HÖGSKOLAN
DALARNA

Data Warehousing (AIK232)

REPORT ON LAB-2

Group members

Abu Abdullah Dhrubo

Zigga Kweku

Nahom Gebremeskel

Chinonso Williams Ubani

Table of content

1. Introduction

In this project, we have implemented an **ETL** in python using the Pandas library and other libraries on the given dataset. **ETL** stands for **Extract**, **Transform** and **Load**, thus performing all the conversions, summarization, and changes needed to transform data into a unified format in the Data warehouse. The data is extracted using "The Movie Database API".

The database consists of data on several different movie characteristics. API key was used to safeguard the database. An application programming interface (API) is a code that is used to identify and authenticate a user or application. It also serves as a special identity as well as a secret token for authentication. In order to access the database an API key was requested by creating an account and following the instructions listed on the following link: https://developers.themoviedb.org/3/getting-started/authentication.

This project consists of 4 sections, section-1 Introduction, section-2 aim, section-3 methodology, section-4, and section-5 conclusion.

2. Aim / Tasks

The aim of this lab is to:

- Formatting data from JSON responses.
- Implement **ETL**.
- Manipulate the data from the database.
- Creating 3 CSV files.

3. Methodology

Two different tools have been utilized to perform the previously mentioned tasks or aims. Python (Jupyter notebook) and SQL are the two main tools extensively consumed in this project. With python several different libraries were used for different tasks as required. Below are the list of the libraries: -

- import requests
- import json
- import pandas as pd
- from datetime import datetime

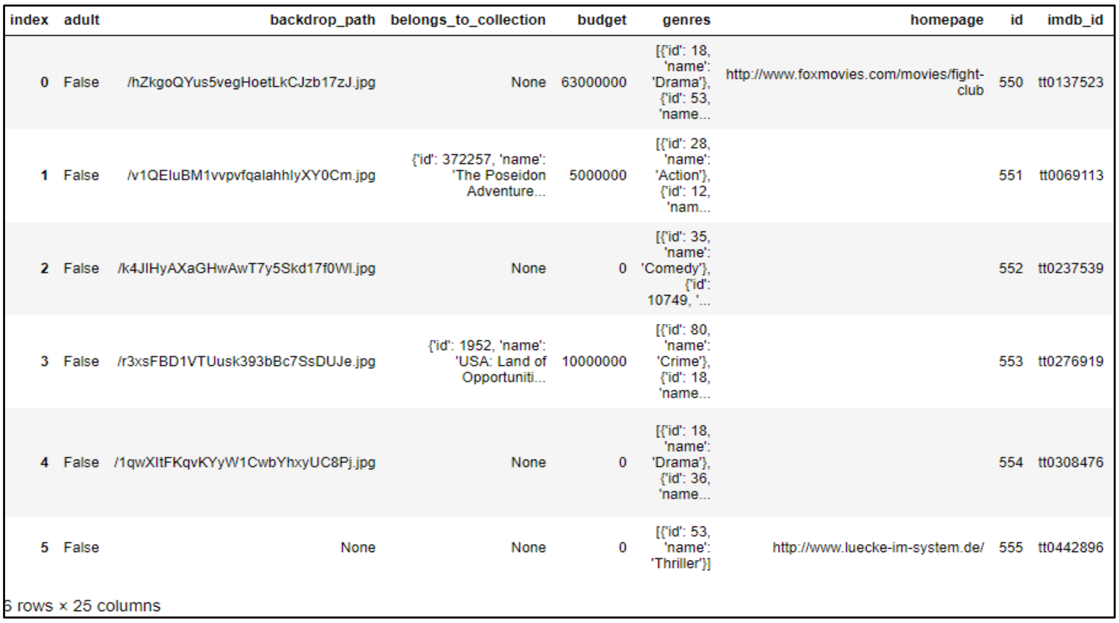
SQL is used in order to load the transformed data into a relational database.

4. Implementation

Once we got access to the movie database and set the working environment to code in python where we can copy the API key and write a script, the ETL models (EXTRACT, TRANSFORM, LOAD) are used to analyze our data from the Movie database API and below is a detailed explanation:

4.1. Extract

Using the get method in python we have managed to extract 6 movies from 550 to 555. To extract all 6 movies, we used a for loop parsing the different APIs on each iteration. Once extracted, it is saved in a dataframe using from_dict(). See the below outputs.



index	adult	backdrop_path	belongs_to_collection	budget	genres	homepage	id	imdb_id
0	False	/hZkgoQYus5vegHoetLkCJzb17zJ.jpg	None	63000000	[{'id': 18, 'name': 'Drama'}, {'id': 53, 'name...	http://www.foxmovies.com/movies/fight-club	550	tt0137523
1	False	/v1QEluBM1vvpvfqalahhlyXY0Cm.jpg	{'id': 372257, 'name': 'The Poseidon Adventure...	5000000	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...		551	tt0069113
2	False	/k4JIHyAXaGHwAwT7y5Skd17f0WI.jpg	None	0	[{'id': 35, 'name': 'Comedy'}, {'id': 10749, '...		552	tt0237539
3	False	/r3xsFBD1VTUusk393bBc7SsDUJe.jpg	{'id': 1952, 'name': 'USA: Land of Opportuniti...	10000000	[{'id': 80, 'name': 'Crime'}, {'id': 18, 'name...		553	tt0276919
4	False	/1qwXItFKqvKYyW1CwbYhxyUC8Pj.jpg	None	0	[{'id': 18, 'name': 'Drama'}, {'id': 36, 'name...		554	tt0308476
5	False	None	None	0	[{'id': 53, 'name': 'Thriller'}]	http://www.luecke-im-system.de/	555	tt0442896

6 rows × 25 columns

Figure.1- Extracted data from “The Movie Database API”.

4.2. Transformation

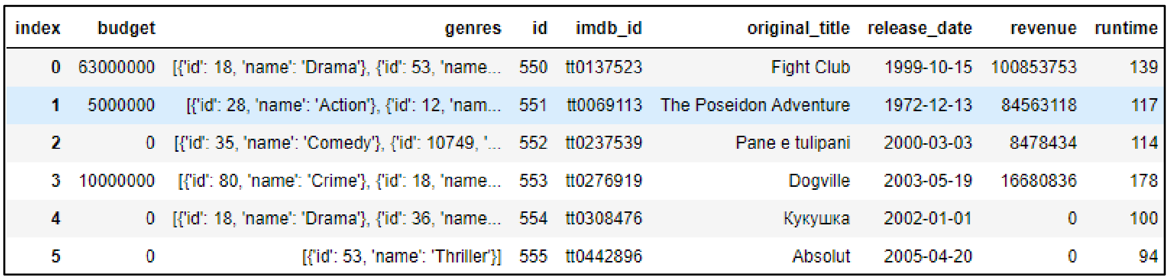
In this section, we have transformed the data into two main sections,

Transformation 1: in this section, release_date column was extracted and stored in a data frame. Furthermore, the **genre** column is also transformed and expanded.

Transformation 2: The **datatime & runtime** are transformed and expanded.

4.2.1. Transformation-1

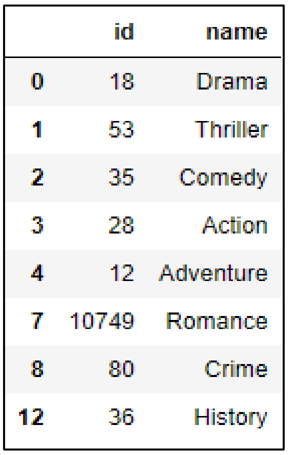
Once the data was extracted, we noticed that the data have many information which is not necessary for our project. As a result, the data was transformed in to the required format as per given instruction for the project. Columns 'budget', 'genres', 'id', 'imdb_id', 'original_title', 'release_date', 'revenue', and 'runtime' were extracted and save in to one dataframe called movies.



index	budget	genres	id	imdb_id	original_title	release_date	revenue	runtime
0	63000000	[{'id': 18, 'name': 'Drama'}, {'id': 53, 'name...	550	tt0137523	Fight Club	1999-10-15	100853753	139
1	5000000	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	551	tt0069113	The Poseidon Adventure	1972-12-13	84563118	117
2	0	[{'id': 35, 'name': 'Comedy'}, {'id': 10749, '...	552	tt0237539	Pane e tulipani	2000-03-03	8478434	114
3	10000000	[{'id': 80, 'name': 'Crime'}, {'id': 18, 'name...	553	tt0276919	Dogville	2003-05-19	16680836	178
4	0	[{'id': 18, 'name': 'Drama'}, {'id': 36, 'name...	554	tt0308476	Кукушка	2002-01-01	0	100
5	0	[{'id': 53, 'name': 'Thriller'}]	555	tt0442896	Absolut	2005-04-20	0	94

Figure. 2- contents of Movies dataframe.

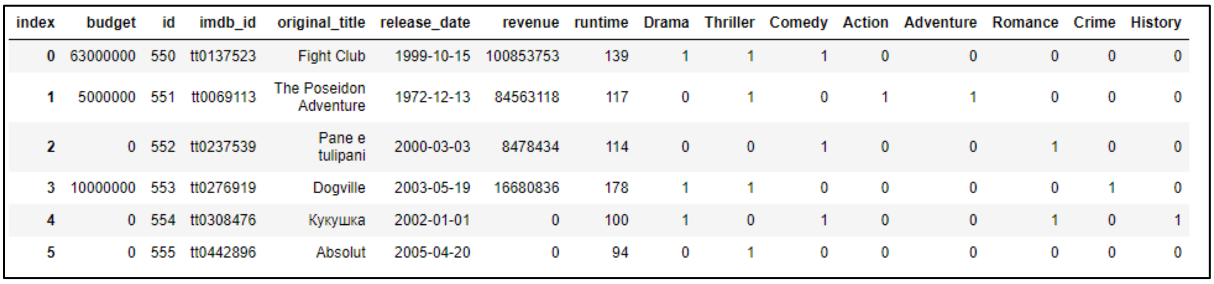
As we can see in the above figure the column genres are quite difficult to understand. So, transformation is needed only for the column genres. To do this, a separate table for genres and a column of explode-out lists is created and the duplicate data are dropped.



	id	name
0	18	Drama
1	53	Thriller
2	35	Comedy
3	28	Action
4	12	Adventure
7	10749	Romance
8	80	Crime
12	36	History

Figure.3 Genres table after transformation

Finally in this section, we have created a set of columns of frequency values from the main table based on the genre column. If the genre for a specific movie is Yes, then it will be denoted by “1” or else by “0”. Below you can see the data with frequency values.

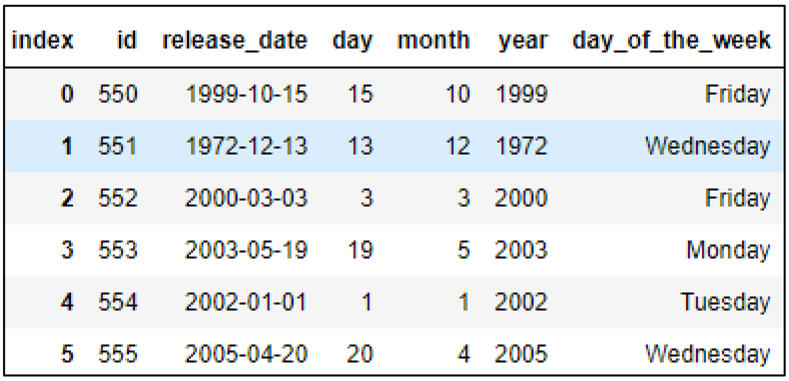


index	budget	id	imdb_id	original_title	release_date	revenue	runtime	Drama	Thriller	Comedy	Action	Adventure	Romance	Crime	History
0	63000000	550	tt0137523	Fight Club	1999-10-15	100853753	139	1	1	1	0	0	0	0	0
1	5000000	551	tt0069113	The Poseidon Adventure	1972-12-13	84563118	117	0	1	0	1	1	0	0	0
2	0	552	tt0237539	Pane e tulipani	2000-03-03	8478434	114	0	0	1	0	0	1	0	0
3	10000000	553	tt0276919	Dogville	2003-05-19	16680836	178	1	1	0	0	0	0	1	0
4	0	554	tt0308476	Кукушка	2002-01-01	0	100	1	0	1	0	0	1	0	1
5	0	555	tt0442896	Absolut	2005-04-20	0	94	0	1	0	0	0	0	0	0

Figure. 4. Transformed main table with frequency.

4.2.2. Transformation 2

In this section, we have transformed the column release_date to, day, month, year, and day of the week after creating a temporary date dataframe and transforming it. A for loop was used in the transformation process and different functions in python like iterrows, datetime.strptime, strftime were employed.

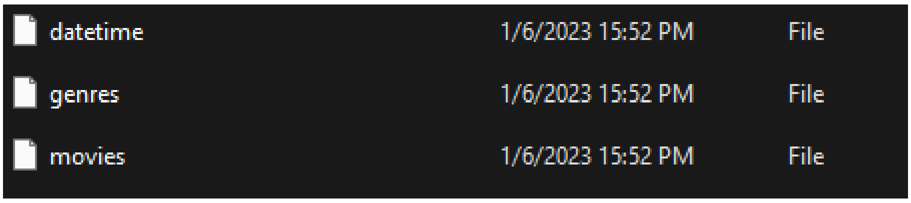


index	id	release_date	day	month	year	day_of_the_week
0	550	1999-10-15	15	10	1999	Friday
1	551	1972-12-13	13	12	1972	Wednesday
2	552	2000-03-03	3	3	2000	Friday
3	553	2003-05-19	19	5	2003	Monday
4	554	2002-01-01	1	1	2002	Tuesday
5	555	2005-04-20	20	4	2005	Wednesday

Figure. 5- transformed datetime column

5. Load

Finally, after all the data are extracted and transformed, the tables are loaded into three different CVS files called **'movies', 'genres', and 'datetimes'.**

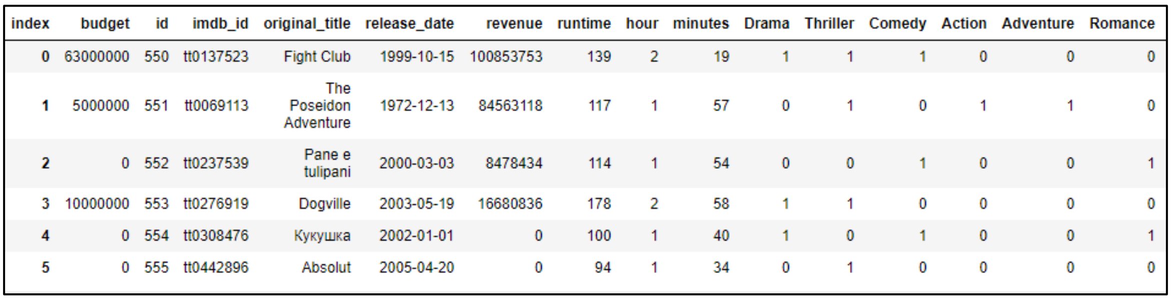


datetime	1/6/2023 15:52 PM	File
genres	1/6/2023 15:52 PM	File
movies	1/6/2023 15:52 PM	File

Figure. 6- List of CSV files

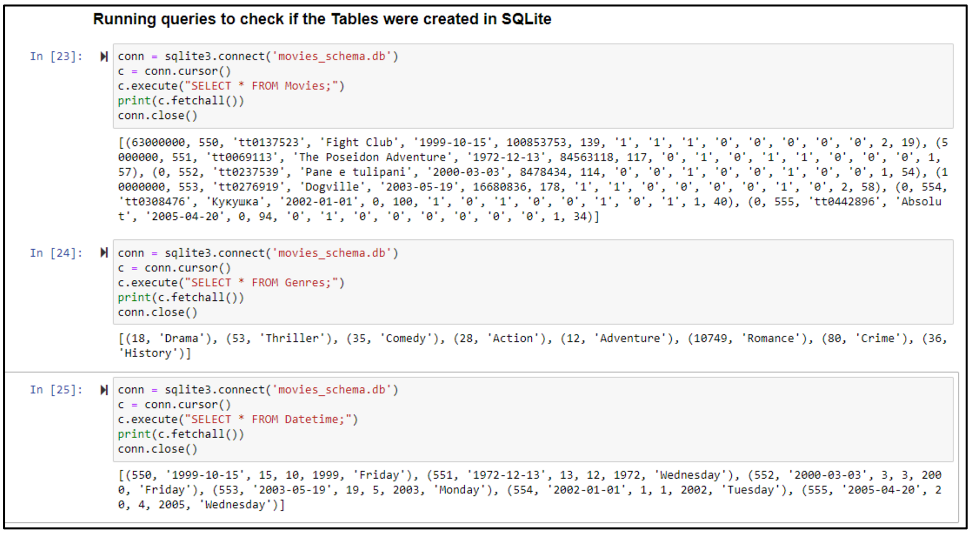
6. Additional task

The task is to build upon the Movie database computed in methodology A. The findings were to transform run time into hours and minutes and load the data in a relational database instead of Comma separated value (CSV). The relational database uses SQlite3(open a cursor to perform the database operations) to create database tables. Below is the screenshot of the findings:



index	budget	id	imdb_id	original_title	release_date	revenue	runtime	hour	minutes	Drama	Thriller	Comedy	Action	Adventure	Romance
0	63000000	550	tt0137523	Fight Club	1999-10-15	100853753	139	2	19	1	1	1	0	0	0
1	5000000	551	tt0069113	The Poseidon Adventure	1972-12-13	84563118	117	1	57	0	1	0	1	1	0
2	0	552	tt0237539	Pane e tulipani	2000-03-03	8478434	114	1	54	0	0	1	0	0	1
3	10000000	553	tt0276919	Dogville	2003-05-19	16680836	178	2	58	1	1	0	0	0	0
4	0	554	tt0308476	Кукушка	2002-01-01	0	100	1	40	1	0	1	0	0	1
5	0	555	tt0442896	Absolut	2005-04-20	0	94	1	34	0	1	0	0	0	0

Figure. 7- Transformed runtime into hours and minutes.



Running queries to check if the Tables were created in SQLite

```
In [23]:  conn = sqlite3.connect('movies_schema.db')
          c = conn.cursor()
          c.execute("SELECT * FROM Movies;")
          print(c.fetchall())
          conn.close()

          [(63000000, 550, 'tt0137523', 'Fight Club', '1999-10-15', 100853753, 139, '1', '1', '1', '0', '0', '0', '0', '0', 2, 19), (5
          000000, 551, 'tt0069113', 'The Poseidon Adventure', '1972-12-13', 84563118, 117, '0', '1', '0', '1', '1', '0', '0', '0', 1,
          57), (0, 552, 'tt0237539', 'Pane e tulipani', '2000-03-03', 8478434, 114, '0', '0', '1', '0', '0', '1', '0', '0', 1, 54), (1
          0000000, 553, 'tt0276919', 'Dogville', '2003-05-19', 16680836, 178, '1', '1', '0', '0', '0', '0', '1', '0', 2, 58), (0, 554,
          'tt0308476', 'Кукушка', '2002-01-01', 0, 100, '1', '0', '1', '0', '0', '1', '0', '1', 1, 40), (0, 555, 'tt0442896', 'Absolu
          t', '2005-04-20', 0, 94, '0', '1', '0', '0', '0', '0', '0', '0', 1, 34)]

In [24]:  conn = sqlite3.connect('movies_schema.db')
          c = conn.cursor()
          c.execute("SELECT * FROM Genres;")
          print(c.fetchall())
          conn.close()

          [(18, 'Drama'), (53, 'Thriller'), (35, 'Comedy'), (28, 'Action'), (12, 'Adventure'), (10749, 'Romance'), (80, 'Crime'), (36,
          'History')]

In [25]:  conn = sqlite3.connect('movies_schema.db')
          c = conn.cursor()
          c.execute("SELECT * FROM Datetime;")
          print(c.fetchall())
          conn.close()

          [(550, '1999-10-15', 15, 10, 1999, 'Friday'), (551, '1972-12-13', 13, 12, 1972, 'Wednesday'), (552, '2000-03-03', 3, 3, 200
          0, 'Friday'), (553, '2003-05-19', 19, 5, 2003, 'Monday'), (554, '2002-01-01', 1, 1, 2002, 'Tuesday'), (555, '2005-04-20', 2
          0, 4, 2005, 'Wednesday')]
```

Figure. 8- Upon creating the tables in SQLite Database.

7. Conclusion

Business entities produce a heterogeneous amount of data daily through their operation and cannot be used for decision-making in most cases because it needs to be structured correctly. In data warehousing, ETL tools play a significant role by aiding various stakeholders to transform data into information, thus helping in optimal decision-making.