '%Y-%m-%d')") if
            type(data['release_date'])==str else ""
        if 'revenue' not in data:
            continue
        revenue=str(data['revenue'])
        runtime=str(data['runtime'])
        if runtime == "":
            runtime = "0";
        status=str(data['status']).replace("'", "''")
        tagline=str(data['tagline']).replace("'", "''").replace("-", "''")
        title=str(data['title']).replace("'", "").replace("-", "").replace(",", "")
        video=str(data['video'].lower()) if type(data['video'])==str else ""
        vote_average=str(data['vote_average'])
        vote_count=str(data['vote_count'])
        values_1 = "(" + _id + ", " + adult + ", " + budget + ", '" + homepage + "', '" +
             original_title + "', '" + overview + "', " + popularity + ", '" +
            poster_path + "', " + release_date + ", " + revenue + ", " + runtime + ", '"
             + status + "', '" + tagline + "', '" + title + "', " + video + ", " +
            vote_average + ", " + vote_count + " )"
        sql_1 = "INSERT INTO movies " + columns_1 + " VALUES " + values_1
        cursor.execute(sql_1)
        belongs_to_collection=ast.literal_eval(data['belongs_to_collection']) if len(data
            ['belongs_to_collection']) > 0 else[]
        for i in belongs_to_collection:
            collection_id=str(belongs_to_collection['id'])
            collection_name=str(belongs_to_collection['name']).replace("'", "''")
            collection_poster_path=str(belongs_to_collection['poster_path']).replace("'",
                 "''")
            collection_backdrop_path=str(belongs_to_collection['backdrop_path']).replace(
                "'", "''")
            values_2 = "(" + collection_id + ", '" + collection_name + "', '" +
                collection_poster_path + "', '" + collection_backdrop_path + "')"
            sql_2 = "INSERT INTO collections " + columns_2 + " VALUES " + values_2
            sql_3 = "INSERT INTO movies_collection ( id_movie, id_coll) VALUES (" + _id +
                 ", " + collection_id + ")"
            cursor.execute(sql_2)
            cursor.execute(sql_3)
        if len(data['genres']) <= 4:
            genres = []
        else:
            genres=ast.literal_eval(data['genres']) if len(data['belongs_to_collection'])
                 > 0 else []
        for i in genres:
            genre_id=str(i['id'])
            genre_name=str(i['name']).replace("'", "''")
            sql_4 = "INSERT INTO genres ( genre_id, name) VALUES ( " + genre_id + ", '" +
                 genre_name + "' )"
            sql_5 = "INSERT INTO movies_genres (id_movie, id_genre) VALUES (" + _id + ",
                " + genre_id + ")"
            cursor.execute(sql_4)
            cursor.execute(sql_5)
        languages=ast.literal_eval(data['spoken_languages']) if len(data['
             spoken_languages']) > 0 else []
        for i in languages:
            lang_iso=str(i['iso_639_1'])
            lang_name=str(i['name']).replace("'", "''")
            sql_6 = "INSERT INTO languages ( iso_639_1, name) VALUES ('" + lang_iso + "',
                 '" + lang_name + "')"
            sql_7 = "INSERT INTO movies_languages ( id_movie, id_lang) VALUES (" + _id +
                 ", '" + lang_iso + "')"
            cursor.execute(sql_6)
            cursor.execute(sql_7)
        production_companies=ast.literal_eval(data['production_companies']) if len(data['
             production_companies']) > 0 else []
        for i in production_companies:
            company_name=str(i['name']).replace("'", "''")
            company_id=str(i['id'])
            sql_8 = "INSERT INTO production_companies ( comp_id, name) VALUES (" +
                 company_id + ", '" + company_name + "')"
            sql_9 = "INSERT INTO movies_companies ( id_movie, id_company) VALUES  (" +
                 _id + ", " + company_id + ")"
            cursor.execute(sql_8)
            cursor.execute(sql_9)

        countries=ast.literal_eval(data['production_countries']) if len(data['
             production_countries']) > 0 else []
        for i in countries:
            country_iso=str(i['iso_3166_1'])
            country_name=str(i['name']).replace("'", "''")
            sql_10 = "INSERT INTO countries ( iso_3166_1, name) VALUES ('" + country_iso
                 + "', '" + country_name + "')"
            sql_11 = "INSERT INTO movies_countries ( id_movie, id_country) VALUES (" +
                 _id + ", '" + country_iso + "')"
            cursor.execute(sql_10)
            cursor.execute(sql_11)
        db.commit()

def read_file(file):
    with open(file, encoding="utf8") as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        ind = 0
        intest = []
        for row in csv_reader:
            d = {}
            for i in range(len(row)):
                cell = row[i]
                try:
                    obj = json.loads(cell)
                    if ind > 0:
                        d[intest[i]] = obj
                except ValueError as e:
                     if ind == 0:
                         intest.append(cell)
                     else:
                        d[intest[i]] = cell
            if ind > 0:
                dataFile.append(d)
            ind += 1
    process_data(dataFile)
read_file(file)
```

In order to easily import the data into other DBMSs without reimplementing the import script we exported the MySQL database into CSV format using the INTO OUTFILE SQL query.
Each table has been exported in a separate CSV file.

```
SHOW TABLES;
SELECT * INTO outfile 'exported_characters_.csv' FIELDS TERMINATED BY ',' LINES
     TERMINATED BY '\n' FROM characters_;
SELECT * INTO outfile 'exported_collections.csv' FIELDS TERMINATED BY ',' LINES
     TERMINATED BY '\n' FROM collections;
SELECT * INTO outfile 'exported_countries.csv' FIELDS TERMINATED BY ',' LINES TERMINATED
     BY '\n' FROM countries;
SELECT * INTO outfile 'exported_departments.csv' FIELDS TERMINATED BY ',' LINES
     TERMINATED BY '\n' FROM departments;
SELECT * INTO outfile 'exported_genres.csv' FIELDS TERMINATED BY ',' LINES TERMINATED BY
     '\n' FROM genres;
SELECT * INTO outfile 'exported_jobs.csv' FIELDS TERMINATED BY ',' LINES TERMINATED BY '\
     n' FROM jobs;
SELECT * INTO outfile 'exported_keywords.csv' FIELDS TERMINATED BY ',' LINES TERMINATED
     BY '\n' FROM keywords;
SELECT * INTO outfile 'exported_languages.csv' FIELDS TERMINATED BY ',' LINES TERMINATED
     BY '\n' FROM languages;
SELECT * INTO outfile 'exported_movies.csv' FIELDS TERMINATED BY ',' LINES TERMINATED BY
     '\n' FROM movies;
SELECT * INTO outfile 'exported_movies_collection.csv' FIELDS TERMINATED BY ',' LINES
     TERMINATED BY '\n' FROM movies_collection;
SELECT * INTO outfile 'exported_movies_companies.csv' FIELDS TERMINATED BY ',' LINES
     TERMINATED BY '\n' FROM movies_companies;
SELECT * INTO outfile 'exported_movies_countries.csv' FIELDS TERMINATED BY ',' LINES
     TERMINATED BY '\n' FROM movies_countries;
SELECT * INTO outfile 'exported_movies_genres.csv' FIELDS TERMINATED BY ',' LINES
     TERMINATED BY '\n' FROM movies_genres;
SELECT * INTO outfile 'exported_movies_keywords.csv' FIELDS TERMINATED BY ',' LINES
     TERMINATED BY '\n' FROM movies_keywords;
SELECT * INTO outfile 'exported_movies_languages.csv' FIELDS TERMINATED BY ',' LINES
     TERMINATED BY '\n' FROM movies_languages;
SELECT * INTO outfile 'exported_people.csv' FIELDS TERMINATED BY ',' LINES TERMINATED BY
     '\n' FROM people;
SELECT * INTO outfile 'exported_production_companies.csv' FIELDS TERMINATED BY ',' LINES
     TERMINATED BY '\n' FROM production_companies;
SELECT * INTO outfile 'exported_ratings.csv' FIELDS TERMINATED BY ',' LINES TERMINATED BY
      '\n' FROM ratings;
```

This allowed us to more conveniently import the dataset into both MongoDB and Spark-SQL using already present tools instead of having to interface each DBMS to our custom data normalization script.
We then added an header line to each file so that column names are also included in the CSV file.
In order to obtain the header line the following query has been run for each table:

```
SHOW TABLES;

SELECT GROUP_CONCAT(CONCAT("'",COLUMN_NAME,"'")) from INFORMATION_SCHEMA.COLUMNS WHERE
     TABLE_NAME = 'characters_' AND TABLE_SCHEMA = 'movies' order BY ORDINAL_POSITION;
```

MongoDB import

In order to import into MongoDB server we used the *mongoimport* tool.
MongoImport is able to import CSV files as a collection and parses the first line of the CSV file to obtain field names.
Another advantage of using MongoImport is that we do not need to iterate over each line with a Python interpreter (which is relatively slow when compared to compiled code) and we do not need to run a query for each line we need to import. Therefore, while the MySQL import phase took several hours (mainly due to the ”ratings” table, that is around 20 mil-

lion rows long) the MongoDB import phase was completed just a few seconds shy of a minute.

```
for file in *.csv; do file2=$(echo ${file:9} | cut -d "." -f 1); echo $file2; mongoimport
      -u REDACTED -p REDACTED --host=127.0.0.1 -d movies -c $file2 --file $file --type
     csv --headerline --authenticationDatabase admin; done
characters_2
2020-01-15T23:37:10.157+0000    connected to: mongodb://127.0.0.1/
2020-01-15T23:37:13.157+0000    [#########..............] movies.characters_2  7.66MB
     /18.6MB (41.1%)
2020-01-15T23:37:16.157+0000    [###################.....] movies.characters_2  15.5MB
     /18.6MB (83.1%)
2020-01-15T23:37:17.372+0000    [########################] movies.characters_2  18.6MB
     /18.6MB (100.0%)
2020-01-15T23:37:17.372+0000    562474 document(s) imported successfully. 0 document(s)
     failed to import.
collections
2020-01-15T23:37:17.397+0000    connected to: mongodb://127.0.0.1/
2020-01-15T23:37:17.670+0000    17960 document(s) imported successfully. 0 document(s)
     failed to import.
countries
2020-01-15T23:37:17.696+0000    connected to: mongodb://127.0.0.1/
2020-01-15T23:37:18.263+0000    49423 document(s) imported successfully. 0 document(s)
     failed to import.
departments
2020-01-15T23:37:18.288+0000    connected to: mongodb://127.0.0.1/
2020-01-15T23:37:18.288+0000    0 document(s) imported successfully. 0 document(s) failed
      to import.
genres
2020-01-15T23:37:18.314+0000    connected to: mongodb://127.0.0.1/
2020-01-15T23:37:18.437+0000    11080 document(s) imported successfully. 0 document(s)
     failed to import.
jobs
2020-01-15T23:37:18.463+0000    connected to: mongodb://127.0.0.1/
2020-01-15T23:37:18.463+0000    0 document(s) imported successfully. 0 document(s) failed
      to import.
keywords2
2020-01-15T23:37:18.494+0000    connected to: mongodb://127.0.0.1/
2020-01-15T23:37:20.295+0000    158680 document(s) imported successfully. 0 document(s)
     failed to import.
languages
2020-01-15T23:37:20.322+0000    connected to: mongodb://127.0.0.1/
2020-01-15T23:37:20.942+0000    53300 document(s) imported successfully. 0 document(s)
     failed to import.
movies2
2020-01-15T23:37:20.970+0000    connected to: mongodb://127.0.0.1/
2020-01-15T23:37:23.128+0000    45566 document(s) imported successfully. 0 document(s)
     failed to import.
movies_collection
2020-01-15T23:37:23.156+0000    connected to: mongodb://127.0.0.1/
2020-01-15T23:37:23.356+0000    17960 document(s) imported successfully. 0 document(s)
     failed to import.
movies_companies
2020-01-15T23:37:23.384+0000    connected to: mongodb://127.0.0.1/
2020-01-15T23:37:24.139+0000    70545 document(s) imported successfully. 0 document(s)
     failed to import.
movies_countries
2020-01-15T23:37:24.167+0000    connected to: mongodb://127.0.0.1/
2020-01-15T23:37:24.724+0000    49423 document(s) imported successfully. 0 document(s)
     failed to import.
movies_genres
2020-01-15T23:37:24.753+0000    connected to: mongodb://127.0.0.1/
2020-01-15T23:37:24.880+0000    11080 document(s) imported successfully. 0 document(s)
     failed to import.
movies_keywords
2020-01-15T23:37:24.904+0000    connected to: mongodb://127.0.0.1/
2020-01-15T23:37:26.552+0000    158680 document(s) imported successfully. 0 document(s)
     failed to import.
movies_languages
2020-01-15T23:37:26.580+0000    connected to: mongodb://127.0.0.1/
```

```
2020-01-15T23:37:27.141+0000    53300 document(s) imported successfully. 0 document(s)
     failed to import.
people2
2020-01-15T23:37:27.167+0000    connected to: mongodb://127.0.0.1/
2020-01-15T23:37:30.168+0000    [#######.................] movies.people2       14.4MB
     /45.4MB (31.7%)
2020-01-15T23:37:33.168+0000    [###############.........] movies.people2       29.1MB
     /45.4MB (64.2%)
2020-01-15T23:37:36.168+0000    [#######################.] movies.people2       44.2MB
     /45.4MB (97.4%)
2020-01-15T23:37:36.403+0000    [########################] movies.people2       45.4MB
     /45.4MB (100.0%)
2020-01-15T23:37:36.403+0000    562474 document(s) imported successfully. 0 document(s)
     failed to import.
production_companies2
2020-01-15T23:37:36.430+0000    connected to: mongodb://127.0.0.1/
2020-01-15T23:37:37.237+0000    70545 document(s) imported successfully. 0 document(s)
     failed to import.
ratings
2020-01-15T23:37:37.264+0000    connected to: mongodb://127.0.0.1/
2020-01-15T23:37:40.264+0000    [........................] movies.ratings       6.85MB
     /223MB (3.1%)
2020-01-15T23:37:43.264+0000    [#.......................] movies.ratings       13.3MB
     /223MB (6.0%)
2020-01-15T23:37:46.264+0000    [##......................] movies.ratings       20.6MB
     /223MB (9.2%)
2020-01-15T23:37:49.264+0000    [##......................] movies.ratings       27.5MB
     /223MB (12.3%)
2020-01-15T23:37:52.264+0000    [###.....................] movies.ratings       35.5MB
     /223MB (15.9%)
2020-01-15T23:37:55.264+0000    [####....................] movies.ratings       43.5MB
     /223MB (19.5%)
2020-01-15T23:37:58.264+0000    [#####...................] movies.ratings       51.4MB
     /223MB (23.0%)
2020-01-15T23:38:01.264+0000    [######..................] movies.ratings       59.1MB
     /223MB (26.5%)
2020-01-15T23:38:04.264+0000    [#######.................] movies.ratings       66.4MB
     /223MB (29.8%)
2020-01-15T23:38:07.264+0000    [#######.................] movies.ratings       74.2MB
     /223MB (33.3%)
2020-01-15T23:38:10.264+0000    [########................] movies.ratings       82.0MB
     /223MB (36.8%)
2020-01-15T23:38:13.264+0000    [#########...............] movies.ratings       89.3MB
     /223MB (40.1%)
2020-01-15T23:38:16.264+0000    [##########..............] movies.ratings       96.9MB
     /223MB (43.4%)
2020-01-15T23:38:19.264+0000    [###########.............] movies.ratings       105MB/223
     MB (46.9%)
2020-01-15T23:38:22.264+0000    [############............] movies.ratings       112MB/223
     MB (50.3%)
2020-01-15T23:38:25.264+0000    [############............] movies.ratings       119MB/223
     MB (53.5%)
2020-01-15T23:38:28.264+0000    [#############...........] movies.ratings       127MB/223
     MB (56.8%)
2020-01-15T23:38:31.264+0000    [##############..........] movies.ratings       134MB/223
     MB (60.1%)
2020-01-15T23:38:34.264+0000    [###############.........] movies.ratings       141MB/223
     MB (63.4%)
2020-01-15T23:38:37.264+0000    [###############.........] movies.ratings       148MB/223
     MB (66.6%)
2020-01-15T23:38:40.264+0000    [################........] movies.ratings       156MB/223
     MB (69.9%)
2020-01-15T23:38:43.264+0000    [#################.......] movies.ratings       163MB/223
     MB (73.2%)
2020-01-15T23:38:46.264+0000    [##################......] movies.ratings       171MB/223
     MB (76.5%)
2020-01-15T23:38:49.264+0000    [###################.....] movies.ratings       178MB/223
     MB (79.9%)
2020-01-15T23:38:52.264+0000    [####################....] movies.ratings       186MB/223
     MB (83.4%)
2020-01-15T23:38:55.264+0000    [####################....] movies.ratings       194MB/223
     MB (86.9%)
```

```
2020-01-15T23:38:58.264+0000    [####################...] movies.ratings       201MB/223
     MB (90.2%)
2020-01-15T23:39:01.266+0000    [#####################..] movies.ratings       209MB/223
     MB (93.7%)
2020-01-15T23:39:04.264+0000    [######################.] movies.ratings       216MB/223
     MB (97.1%)
2020-01-15T23:39:06.883+0000    [#######################] movies.ratings       223MB/223
     MB (100.0%)
2020-01-15T23:39:06.883+0000    7448798 document(s) imported successfully. 0 document(s)
     failed to import.
```

Apache Spark import

Apache Spark is capable of reading directly a CSV file, in order to interface with Apache Spark we used the PySpark Spark Python API.
We developed a custom Python script to automatically import all CSV files in a folder and to correctly assign table names based on each CSV filename.

```python
import pyspark
from os import listdir
from os.path import isfile, join

print("Connecting to spark")
spark = pyspark.sql.SparkSession.builder.master("local").appName("Film").
     enableHiveSupport().getOrCreate()
sc = spark.sparkContext

print("Obtaining files")
path = "./"
files = [f for f in listdir(path) if isfile(join(path, f))]
tables = {}
for f in files:
    name = f[9:len(f) - 4] #f[9:len(f) - 4] is to remove "exported" and ".csv" from the
         name
    print("Loading table", name)
    df = spark.read.option("header", "true").csv(join(path, f))
    tables[name] = df
    df.registerTempTable(name)
```

The import phase for Apache Spark was completed within 10 seconds:

```
Connecting to spark
20/01/16 16:04:30 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
20/01/16 16:04:31 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Obtaining files
Loading table movies
Completed in 3.68s
Loading table keywords
Completed in 0.37s
Loading table movies_languages
Completed in 0.31s
Loading table countries
```

```
Completed in 0.27s
Loading table jobs
Completed in 0.32s
Loading table languages
Completed in 0.24s
Loading table movies_genres
Completed in 0.22s
Loading table movies_companies
Completed in 0.24s
Loading table collections
Completed in 0.30s
Loading table ratings
Completed in 0.21s
Loading table movies_keywords
Completed in 0.26s
Loading table movies_countries
Completed in 0.25s
Loading table departments
Completed in 0.19s
Loading table characters
Completed in 0.20s
Loading table production_companies
Completed in 0.25s
Loading table movies_collection
Completed in 0.20s
Loading table people
Completed in 0.21s
Loading table genres
Completed in 0.23s
```

5 Data processing

The assignment requires four different queries to be written:

i): The actor who acted in the most movies

ii): For each year, retrieve the best rated movie

iii): For each year, the best rated genre, the most revenued genre, and the best rated movie that revenued the most.

iv): For each year, the ranking of the top 10 european countries for movie revenues

The results of all but the last assignment can be rendered in a single table.
Since both Apache Spark and Oracle MySQL are relational databases it is not possible to get all the results of the last assignment in a single query (since a query needs to return a single table); therefore the last assignment has been implemented as one query for each year. This limitation is not present in MongoDB and, by tackling the problem with a NoSQL approach, we were able to fit the last assignment in a single aggregation pipeline.

Oracle MySQL data processing

Following there are our Oracle MySQL queries, with query results and the running time for each query.

Query 1: The actor who acted in the most movies

```
SELECT NAME FROM people WHERE ID = (
        SELECT PEOPLE_ID FROM (
                SELECT COUNT(*) AS NUM, PEOPLE_ID FROM characters_
                GROUP BY PEOPLE_ID, MOVIE_ID ORDER BY NUM DESC LIMIT 1
        ) AS alias_table
);
```

```
+----------------+
| NAME           |
+----------------+
| Aubree Miller |
+----------------+
1 row in set (3.376 sec)
```

Query 2: For each year, retrieve the best rated movie

```
SELECT TITLE, MAX(AVERAGE), YEAR(RELEASE_DATE) FROM (
SELECT
TITLE,
AVG(RATING) AS AVERAGE,
RELEASE_DATE
FROM movies
JOIN ratings ON ratings.MOVIE_ID = movies.ID
GROUP BY MOVIE_ID
) AS alias_table GROUP BY YEAR(RELEASE_DATE)
ORDER BY RELEASE_DATE;
```

```
+--------------------------------------------+--------------+--------------------+
| TITLE                                      | MAX(AVERAGE) | YEAR(RELEASE_DATE) |
+--------------------------------------------+--------------+--------------------+
| WarStoriesOurMotherNeverToldUs             |       4.1896 |                  0 |
| WorkersLeavingtheLumireFactory             |       3.5556 |               1895 |
| SerpentineDance:LoeFuller                  |       3.8356 |               1897 |
| TheDevilinaConvent                         |       3.5714 |               1899 |
| TheGhostTrain                              |       3.0000 |               1901 |
| TheInfernalCakewalk                        |       4.0812 |               1903 |
| TheMermaid                                 |       3.6957 |               1904 |
| TheHilariousPosters                        |       3.4000 |               1906 |
| TheRedSpectre                              |       3.8571 |               1907 |
| Lespapillonsjaponais                       |       3.9091 |               1908 |
| ThoseAwfulHats                             |       3.9286 |               1909 |
| WinsorMcCaytheFamousCartoonistoftheN.Y.    |       3.7692 |               1911 |
| TheGardener                                |       3.5714 |               1912 |
| GertietheDinosaur                              |     3.8000 |            1914 |
| TheBirthofaNation                              |     3.8976 |            1915 |
| Intolerance:LovesStruggleThroughouttheAges     |     4.1692 |            1916 |
| TheImmigrant                                   |     3.7000 |            1917 |
| ADogsLife                                      |     3.7778 |            1918 |
| MaleandFemale                                  |     3.8298 |            1919 |
| TheSaphead                                     |     3.9567 |            1920 |
| TheKid                                         |     4.0000 |            1921 |
| Nosferatu                                      |     4.1316 |            1922 |
| ThreeAges                                      |     3.8310 |            1923 |
| TheLastLaugh                                   |     4.0000 |            1924 |
| ThePleasureGarden                              |     4.3333 |            1925 |
| TheScarletLetter                               |     4.0253 |            1926 |
| Wings                                          |     4.2800 |            1927 |
| SteamboatWillie                                |     4.2727 |            1928 |
| TheManxman                                     |     4.0625 |            1929 |
| AllQuietontheWesternFront                      |     4.1467 |            1930 |
| M                                              |     4.1429 |            1931 |
| AFarewelltoArms                                |     4.2349 |            1932 |
| Liebelei                                       |     4.3235 |            1933 |
| ItHappenedOneNight                             |     4.1860 |            1934 |
| TopHat                                         |     4.1111 |            1935 |
| MyManGodfrey                                   |     4.2216 |            1936 |
| SnowWhiteandtheSevenDwarfs                     |     4.0839 |            1937 |
| TheAdventuresofRobinHood                       |     4.1412 |            1938 |
| TheLittlePrincess                              |     4.2020 |            1939 |
| Pinocchio                                      |     4.2593 |            1940 |
| TheMalteseFalcon                               |     4.2553 |            1941 |
| Casablanca                                     |     4.1429 |            1942 |
| TheManfromDownUnder                            |     4.1111 |            1943 |
| Gaslight                                       |     4.1855 |            1944 |
| Spellbound                                     |     4.2532 |            1945 |
| Notorious                                      |     4.3559 |            1946 |
| GoldenEarrings                                 |     4.2604 |            1947 |
| ForceofEvil                                    |     4.2500 |            1948 |
| CrowsandSparrows                               |     4.2558 |            1949 |
| Kim                                            |     4.3289 |            1950 |
| AnAmericaninParis                              |     4.2343 |            1951 |
| SinginintheRain                                |     4.5000 |            1952 |
| Wife                                           |     4.3197 |            1953 |
| RearWindow                                     |     4.3333 |            1954 |
| PatherPanchali                                 |     4.2082 |            1955 |
| DeathintheGarden                               |     4.3395 |            1956 |
| FunnyFace                                      |     4.2745 |            1957 |
| TheBalladofNarayama                            |     4.3231 |            1958 |
| TheWorldofApu                                  |     5.0000 |            1959 |
| PurpleNoon                                     |     4.2333 |            1960 |
| BreakfastatTiffanys                            |     4.1004 |            1961 |
| LawrenceofArabia                              |       4.4000 |                1962 |
| Charade                                       |       4.5000 |                1963 |
| TheUmbrellasofCherbourg                       |       4.1707 |                1964 |
| FasterPussycat!Kill!Kill!                     |       5.0000 |                1965 |
| TheGoodtheBadandtheUgly                       |       4.3034 |                1966 |
| BelledeJour                                   |       4.3685 |                1967 |
| Barbarella                                    |       4.2642 |                1968 |
| TheWildBunch                                  |       4.3132 |                1969 |
| TheAristocats                                 |       4.2711 |                1970 |
| BedknobsandBroomsticks                        |       4.3333 |                1971 |
| TheGodfather                                  |       4.4128 |                1972 |
| ToukiBouki                                    |       4.2546 |                1973 |
| BreadandChocolate                             |       4.2529 |                1974 |
| SwitchbladeSisters                            |       4.3030 |                1975 |
| TaxiDriver                                    |       4.3333 |                1976 |
| StarWars                                      |       4.3333 |                1977 |
| UpinSmoke                                     |       4.2708 |                1978 |
| Lassoci                                       |       5.0000 |                1979 |
| Windows                                       |       4.3572 |                1980 |
| HeavyMetal                                    |       5.0000 |                1981 |
| BladeRunner                                   |       4.3803 |                1982 |
| ReturnoftheJedi                               |       4.5000 |                1983 |
| Amadeus                                       |       4.5000 |                1984 |
| Harem                                         |       4.3135 |                1985 |
| Platoon                                       |       4.2888 |                1986 |
| LifeisRosy                                    |       4.2950 |                1987 |
| Oliver&Company                                |       5.0000 |                1988 |
| Batman                                        |       4.2625 |                1989 |
| HomeAlone                                     |       4.3505 |                1990 |
| Terminator2:JudgmentDay                       |       5.0000 |                1991 |
| TheBoysofSt.Vincent                           |       4.2612 |                1992 |
| Love&HumanRemains                             |       4.6429 |                1993 |
| GuardianAngel                                 |       5.0000 |                1994 |
| ToyStory                                      |       4.4443 |                1995 |
| WingsofCourage                                |       4.5000 |                1996 |
| KidsoftheRoundTable                           |       4.2676 |                1997 |
| TalkofAngels                                  |       5.0000 |                1998 |
| MenofMeans                                    |       4.3572 |                1999 |
| TheYards                                      |       4.5000 |                2000 |
| Songcatcher                                   |       4.2977 |                2001 |
| TwoFriends                                    |       5.0000 |                2002 |
| 30YEARSTOLIFE                                 |       5.0000 |                2003 |
| SpookyHouse                                   |       4.3107 |                2004 |
| ShakaZulu:TheLastGreatWarrior                 |       4.3333 |                2005 |
| Mybestenemy                                   |       4.2596 |                2006 |
| GeorgeLopez:AmericasMexican                   |       4.6667 |                2007 |
| ForeignExchange                               |       4.3333 |                2008 |
| MyRainyDays                                   |       4.3913 |                2009 |
| Venice                                        |       5.0000 |               2010 |
| KingdomCome                                   |       4.5385 |               2011 |
| TheFarmersWife                                |       4.4000 |               2012 |
| TheSleepover                                  |       5.0000 |               2013 |
| America:ImaginetheWorldWithoutHer             |       4.5000 |               2014 |
| Chappie                                       |       4.5000 |               2015 |
| BenHur                                        |       5.0000 |               2016 |
| PiratesoftheCaribbean:DeadMenTellNoTales      |       4.5000 |               2017 |
| Avatar2                                       |       3.5778 |               2020 |
+-----------------------------------------------+--------------+--------------------+
118 rows in set (12.018 sec)
```

Query 3: For each year, the best rated genre, the most revenued genre, and the best rated movie that revenued the most

```
SELECT A.YEAR, A.NAME, B.NAME, C.TITLE
FROM
(
SELECT NAME, MAX(AVERAGE), YEAR(RELEASE_DATE) AS YEAR FROM (
        SELECT
                genres.NAME,
                AVG(RATING) AS AVERAGE,
                RELEASE_DATE
        FROM movies
        JOIN ratings ON ratings.MOVIE_ID = movies.ID
        JOIN movies_genres ON movies_genres.ID_MOVIE = movies.ID
        JOIN genres ON movies_genres.ID_GENRE = genres.ID
        GROUP BY genres.ID
) AS alias_table1 GROUP BY YEAR(RELEASE_DATE)
ORDER BY RELEASE_DATE ) as A

JOIN
        (

        SELECT NAME, MAX(TOT_REVENUE), YEAR(RELEASE_DATE) AS YEAR FROM (
                SELECT
                        genres.NAME,
                        SUM(REVENUE) AS TOT_REVENUE,
                        RELEASE_DATE
                FROM movies
                JOIN ratings ON ratings.MOVIE_ID = movies.ID
                JOIN movies_genres ON movies_genres.ID_MOVIE = movies.ID
                JOIN genres ON movies_genres.ID_GENRE = genres.ID
                GROUP BY genres.ID
        ) as alias_table3 GROUP BY YEAR(RELEASE_DATE)
        ORDER BY RELEASE_DATE

) as B ON  A.YEAR=B.YEAR
JOIN



(

        SELECT TITLE, AVERAGE, MAX(REVENUE), YEAR(RELEASE_DATE) AS YEAR
                                             FROM (
                SELECT
                        genres.NAME,
                        AVG(RATING) AS AVERAGE,
                        RELEASE_DATE,
                        REVENUE,
                        TITLE
                FROM movies
                JOIN ratings ON ratings.MOVIE_ID = movies.ID
                JOIN movies_genres ON movies_genres.ID_MOVIE = movies.ID
                JOIN genres ON movies_genres.ID_GENRE = genres.ID
                GROUP BY movies.ID
        ) as alias_table15 GROUP BY YEAR(RELEASE_DATE), AVERAGE
        ORDER BY RELEASE_DATE

)  as C ON A.YEAR = C.YEAR;
```

```
+------+-----------+-----------+--------------------------------------------------+
| YEAR | NAME      | NAME      | TITLE                                            |
+------+-----------+-----------+--------------------------------------------------+
| 1976 | Crime     | Crime     | EatenAlive                                       |
| 1976 | Crime     | Crime     | TheShaggyD.A.                                    |
| 1976 | Crime     | Crime     | SilverStreak                                     |
| 1976 | Crime     | Crime     | TheLastTycoon                                    |
| 1976 | Crime     | Crime     | GodToldMeTo                                      |
| 1976 | Crime     | Crime     | TheCalloftheWild                                 |
| 1976 | Crime     | Crime     | ThatsEntertainmentPartII                         |
| 1976 | Crime     | Crime     | FamilyPlot                                       |
| 1976 | Crime     | Crime     | TaxiDriver                                       |
| 1985 | Crime     | Crime     | PoliceStory                                      |
| 1985 | Crime     | Crime     | ARoomwithaView                                   |
| 1985 | Crime     | Crime     | TheUnknownSoldier                                |
| 1985 | Crime     | Crime     | Demons                                           |
| 1985 | Crime     | Crime     | AmericanFlyers                                   |
| 1985 | Crime     | Crime     | MyScienceProject                                 |
| 1985 | Crime     | Crime     | Cocoon                                           |
| 1985 | Crime     | Crime     | St.ElmosFire                                     |
| 1985 | Crime     | Crime     | AViewtoaKill                                     |
| 1985 | Crime     | Crime     | Mishima:ALifeinFourChapters                      |
| 1985 | Crime     | Crime     | Fridaythe13th:ANewBeginning                      |
| 1985 | Crime     | Crime     | VisionQuest                                      |
| 1985 | Crime     | Crime     | Witness                                          |
| 1985 | Crime     | Crime     | YesMadam                                         |
| 1990 | Drama     | Drama     | Hamlet                                           |
| 1990 | Drama     | Drama     | DaysofBeingWild                                  |
| 1990 | Drama     | Drama     | Cadence                                          |
| 1990 | Drama     | Drama     | HomeAlone                                        |
| 1990 | Drama     | Drama     | ChildsPlay2                                      |
| 1990 | Drama     | Drama     | DanceswithWolves                                 |
| 1990 | Drama     | Drama     | DeathinBrunswick                                 |
| 1990 | Drama     | Drama     | GraveyardShift                                   |
| 1990 | Drama     | Drama     | WhitePalace                                      |
| 1990 | Drama     | Drama     | TheChallengers                                   |
| 1990 | Drama     | Drama     | TheExorcistIII                                   |
| 1990 | Drama     | Drama     | MyBlueHeaven                                     |
| 1990 | Drama     | Drama     | TakingCareofBusiness                             |
| 1990 | Drama     | Drama     | Metropolitan                                     |
| 1990 | Drama     | Drama     | DickTracy                                        |
| 1990 | Drama     | Drama     | SpacedInvaders                                    |
| 1990 | Drama     | Drama     | ILoveYoutoDeath                                   |
| 1990 | Drama     | Drama     | PrettyWoman                                       |
| 1990 | Drama     | Drama     | Halfaouine:BoyoftheTerraces                       |
| 1993 | Drama     | Drama     | Shadowlands                                       |
| 1993 | Drama     | Drama     | SixDegreesofSeparation                            |
| 1993 | Drama     | Drama     | LoveCheat&Steal                                   |
| 1993 | Drama     | Drama     | NakedinNewYork                                    |
| 1993 | Drama     | Drama     | Kika                                              |
| 1993 | Drama     | Drama     | TheBeverlyHillbillies                             |
| 1993 | Drama     | Drama     | TheNostradamusKid                                 |
| 1993 | Drama     | Drama     | Mr.Jones                                          |
| 1993 | Drama     | Drama     | DemolitionMan                                     |
| 1993 | Drama     | Drama     | Sirens                                            |
| 1993 | Drama     | Drama     | Germinal                                          |
| 1993 | Drama     | Drama     | TheSlingshot                                      |
| 1993 | Drama     | Drama     | DazedandConfused                                  |
| 1993 | Drama     | Drama     | TrueRomance                                       |
| 1993 | Drama     | Drama     | ShortCuts                                         |
| 1993 | Drama     | Drama     | KingoftheHill                                     |
| 1993 | Drama     | Drama     | TheSecretGarden                                   |
| 1993 | Drama     | Drama     | SearchingforBobbyFischer                          |
| 1993 | Drama     | Drama     | SoIMarriedanAxeMurderer                           |
| 1993 | Drama     | Drama     | RisingSun                                         |
| 1993 | Drama     | Drama     | Coneheads                                         |
| 1993 | Drama     | Drama     | AnotherStakeout                                   |
| 1993 | Drama     | Drama     | WeekendatBerniesII                                |
| 1993 | Drama     | Drama     | GuiltyasSin                                       |
| 1993 | Drama     | Drama     | LifeWithMikey                                     |
| 1993 | Drama     | Drama     | TheEyeofVichy                                     |
| 1993 | Drama     | Drama     | Love&HumanRemains                                 |
| 1993 | Drama     | Drama     | MuchAdoAboutNothing                               |
| 1993 | Drama     | Drama     | BoilingPoint                                      |
| 1993 | Drama     | Drama     | BodyofEvidence                                    |
| 1994 | Adventure | Adventure | MixedNuts                                         |
| 1994 | Adventure | Adventure | LittleWomen                                       |
| 1994 | Adventure | Adventure | DropZone                                          |
| 1994 | Adventure | Adventure | Shopping                                          |
| 1994 | Adventure | Adventure | ThePagemaster                                     |
| 1994 | Adventure | Adventure | Junior                                            |
| 1994 | Adventure | Adventure | Miracleon34thStreet                               |
| 1994 | Adventure | Adventure | TheSwanPrincess                                   |
| 1994 | Adventure | Adventure | BacktoBackFacetoFace                              |
| 1994 | Adventure | Adventure | InterviewwiththeVampire                           |
| 1994 | Adventure | Adventure | NobodyLovesMe                                     |
| 1994 | Adventure | Adventure | CountryLife                                       |
| 1994 | Adventure | Adventure | RedFirecrackerGreenFirecracker                    |
| 1994 | Adventure | Adventure | BulletsOverBroadway                               |
| 1994 | Adventure | Adventure | ThePostman                                         |
| 1994 | Adventure | Adventure | TheGlassShield                                     |
| 1994 | Adventure | Adventure | Vanyaon42ndStreet                                  |
| 1994 | Adventure | Adventure | HeavenlyCreatures                                  |
| 1994 | Adventure | Adventure | ASimpleTwistofFate                                 |
| 1994 | Adventure | Adventure | CampNowhere                                        |
| 1994 | Adventure | Adventure | NaturalBornKillers                                 |
| 1994 | Adventure | Adventure | Fresh                                              |
| 1994 | Adventure | Adventure | Amateur                                            |
| 1994 | Adventure | Adventure | EatDrinkManWoman                                   |
| 1994 | Adventure | Adventure | AngelsintheOutfield                                |
| 1994 | Adventure | Adventure | AGoodManinAfrica                                   |
| 1994 | Adventure | Adventure | ChungkingExpress                                   |
| 1994 | Adventure | Adventure | LittleBigLeague                                    |
| 1994 | Adventure | Adventure | GoFish                                             |
| 1994 | Adventure | Adventure | Oblivion                                           |
| 1994 | Adventure | Adventure | Maverick                                           |
| 1994 | Adventure | Adventure | AnUnforgettableSummer                              |
| 1994 | Adventure | Adventure | Crooklyn                                           |
| 1994 | Adventure | Adventure | EvenCowgirlsGettheBlues                            |
| 1994 | Adventure | Adventure | OfLoveandShadows                                   |
| 1994 | Adventure | Adventure | CleanSlate                                         |
| 1994 | Adventure | Adventure | BeingHuman                                         |
| 1994 | Adventure | Adventure | WhenaManLovesaWoman                                |
| 1994 | Adventure | Adventure | Chasers                                            |
| 1994 | Adventure | Adventure | SomeFolksCallItaSlingBlade                         |
| 1994 | Adventure | Adventure | BhajiontheBeach                                    |
| 1994 | Adventure | Adventure | DestinyinSpace                                     |
| 1994 | Adventure | Adventure | CabinBoy                                           |
| 1994 | Adventure | Adventure | NothingtoLose                                      |
| 1994 | Adventure | Adventure | Tigrero:AFilmThatWasNeverMade                      |
| 1994 | Adventure | Adventure | Priest                                             |
| 1995 | Comedy    | Comedy    | Mr.HollandsOpus                                    |
| 1995 | Comedy    | Comedy    | HeavyWeather                                       |
| 1995 | Comedy    | Comedy    | Dracula:DeadandLovingIt                            |
| 1995 | Comedy    | Comedy    | TheDeathmaker                                      |
| 1995 | Comedy    | Comedy    | Casino                                             |
| 1995 | Comedy    | Comedy    | NickofTime                                         |
| 1995 | Comedy    | Comedy    | Reckless                                           |
| 1995 | Comedy    | Comedy    | TheAmericanPresident                               |
| 1995 | Comedy    | Comedy    | TheCrossingGuard                                   |
| 1995 | Comedy    | Comedy    | TotalEclipse                                       |
| 1995 | Comedy    | Comedy    | LeavingLasVegas                                    |
| 1995 | Comedy    | Comedy    | Powder                                             |
| 1995 | Comedy    | Comedy    | Copycat                                            |
| 1995 | Comedy    | Comedy    | ThreeWishes                                        |
| 1995 | Comedy    | Comedy    | GirlintheCadillac                                  |
| 1995 | Comedy    | Comedy    | GetShorty                                          |
| 1995 | Comedy    | Comedy    | TheBabysitter                                     |
| 1995 | Comedy    | Comedy    | StrangeDays                                       |
| 1995 | Comedy    | Comedy    | TheScarletLetter                                  |
| 1995 | Comedy    | Comedy    | FeastofJuly                                       |
| 1995 | Comedy    | Comedy    | VirginMary                                        |
| 1995 | Comedy    | Comedy    | Stonewall                                         |
| 1995 | Comedy    | Comedy    | TheRunoftheCountry                                |
| 1995 | Comedy    | Comedy    | TheStarMaker                                      |
| 1995 | Comedy    | Comedy    | BlueintheFace                                     |
| 1995 | Comedy    | Comedy    | Clockers                                          |
| 1995 | Comedy    | Comedy    | Hackers                                           |
| 1995 | Comedy    | Comedy    | TheJourneyofAugustKing                            |
| 1995 | Comedy    | Comedy    | Roula                                             |
| 1995 | Comedy    | Comedy    | AngelsandInsects                                  |
| 1995 | Comedy    | Comedy    | Rude                                              |
| 1995 | Comedy    | Comedy    | Screamers                                         |
| 1995 | Comedy    | Comedy    | ToWongFooThanksforEverything!JulieNewmar          |
| 1995 | Comedy    | Comedy    | Dadetown                                          |
| 1995 | Comedy    | Comedy    | HeartsandMinds                                    |
| 1995 | Comedy    | Comedy    | Desperado                                         |
| 1995 | Comedy    | Comedy    | Jeffrey                                           |
| 1995 | Comedy    | Comedy    | AKidinKingArthursCourt                            |
| 1995 | Comedy    | Comedy    | CastleFreak                                       |
| 1995 | Comedy    | Comedy    | SomethingtoTalkAbout                              |
| 1995 | Comedy    | Comedy    | Target                                            |
| 1995 | Comedy    | Comedy    | InstituteBenjamentaorThisDreamPeopleCallHumanLife |
| 1995 | Comedy    | Comedy    | AnAwfullyBigAdventure                             |
| 1995 | Comedy    | Comedy    | FirstKnight                                       |
| 1995 | Comedy    | Comedy    | JudgeDredd                                        |
| 1995 | Comedy    | Comedy    | DeltaofVenus                                      |
| 1995 | Comedy    | Comedy    | CanadianBacon                                     |
| 1995 | Comedy    | Comedy    | Fluke                                             |
| 1995 | Comedy    | Comedy    | LaHaine                                           |
| 1995 | Comedy    | Comedy    | JohnnyMnemonic                                    |
| 1995 | Comedy    | Comedy    | Braveheart                                        |
| 1995 | Comedy    | Comedy    | HeadlessBodyinToplessBar                          |
| 1995 | Comedy    | Comedy    | DieHard:WithaVengeance                            |
| 1995 | Comedy    | Comedy    | Gordy                                             |
| 1995 | Comedy    | Comedy    | ALittlePrincess                                   |
| 1995 | Comedy    | Comedy    | MidaqAlley                                        |
| 1995 | Comedy    | Comedy    | PictureBride                                      |
| 1995 | Comedy    | Comedy    | ShanghaiTriad                                     |
| 1995 | Comedy    | Comedy    | TheUnderneath                                     |
| 1995 | Comedy    | Comedy    | DestinyTurnsontheRadio                            |
| 1995 | Comedy    | Comedy    | TheCure                                           |
| 1995 | Comedy    | Comedy    | KissofDeath                                       |
| 1995 | Comedy    | Comedy    | TheBasketballDiaries                              |
| 1995 | Comedy    | Comedy    | NewJerseyDrive                                    |
| 1995 | Comedy     | Comedy     | JuryDuty                                           |
| 1995 | Comedy     | Comedy     | TommyBoy                                           |
| 1995 | Comedy     | Comedy     | TallTale                                           |
| 1995 | Comedy     | Comedy     | LosingIsaiah                                       |
| 1995 | Comedy     | Comedy     | TheLandBeforeTimeIII:TheTimeoftheGreatGiving       |
| 1995 | Comedy     | Comedy     | AllThingsFair                                      |
| 1995 | Comedy     | Comedy     | FederalHill                                        |
| 1995 | Comedy     | Comedy     | TheWalkingDead                                     |
| 1995 | Comedy     | Comedy     | TheBradyBunchMovie                                 |
| 1995 | Comedy     | Comedy     | ToughandDeadly                                     |
| 1995 | Comedy     | Comedy     | BillyMadison                                       |
| 1995 | Comedy     | Comedy     | MadagascarSkin                                     |
| 1995 | Comedy     | Comedy     | MurderintheFirst                                   |
| 1995 | Comedy     | Comedy     | LivinginOblivion                                   |
| 1995 | Comedy     | Comedy     | TheFear                                            |
| 1995 | Comedy     | Comedy     | TheAddiction                                       |
| 1995 | Comedy     | Comedy     | TheWhiteBalloon                                    |
| 1995 | Comedy     | Comedy     | RentaKid                                           |
| 1995 | Comedy     | Comedy     | TokyoFist                                          |
| 1996 | Comedy     | Comedy     | Hamlet                                             |
| 1996 | Comedy     | Comedy     | Michael                                            |
| 1996 | Comedy     | Comedy     | ImNotRappaport                                     |
| 1996 | Comedy     | Comedy     | EscapeClause                                       |
| 1996 | Comedy     | Comedy     | Evita                                              |
| 1996 | Comedy     | Comedy     | SpaceJam                                           |
| 1996 | Comedy     | Comedy     | Twisted                                            |
| 1996 | Comedy     | Comedy     | BreathingRoom                                      |
| 1996 | Comedy     | Comedy     | MadDogTime                                         |
| 1996 | Comedy     | Comedy     | SantawithMuscles                                   |
| 1996 | Comedy     | Comedy     | Ransom                                             |
| 1996 | Comedy     | Comedy     | HighSchoolHigh                                     |
| 1996 | Comedy     | Comedy     | Solo                                               |
| 1996 | Comedy     | Comedy     | ThePortraitofaLady                                 |
| 1996 | Comedy     | Comedy     | inf                                                |
| 1996 | Comedy     | Comedy     | MURDERandmurder                                    |
| 1996 | Comedy     | Comedy     | Bulletproof                                        |
| 1996 | Comedy     | Comedy     | TheCrow:CityofAngels                               |
| 1996 | Comedy     | Comedy     | AVeryBradySequel                                   |
| 1996 | Comedy     | Comedy     | Foxfire                                            |
| 1996 | Comedy     | Comedy     | KansasCity                                         |
| 1996 | Comedy     | Comedy     | TinCup                                             |
| 1996 | Comedy     | Comedy     | ChainReaction                                      |
| 1996 | Comedy     | Comedy     | Crimetime                                          |
| 1996 | Comedy     | Comedy     | TheAdventuresofPinocchio                           |
| 1996 | Comedy     | Comedy     | LoversKnot                                         |
| 1996 | Comedy     | Comedy     | Illtown                                            |
| 1996 | Comedy     | Comedy     | Kingpin                                            |
| 1996 | Comedy     | Comedy     | Eraser                                             |
| 1996 | Comedy     | Comedy     | TheHunchbackofNotreDame                              |
| 1996 | Comedy     | Comedy     | TheCableGuy                                          |
| 1996 | Comedy     | Comedy     | Eddie                                                |
| 1996 | Comedy     | Comedy     | Flipper                                              |
| 1996 | Comedy     | Comedy     | Power98                                              |
| 1996 | Comedy     | Comedy     | ThePallbearer                                        |
| 1996 | Comedy     | Comedy     | TheGreatWhiteHype                                    |
| 1996 | Comedy     | Comedy     | TheCraft                                             |
| 1996 | Comedy     | Comedy     | TheHauntedWorldofEdwardD.WoodJr.                     |
| 1996 | Comedy     | Comedy     | SunsetPark                                           |
| 1996 | Comedy     | Comedy     | TheQuest                                             |
| 1996 | Comedy     | Comedy     | MysteryScienceTheater3000:TheMovie                   |
| 1996 | Comedy     | Comedy     | TheSubstitute                                        |
| 1996 | Comedy     | Comedy     | Loaded                                               |
| 1996 | Comedy     | Comedy     | AFamilyThing                                         |
| 1996 | Comedy     | Comedy     | ItsMyParty                                           |
| 1996 | Comedy     | Comedy     | Frisk                                                |
| 1996 | Comedy     | Comedy     | TrueCrime                                            |
| 1996 | Comedy     | Comedy     | Hellraiser:Bloodline                                 |
| 1996 | Comedy     | Comedy     | HomewardBoundII:LostinSanFrancisco                   |
| 1996 | Comedy     | Comedy     | BloodsportII                                         |
| 1996 | Comedy     | Comedy     | MuppetTreasureIsland                                 |
| 1996 | Comedy     | Comedy     | MessagetoLove:TheIsleofWightFestival                 |
| 1996 | Comedy     | Comedy     | BlackSheep                                           |
| 1996 | Comedy     | Comedy     | OnceUponaTime...WhenWeWereColored                    |
| 1996 | Comedy     | Comedy     | Manny&Lo                                             |
| 1996 | Comedy     | Comedy     | DunstonChecksIn                                      |
| 1996 | Comedy     | Comedy     | PreciousFind                                         |
| 1996 | Comedy     | Comedy     | IrisBlond                                            |
+------+-----------+-----------+----------------------------------------------------+
259 rows in set (4 hours 41 min 47.433 sec)
```

Query 4: For each year, the ranking of the top 10 european countries for movie revenues

Since the assignment results cannot be rendered in a single table it is not possible to design a single query that outputs all the required rankings.
This constrain is given by the fact that MySQL is a relational DBMS and no such query that fully satisfies the assignment can exist due to this constrain.
Our query yields the ranking for the required year, in order to obtain all the rankings a query for each year is required.

```
# The following query outputs a list of all distinct years in the dataset
SELECT DISTINCT YEAR(RELEASE_DATE) FROM movies;

# The following query outputs the required rankings for the specified year
SELECT sum(revenue) as tot_revenue, countries.iso_3166_1 as country_code FROM movies
JOIN movies_countries ON movies_countries.id_movie = movies.id
JOIN countries ON countries.id = movies_countries.id_movie
```

```
WHERE UPPER(countries.iso_3166_1) IN ('AL', 'AD', 'AM', 'AT', 'BY', 'BE', 'BA', 'BG', 'CH
    ', 'CY', 'CZ', 'DE', 'DK', 'EE', 'ES', 'FO', 'FI', 'FR', 'GB', 'GE', 'GI', 'GR', 'HU
    ', 'HR',
 'IE', 'IS', 'IT', 'LI', 'LT', 'LU', 'LV', 'MC', 'MK', 'MT', 'NO', 'NL', 'PL',
 'PT', 'RO', 'RU', 'SE', 'SI', 'SK', 'SM', 'TR', 'UA', 'VA')
AND YEAR(release_date) = 2010
group by country_code
ORDER BY tot_revenue DESC LIMIT 10;
```

```
Sample results for year 2010:

+-------------+--------------+
| tot_revenue | country_code |
+-------------+--------------+
|   755929529 | GB           |
|   562524570 | AT           |
|   461634780 | FR           |
|   433198123 | ES           |
|   410705777 | DE           |
|   400062763 | SE           |
|   366962627 | FI           |
|    35870055 | PL           |
|     4644108 | NL           |
|     1268793 | RU           |
+-------------+--------------+
10 rows in set (0.518 sec)
```

MongoDB data processing

For each MongoDB task we decided to implement a Multi-Stage MongoDB Aggregation Pipeline.
We deem the aggregation pipeline the best framework for these specific tasks since it allows us to split each task in multiple, less-complex, stages and build each query one stage at a time using a bottom-up approach.
Developing multiple stages individually also speeds up the debugging process since a single stage is very simple and powerful tools exist for debugging, such as MongoDB Compass.
We also created indexes on all the fields that required either a $*lookup* or a $*match* as to minimize query running time.
Due to the large amount of output, and the more verbose output structure of MongoDB, the output of some queries has beed redacted.

Query 1: The actor who acted in the most movies

```
db.getCollection('characters_2').aggregate([
{
    $group: {
        _id: {
            'people_id': "$'people_id'",
            'movie_id': "$'movie_id'"
        },
        count: {
            $sum: 1
        }
    }},
    { $sort: { count: -1} },
    { $limit: 1 },
    { $lookup: {
        from: 'people2',
        localField: "_id.people_id",
        foreignField: "'id'",
        as: 'character'
    }}


], {allowDiskUse: true})


/* 1 */
{
    "_id" : {
        "people_id" : 145252,
        "movie_id" : 211256
    },
    "count" : 6.0,
    "character" : [
        {
            "_id" : ObjectId("5e1fa239b2e7d195078f7674"),
            "'id'" : 145252,
            "'people_id'" : 19750,
            "'movie_id'" : 1884,
            "'credit_id'" : "52fe431dc3a36847f803b7c5",
            "'name'" : "Aubree Miller",
            "'gender'" : 0,
            "'profile_path'" : "/vaIvxK694mKfS6g006SqevkCyRM.jpg"
        }
    ]
}
```

Query 2: For each year, retrieve the best rated movie

```
db.getCollection('ratings').aggregate([
{
    $group: {
        _id: {
            'movie_id': "$'movie_id'"
        },
        rating: {
            $avg: "$'rating'"
        }
    }},
    {
        $lookup: {
            from: 'movies2',
            localField: "_id.movie_id",
            foreignField: "'id'",
            as: 'movie'
        }
    },
    { $match: { "movie.0": {$exists: 1} } },
    {
        $group: {
            _id: {
                    $year: { $toDate: { $arrayElemAt: [ "$movie.'release_date'", 0 ]}}
            },
            rating: {
                $max: '$rating'
            },
            allfilms: {
                $push: { title: "$movie.'original_title'", rating: '$rating'}
            }
        }
    },
    {
        $project: {
            _id: "$_id",
            name: { $filter: { input: "$allfilms", as: "rating", cond: { $gte: ["$rating"
                , "$$rating.rating"] } } }
        }
    },
    {
        $project: {
            _id: "$_id",
            name: { $arrayElemAt: [ "$name", 0 ] }
        }
    },
    {
        $project: {
            _id: "$_id",
            name: { $arrayElemAt: [ "$name.title", 0 ] }
        }
    }
], {allowDiskUse: true})
```

Query 3: For each year, the best rated genre, the most revenued genre, and the best rated movie that revenued the most

```
db.getCollection('ratings').aggregate([
    {
        $lookup: {
            from: 'movies2',
            localField: "'movie_id'",
            foreignField: "'id'",
            as: 'movie'
        }
    },{
        $lookup: {
            from: 'movies_genres',
            localField: "'movie_id'",
            foreignField: "'id'",
            as: 'genre'
        }
    },
    {
        $group: {
            _id: { $year: { $toDate: { $arrayElemAt: [ "$movie.'release_date'", 0 ]}} },
            rating: { $avg: "$'rating'"},
            allgenres: {
                $push: { genre_id: { $arrayElemAt: [ "$genre.'id_genre'", 0 ]}, rating: "
                    $'rating'"}
            }
        }
    },
    {
        $project: {
            _id: "$_id",
            name: { $filter: { input: "$allgenres", as: "rating", cond: { $gte: ["$rating
                ", "$$rating.rating"] } } }
        }
    },
    {
        $project: {
            _id: "$_id",
            name: { $arrayElemAt: [ "$name.genre_id", 0 ] }
        }
    },
    {
        $lookup: {
            from: 'genres',
            localField: "name",
            foreignField: "'id'",
            as: 'name'
        }
    },
    {
        $project: {
            _id: "$_id",
            name: { $arrayElemAt: [ "$name.'name'", 0 ] }
        }
    }
], {allowDiskUse: true})

db.getCollection('movies2').aggregate([
{
        $lookup: {
            from: 'movies_genres',
            localField: "'_id'",
            foreignField: "'id'",
            as: 'genre'
}},
{
    $project: {
        genre: "$genre.'id_genre'",
        revenue: "$'revenue'",
        release_date: "$'release_date'",
        date_length: { $strLenCP: { $ifNull: [{ $convert: { input: "$'release_date'", to:
             "string", onError: ""}}, ""]}},
    }
},
{$match: { date_length: { $gt: 2 }}},
{
        $group: {
            _id: { genre: {$arrayElemAt: ["$genre", 0]}, year: { $year: { $toDate:  "
                 $release_date"} }},
            revenue: { $avg: "$revenue" }
        }
    },
    {
        $group: {
            _id: "$_id.year",
            revenue: { $max: "$revenue" },
            allgenres: {
                $push: { genre_id: "$_id.genre", revenue: "$revenue"}
            }
        }
    },
    {
        $project: {
            _id: "$_id",
            name: { $filter: { input: "$allgenres", as: "revenue", cond: { $gte: ["
                 $revenue", "$$revenue.revenue"] } } }
        }
    },
    {
        $project: {
            _id: "$_id",
            name: { $arrayElemAt: ["$name.genre_id" , 0]}
        }
    },
    {
        $lookup: {
            from: 'genres',
            localField: "name",
            foreignField: "'id'",
            as: 'genre'
}},
{
        $project: {
            _id: "$_id",
            name: { $arrayElemAt: ["$genre.'name'" , 0]}
        }
    }
], {allowDiskUse: true})


db.getCollection('ratings').aggregate([
    {
        $lookup: {
            from: 'movies2',
            localField: "'movie_id'",
            foreignField: "'id'",
            as: 'movie'
        }
    },
    {
        $group: {
            _id: { movie: {$arrayElemAt: ["$movie.'original_title'", 0]}, year: { $year:
                { $dateFromString: { dateString: { $arrayElemAt: [ "$movie.'release_date
                '", 0 ]}, onError: null} }}},
            rating: { $avg: "$'rating'"},
            revenue: { $avg: {$arrayElemAt: ["$movie.'revenue'", 0]} }
        }
    },
    {
        $group: {
            _id: "$_id.year",
            allmovies: { $push: { rating: "$rating", revenue: "$revenue", title: "$_id.
                movie"} },
            best_rating: { $max: "$rating" }
        }
    },
    {
        $project: {
            _id: "$_id",
            bestRated: { $filter: { input: "$allmovies", as: "rating", cond: { $gte: ["
                $$rating.rating", "$best_rating"] } } }
        }
    },
    { $unwind: "$bestRated"},
    {
        $group: {
            _id: "$_id",
            allmovies: { $push: { rating: "$bestRated.rating", revenue: "$bestRated.
                revenue", title: "$bestRated.title"} },
            best_revenue: { $max: "$bestRated.revenue" }
        }
    },
    {
        $project: {
            _id: "$_id",
            bestFilm: { $filter: { input: "$allmovies", as: "rating", cond: { $gte: ["
                $$rating.revenue", "$best_revenue"] } } }
        }
    },
    {
        $project: {
            _id: "$_id",
            bestFilm: { $arrayElemAt: ["$bestFilm" , 0] }
        }
    }
], {allowDiskUse: true})
```

Query 4: For each year, the ranking of the top 10 european countries for movie revenues

```
db.getCollection('movies2').aggregate([
{
        $lookup: {
            from: 'movies_countries',
            localField: "'_id'",
            foreignField: "'id_movie'",
            as: 'country'
}},
{
    $project: {
        revenue: "$'revenue'",
        year: "$'release_date'",
        date_length: { $ifNull: [{ $convert: { input: "$'release_date'", to: "string",
             onError: ""}}, ""]},
        country: "$country.'id_country'"
    }
},
{
    $project: {
        revenue: "$revenue",
        year: "$year",
        date_length: { $strLenCP: "$date_length"},
        country: "$country"
    }
},
{$match: { date_length: { $gt: 2 }}},
{
    $project: {
        revenue: "$revenue",
        year: { $year: { $toDate: "$year" } },
        country: { $arrayElemAt: ["$country", 0] }
    }
},
{
    $group: {
        _id: {country: "$country", year: "$year"},
        total_revenue: { $sum: "$revenue" }
    }
},
{
    $match: {
        "_id.country": { $in: ['AL', 'AD', 'AM', 'AT', 'BY', 'BE', 'BA', 'BG', 'CH', 'CY'
             , 'CZ', 'DE', 'DK', 'EE', 'ES', 'FO', 'FI', 'FR', 'GB', 'GE', 'GI', 'GR', '
             HU', 'HR',
 'IE', 'IS', 'IT', 'LI', 'LT', 'LU', 'LV', 'MC', 'MK', 'MT', 'NO', 'NL', 'PL',
 'PT', 'RO', 'RU', 'SE', 'SI', 'SK', 'SM', 'TR', 'UA', 'VA']}
    }
},
{ $sort: { total_revenue: -1 }},
{
    $group: {
        _id: "$_id.year",
        ranking: { $push: { "country": "$_id.country", "revenue": "$total_revenue" } },
    }
},
{
    $project: {
        _id: "$_id",
        ranking: { $slice: ["$ranking", 10]},
    }
},
{ $sort: { _id: -1 }}
], {allowDiskUse: true})
```

Apache Spark data processing

Since Apache Spark is also a relational DBMS and provides an SQL interface through Spark SQL we are able to execute the same queries that we used for MySQL with minimal adjustments.
The following sections contain the SQL queries with the right syntax for Spark SQL, the

Python code used for running and timing the queries and actual query results and timings.

Query 1: The actor who acted in the most movies

```
SELECT name FROM people WHERE id = (
        SELECT people_id FROM (
                SELECT COUNT(*) AS NUM, people_id FROM characters
                GROUP BY people_id, movie_id ORDER BY NUM DESC LIMIT 1
        ) AS alias_table
)
```

Query 2: For each year, retrieve the best rated movie

```
SELECT first(title), MAX(average), YEAR(release_date) as release_date FROM (
        SELECT
        first(title) as title,
        AVG(rating) AS average,
        first(release_date) as release_date
        FROM movies
        JOIN ratings ON ratings.movie_id = movies.id
        GROUP BY movie_id
) AS alias_table GROUP BY YEAR(release_date)
ORDER BY release_date DESC
```

Query 3: For each year, the best rated genre, the most revenued genre, and the best rated movie that revenued the most

```
SELECT A.year, A.name, B.name, C.title
FROM
(
SELECT first(name) as name, MAX(average), YEAR(release_date) AS year FROM (
        SELECT
                first(genres.name) as name,
                AVG(rating) AS average,
                first(release_date) as release_date
        FROM movies
        JOIN ratings ON ratings.movie_id = movies.id
        JOIN movies_genres ON movies_genres.id_movie = movies.id
        JOIN genres ON movies_genres.id_genre = genres.id
        GROUP BY genres.id
) AS alias_table1 GROUP BY YEAR(release_date)
ORDER BY year ) as A

JOIN
        (

        SELECT first(name) as name, MAX(tot_revenue), YEAR(release_date) AS year FROM (
                SELECT
                        first(genres.name) as name,
                        SUM(revenue) AS tot_revenue,
                        first(release_date) as release_date
                FROM movies
                JOIN ratings ON ratings.movie_id = movies.id
                JOIN movies_genres ON movies_genres.id_movie = movies.id
                JOIN genres ON movies_genres.id_genre = genres.id
                GROUP BY genres.id
        ) as alias_table3 GROUP BY YEAR(release_date)
        ORDER BY year

) as B ON  A.year=B.year
JOIN
```

```
(

        SELECT first(title) as title, first(average) as average, MAX(revenue), YEAR(
            release_date) AS year                                  FROM (
                SELECT
                        first(genres.name) as name,
                        AVG(rating) AS average,
                        first(release_date) as release_date,
                        first(revenue) as revenue,
                        first(title) as title
                FROM movies
                JOIN ratings ON ratings.movie_id = movies.ID
                JOIN movies_genres ON movies_genres.id_movie = movies.ID
                JOIN genres ON movies_genres.id_genre = genres.ID
                GROUP BY movies.id
        ) as alias_table15 GROUP BY YEAR(release_date), average
        ORDER BY year

)  as C ON A.year = C.year
```

Query 4: For each year, the ranking of the top 10 european countries for movie revenues

```
SELECT DISTINCT YEAR(release_date) FROM movies

SELECT sum(revenue) as tot_revenue, countries.iso_3166_1 as country_code FROM movies
JOIN movies_countries ON movies_countries.id_movie = movies.id
JOIN countries ON countries.id = movies_countries.id_movie
WHERE UPPER(countries.iso_3166_1) IN ('AL', 'AD', 'AM', 'AT', 'BY', 'BE', 'BA', 'BG', 'CH
     ', 'CY', 'CZ', 'DE', 'DK', 'EE', 'ES', 'FO', 'FI', 'FR', 'GB', 'GE', 'GI', 'GR', 'HU
     ', 'HR',
 'IE', 'IS', 'IT', 'LI', 'LT', 'LU', 'LV', 'MC', 'MK', 'MT', 'NO', 'NL', 'PL',
 'PT', 'RO', 'RU', 'SE', 'SI', 'SK', 'SM', 'TR', 'UA', 'VA')
AND YEAR(release_date) = {}
group by country_code
ORDER BY tot_revenue DESC LIMIT 10
```

Python code

We have developed a custom Python script that imports the dataset from the CSV files, runs each SQL query and then output the results and timings.
Our script also automatically iterates over each year and this way it manages to overcome the limitations posed by the relational model in the fourth query.

```
import pyspark
from os import listdir
from os.path import isfile, join
import time



QUERY_1 = """
SELECT name FROM people WHERE id = (
        SELECT people_id FROM (
                SELECT COUNT(*) AS NUM, people_id FROM characters
                GROUP BY people_id, movie_id ORDER BY NUM DESC LIMIT 1
        ) AS alias_table
)
"""
```

```
QUERY_2 = """
SELECT first(title), MAX(average), YEAR(release_date) as release_date FROM (
        SELECT
        first(title) as title,
        AVG(rating) AS average,
        first(release_date) as release_date
        FROM movies
        JOIN ratings ON ratings.movie_id = movies.id
        GROUP BY movie_id
) AS alias_table GROUP BY YEAR(release_date)
ORDER BY release_date DESC
"""


QUERY_3 = """
SELECT A.year, A.name, B.name, C.title
FROM
(
SELECT first(name) as name, MAX(average), YEAR(release_date) AS year FROM (
        SELECT
                first(genres.name) as name,
                AVG(rating) AS average,
                first(release_date) as release_date
        FROM movies
        JOIN ratings ON ratings.movie_id = movies.id
        JOIN movies_genres ON movies_genres.id_movie = movies.id
        JOIN genres ON movies_genres.id_genre = genres.id
        GROUP BY genres.id
) AS alias_table1 GROUP BY YEAR(release_date)
ORDER BY year ) as A

JOIN
        (

        SELECT first(name) as name, MAX(tot_revenue), YEAR(release_date) AS year FROM (
                SELECT
                        first(genres.name) as name,
                        SUM(revenue) AS tot_revenue,
                        first(release_date) as release_date
                FROM movies
                JOIN ratings ON ratings.movie_id = movies.id
                JOIN movies_genres ON movies_genres.id_movie = movies.id
                JOIN genres ON movies_genres.id_genre = genres.id
                GROUP BY genres.id
        ) as alias_table3 GROUP BY YEAR(release_date)
        ORDER BY year

) as B ON  A.year=B.year
JOIN



(

        SELECT first(title) as title, first(average) as average, MAX(revenue), YEAR(
            release_date) AS year                                     FROM (
                SELECT
                        first(genres.name) as name,
                        AVG(rating) AS average,
                        first(release_date) as release_date,
                        first(revenue) as revenue,
                        first(title) as title
                FROM movies
                JOIN ratings ON ratings.movie_id = movies.ID
                JOIN movies_genres ON movies_genres.id_movie = movies.ID
                JOIN genres ON movies_genres.id_genre = genres.ID
                GROUP BY movies.id
        ) as alias_table15 GROUP BY YEAR(release_date), average
        ORDER BY year
)  as C ON A.year = C.year
"""


QUERY_4_1 = """
SELECT DISTINCT YEAR(release_date) FROM movies
"""

QUERY_4_2 = """
SELECT sum(revenue) as tot_revenue, countries.iso_3166_1 as country_code FROM movies
JOIN movies_countries ON movies_countries.id_movie = movies.id
JOIN countries ON countries.id = movies_countries.id_movie
WHERE UPPER(countries.iso_3166_1) IN ('AL', 'AD', 'AM', 'AT', 'BY', 'BE', 'BA', 'BG', 'CH
    ', 'CY', 'CZ', 'DE', 'DK', 'EE', 'ES', 'FO', 'FI', 'FR', 'GB', 'GE', 'GI', 'GR', 'HU
    ', 'HR',
 'IE', 'IS', 'IT', 'LI', 'LT', 'LU', 'LV', 'MC', 'MK', 'MT', 'NO', 'NL', 'PL',
 'PT', 'RO', 'RU', 'SE', 'SI', 'SK', 'SM', 'TR', 'UA', 'VA')
AND YEAR(release_date) = {}
group by country_code
ORDER BY tot_revenue DESC LIMIT 10
"""



print("Connecting to spark")
spark = pyspark.sql.SparkSession.builder.master("local").appName("Film").
    enableHiveSupport().getOrCreate()
sc = spark.sparkContext

print("Obtaining files")
path = "./"
files = [f for f in listdir(path) if isfile(join(path, f))]
files = [f for f in files if "exported_" in f]
tables = {}
for f in files:
    s_time = time.time()
    name = f[9:len(f) - 4] #f[9:len(f) - 4] is to remove "exported" and ".csv" from the
        name
    print("Loading table", name)
    df = spark.read.option("header", "true").csv(join(path, f))
    tables[name] = df
    df.registerTempTable(name)
    print("Completed in {:.02f}s".format(time.time() - s_time))



s_time = time.time()
print(QUERY_1)
spark.sql(QUERY_1).show()
print("Completed in {:.02f}s".format(time.time() - s_time))

s_time = time.time()
print(QUERY_2)
spark.sql(QUERY_2).show()
print("Completed in {:.02f}s".format(time.time() - s_time))

s_time = time.time()
print(QUERY_3)
spark.sql(QUERY_3).show()
print("Completed in {:.02f}s".format(time.time() - s_time))

s_time = time.time()
print(QUERY_4_1)
years = spark.sql(QUERY_4_1)
print("Completed in {:.02f}s".format(time.time() - s_time))

years2 = []
for d in years.select("*").rdd.collect():
    years2.append(d.asDict["year(CAST(release_date AS DATE))"])
print("Completed in {:.02f}s".format(time.time() - s_time))

print(QUERY_4_2)
```

```
for year in years2:
        s_time = time.time()
        print(year)
        spark.sql(QUERY_4_2.format(year)).show()
        print("Completed in {:.02f}s".format(time.time() - s_time))
```

Query results

```
Connecting to spark
20/01/16 17:10:34 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
20/01/16 17:10:34 WARN NativeCodeLoader: Unable to load native-hadoop library for your
    platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(
    newLevel).
Obtaining files
Loading table movies
Completed in 3.54s
Loading table keywords
Completed in 0.34s
Loading table movies_languages
Completed in 0.34s
Loading table countries
Completed in 0.32s
Loading table jobs
Completed in 0.30s
Loading table languages
Completed in 0.29s
Loading table movies_genres
Completed in 0.28s
Loading table movies_companies
Completed in 0.23s
Loading table collections
Completed in 0.28s
Loading table ratings
Completed in 0.24s
Loading table movies_keywords
Completed in 0.28s
Loading table movies_countries
Completed in 0.25s
Loading table departments
Completed in 0.19s
Loading table characters
Completed in 0.18s
Loading table production_companies
Completed in 0.23s
Loading table movies_collection
Completed in 0.19s
Loading table people
Completed in 0.18s
Loading table genres
Completed in 0.21s


SELECT name FROM people WHERE id = (
        SELECT people_id FROM (
                SELECT COUNT(*) AS NUM, people_id FROM characters
                GROUP BY people_id, movie_id ORDER BY NUM DESC LIMIT 1
        ) AS alias_table
)

20/01/16 17:10:47 WARN ObjectStore: Failed to get database global_temp, returning
    NoSuchObjectException
+-------------+
|         name|
+-------------+
|Aubree Miller|
+-------------+
Completed in 10.76s

SELECT first(title), MAX(average), YEAR(release_date) as release_date FROM (
        SELECT
        first(title) as title,
        AVG(rating) AS average,
        first(release_date) as release_date
        FROM movies
        JOIN ratings ON ratings.movie_id = movies.id
        GROUP BY movie_id
) AS alias_table GROUP BY YEAR(release_date)
ORDER BY release_date DESC


+--------------------+-----------------+------------+
| first(title, false)|     max(average)|release_date|
+--------------------+-----------------+------------+
|             Avatar2|3.577777777777778|        2020|
|NickCannon:StandU...|              4.5|        2017|
|    SouthsidewithYou|              5.0|        2016|
|     TheBeautyInside|              4.5|        2015|
|          Monsterman|              4.5|        2014|
|                 Kid|              5.0|        2013|
|            Poolside|              4.4|        2012|
|                Will|4.538461538461538|        2011|
|     AnythingYouWant|              5.0|        2010|
|JimmyCarr:Telling...|4.391304347826087|        2009|
|        TheParanoids|4.333333333333333|        2008|
|WalkingTall:ThePa...|4.666666666666667|        2007|
|DrakeAndJoshGoHol...|4.259615384615385|        2006|
|      BacktoYouandMe|4.333333333333333|        2005|
|         ThePunisher|4.310679611650485|        2004|
|      LovetheHardWay|              5.0|        2003|
|     AWalktoRemember|              5.0|        2002|
|          NoMansLand|4.297709923664122|        2001|
|      AmericanPsycho|              4.5|        2000|
|        BeyondtheMat|4.357203751065644|        1999|
+--------------------+-----------------+------------+
only showing top 20 rows

Completed in 23.23s


SELECT A.year, A.name, B.name, C.title
FROM
(
SELECT first(name) as name, MAX(average), YEAR(release_date) AS year FROM (
        SELECT
                first(genres.name) as name,
                AVG(rating) AS average,
                first(release_date) as release_date
        FROM movies
        JOIN ratings ON ratings.movie_id = movies.id
        JOIN movies_genres ON movies_genres.id_movie = movies.id
        JOIN genres ON movies_genres.id_genre = genres.id
        GROUP BY genres.id
) AS alias_table1 GROUP BY YEAR(release_date)
ORDER BY year ) as A

JOIN
        (

        SELECT first(name) as name, MAX(tot_revenue), YEAR(release_date) AS year FROM (
                SELECT
                        first(genres.name) as name,
                        SUM(revenue) AS tot_revenue,
                        first(release_date) as release_date
                FROM movies
                JOIN ratings ON ratings.movie_id = movies.id
                JOIN movies_genres ON movies_genres.id_movie = movies.id
                JOIN genres ON movies_genres.id_genre = genres.id
                GROUP BY genres.id
        ) as alias_table3 GROUP BY YEAR(release_date)
        ORDER BY year

) as B ON  A.year=B.year
JOIN



(

        SELECT first(title) as title, first(average) as average, MAX(revenue), YEAR(
            release_date) AS year                                    FROM (
                SELECT
                        first(genres.name) as name,
                        AVG(rating) AS average,
                        first(release_date) as release_date,
                        first(revenue) as revenue,
                        first(title) as title
                FROM movies
                JOIN ratings ON ratings.movie_id = movies.ID
                JOIN movies_genres ON movies_genres.id_movie = movies.ID
                JOIN genres ON movies_genres.id_genre = genres.ID
                GROUP BY movies.id
        ) as alias_table15 GROUP BY YEAR(release_date), average
        ORDER BY year

)  as C ON A.year = C.year


+----+--------------+--------------+--------------------+
|year|          name|          name|               title|
+----+--------------+--------------+--------------------+
|1990|         Drama|         Drama|      GraveyardShift|
|1990|         Drama|         Drama|           HomeAlone|
|1990|         Drama|         Drama|         WhitePalace|
|1990|         Drama|         Drama|              Hamlet|
|1990|         Drama|         Drama|             Cadence|
|1990|         Drama|         Drama|    DanceswithWolves|
|1990|         Drama|         Drama|           DickTracy|
|1990|         Drama|         Drama|      TheExorcistIII|
|1990|         Drama|         Drama|         MyBlueHeaven|
|1990|         Drama|         Drama|         PrettyWoman|
|1990|         Drama|         Drama|Halfaouine:Boyoft...|
|1990|         Drama|         Drama|    DeathinBrunswick|
|1990|         Drama|         Drama|         ChildsPlay2|
|1990|         Drama|         Drama|     DaysofBeingWild|
|1990|         Drama|         Drama|      SpacedInvaders|
|1990|         Drama|         Drama|        Metropolitan|
|1990|         Drama|         Drama|      TheChallengers|
|1990|         Drama|         Drama|     ILoveYoutoDeath|
|1990|         Drama|         Drama|TakingCareofBusiness|
|1977|ScienceFiction|ScienceFiction|               Rabid|
+----+--------------+--------------+--------------------+
only showing top 20 rows

Completed in 33.91s


SELECT DISTINCT YEAR(release_date) FROM movies

Completed in 0.01s
Completed in 1.02s

SELECT sum(revenue) as tot_revenue, countries.iso_3166_1 as country_code FROM movies
JOIN movies_countries ON movies_countries.id_movie = movies.id
JOIN countries ON countries.id = movies_countries.id_movie
WHERE UPPER(countries.iso_3166_1) IN ('AL', 'AD', 'AM', 'AT', 'BY', 'BE', 'BA', 'BG', 'CH
     ', 'CY', 'CZ', 'DE', 'DK', 'EE', 'ES', 'FO', 'FI', 'FR', 'GB', 'GE', 'GI', 'GR', 'HU
     ', 'HR',
 'IE', 'IS', 'IT', 'LI', 'LT', 'LU', 'LV', 'MC', 'MK', 'MT', 'NO', 'NL', 'PL',
 'PT', 'RO', 'RU', 'SE', 'SI', 'SK', 'SM', 'TR', 'UA', 'VA')
```

```
AND YEAR(release_date) = {}
group by country_code
ORDER BY tot_revenue DESC LIMIT 10


Completed in 1.24s
2017
+-------------+------------+
|  tot_revenue|country_code|
+-------------+------------+
|1.238764765E9|          FI|
|1.020457354E9|          GB|
| 4.98814908E8|          IT|
| 2.45615916E8|          DE|
| 1.79180063E8|          BE|
| 1.31799925E8|          SE|
| 1.10824373E8|          LU|
|  4.4380155E7|          CZ|
|  2.0497844E7|          RU|
|     457084.0|          FR|
+-------------+------------+

Completed in 1.32s
2016
+-------------+------------+
|  tot_revenue|country_code|
+-------------+------------+
|2.115305587E9|          FR|
|1.148607403E9|          IT|
| 4.12577395E8|          BE|
| 3.05742021E8|          RU|
| 2.99600553E8|          CZ|
| 2.87724753E8|          SE|
| 2.29147509E8|          FI|
| 1.80606856E8|          NL|
| 1.00510864E8|          GB|
|  3.6061704E7|          DE|
+-------------+------------+

Completed in 1.25s
2015
+-------------+------------+
|  tot_revenue|country_code|
+-------------+------------+
|2.068254024E9|          TR|
| 7.55036366E8|          FR|
|  4.9003105E8|          IT|
| 3.68871007E8|          NL|
|  2.2298789E8|          GB|
| 1.34836774E8|          ES|
| 1.22513057E8|          LU|
|  4.6725901E7|          DE|
|  4.2426912E7|          BE|
|  1.5730665E7|          IE|
+-------------+------------+

Completed in 1.19s

[YEARS BEFORE 2015 HAVE BEEN REDACTED FOR CLARITY]
```

6 Performance analysis

Each query has been run on the same hardware, this allows us to compare the performances of different DBMSs.
Each task running time has been summarized in the following table:

	Task 1	Task 2	Task 3	Task 4[1]
Oracle MySQL	3.376 s	12.018 s	4h 41 m 47.433 s	0.518 s
MongoDB	3.35 s	15.64 s	168 s	2.58 s
Apache Spark	10.76 s	23.23 s	33.91 s	1.2533 s

Note: in the table the Task 4 fields refers to the average time required to obtain the ranking of a single year

The most interesting comparison is between Apache Spark and Oracle MySQL, since they are both relational DBMSs and are running the same queries albeit with some minor syntax adjustments; both DBMSs are also using the same Entity-Relation schema and dataset structure.
An interesting fact is the time required for Oracle MySQL to complete the third task. Such a large running time however has not been highlighted in Apache Spark, which is also a relational DBMS; therefore we can hypothesize that the difference is determined by some kind of internal query optimization that MySQL is unable to perform. We can also notice that on all other queries Oracle MySQL is much faster than Apache Spark, this is probably due to the fact that Spark is running in Standalone Mode instead of using a proper cluster and this significantly worsens constant factors. This also corroborates our initial hypothesis that Apache Spark is performing some internal optimizations on the third task.

7 Conclusions

The dataset we chose presented several more challenges when compared to the "soccer" one, we were however able to fully fulfill the assignment by designing an optimized Entity-Relationship model for importing into the two relational DBMS and by exploiting the flexiblity of the non-relational DBMS MongoDB.
We also took advantage of the strengths of each platform, for example, by taking a NoSQL approach to the MongoDB problem we were able to run each query in a single MongoDB aggregation pipeline.
The project has also been a great opportunity to apply and deepen the knowledge obtained during the course; a through understanding of theoretical concepts such as Normalization, Indexes and B+Trees, is what allowed us to further reduce the running time of each query.