Database Project Report

Student 1 : Fahadh Mohamed – 62720

Student 2 : Kuiyu Ding – 62716

Project Overview :

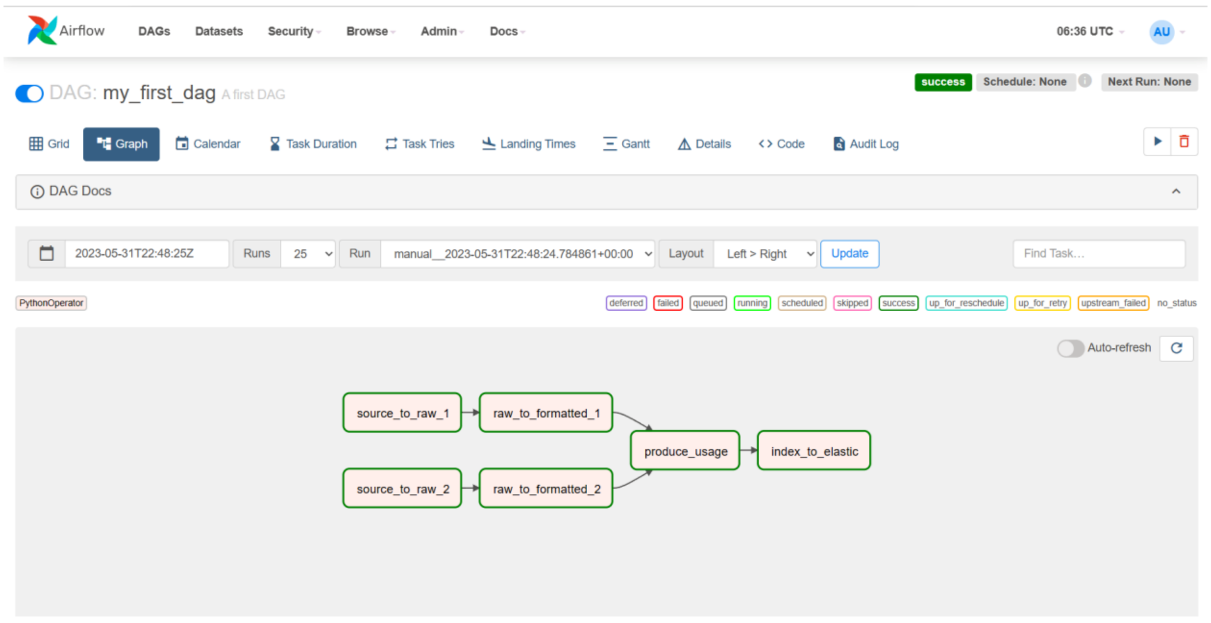
In this project, we analyse the data of movies , we have showed two usages, First usage is to analyse the average rating for each year and the second usage is to recommend the movies based on highest numVotes for the movies came after 2000(year) and has rating above 8.

Airflow :

Airflow is an open-source platform designed to programmatically author, schedule, and monitor workflows. It allows users to define complex data pipelines as a collection of tasks, each with its dependencies and scheduling properties.

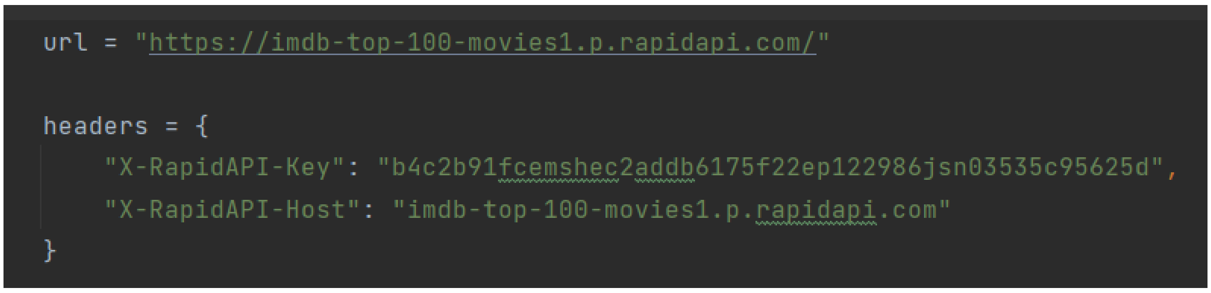
Dag:

We created a dag to perform set of tasks which converts from source to formatted and combine the results, finally it will show in the website using elasticsearch and kibana. Each task will run automatically because of the airflow.



Source_to_raw_1 :

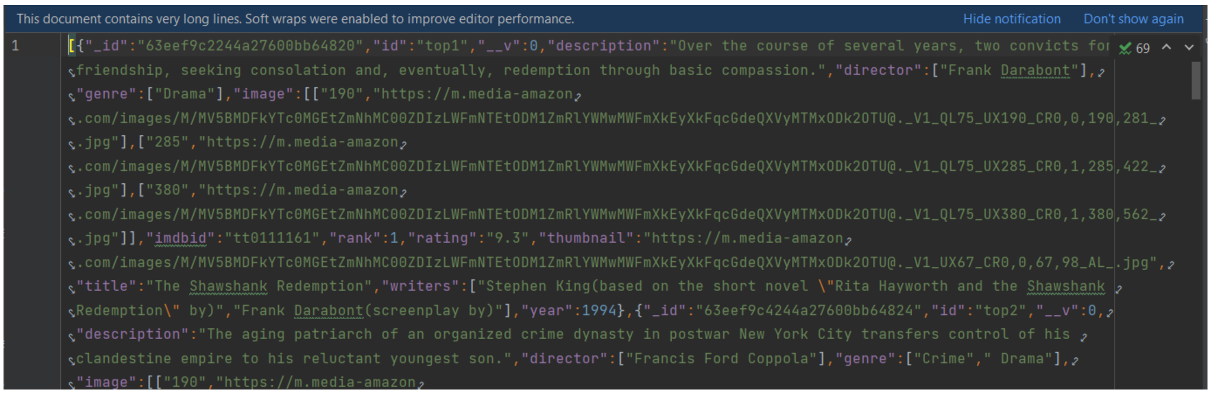
For the source 1, we have to write a code to get the data from api, the api is taken from rapidapi website which consists of top 100 imdb movies.



```
url = "https://imdb-top-100-movies1.p.rapidapi.com/"

headers = {
    "X-RapidAPI-Key": "b4c2b91fcemshec2addb6175f22ep122986jsn03535c95625d",
    "X-RapidAPI-Host": "imdb-top-100-movies1.p.rapidapi.com"
}
```

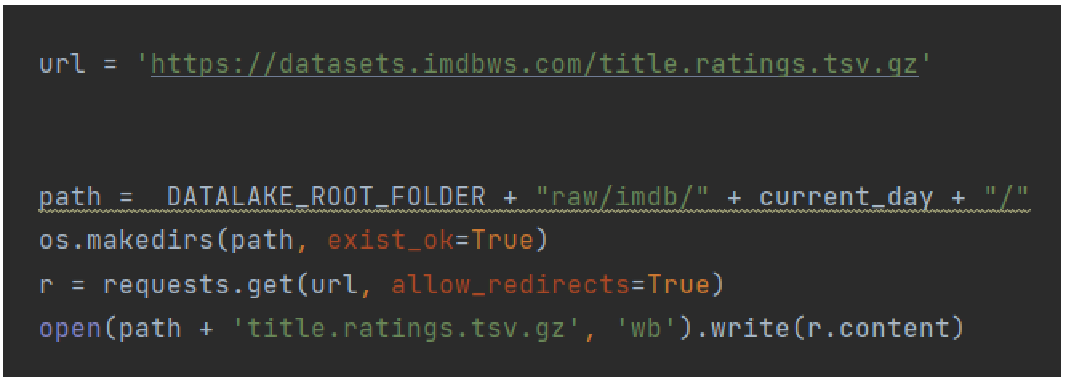
I get the data as Json format and I saved it in the raw folder. You can see the data below,


This document contains very long lines. Soft wraps were enabled to improve editor performance.
Hide notification
Don't show again
69

```
[{"_id":"63eef9c2244a27600bb64820","id":"top1","__v":0,"description":"Over the course of several years, two convicts for
friendship, seeking consolation and, eventually, redemption through basic compassion.","director":["Frank Darabont"],
"genre":["Drama"],"image":[["190","https://m.media-amazon
.com/images/M/MV5BMDFkYTc0MGEtZmNhMC00ZDIzLWFmNTEtODM1ZmRlYWMwMWFmXkEyXkFqcGdeQXVyMTMxODk2OTU@._V1_QL75_UX190_CR0,0,190,281_
.jpg"],["285","https://m.media-amazon
.com/images/M/MV5BMDFkYTc0MGEtZmNhMC00ZDIzLWFmNTEtODM1ZmRlYWMwMWFmXkEyXkFqcGdeQXVyMTMxODk2OTU@._V1_QL75_UX285_CR0,1,285,422_
.jpg"],["380","https://m.media-amazon
.com/images/M/MV5BMDFkYTc0MGEtZmNhMC00ZDIzLWFmNTEtODM1ZmRlYWMwMWFmXkEyXkFqcGdeQXVyMTMxODk2OTU@._V1_QL75_UX380_CR0,1,380,562_
.jpg"]],"imdbid":"tt0111161","rank":1,"rating":"9.3","thumbnail":"https://m.media-amazon
.com/images/M/MV5BMDFkYTc0MGEtZmNhMC00ZDIzLWFmNTEtODM1ZmRlYWMwMWFmXkEyXkFqcGdeQXVyMTMxODk2OTU@._V1_UX67_CR0,0,67,98_AL_.jpg",
"title":"The Shawshank Redemption","writers":["Stephen King(based on the short novel \"Rita Hayworth and the Shawshank
Redemption\" by)","Frank Darabont(screenplay by)"],"year":1994},{"_id":"63eef9c4244a27600bb64824","id":"top2","__v":0,
"description":"The aging patriarch of an organized crime dynasty in postwar New York City transfers control of his
clandestine empire to his reluctant youngest son.","director":["Francis Ford Coppola"],"genre":["Crime"," Drama"],
"image":[["190","https://m.media-amazon
```

Source_to_raw_2 :

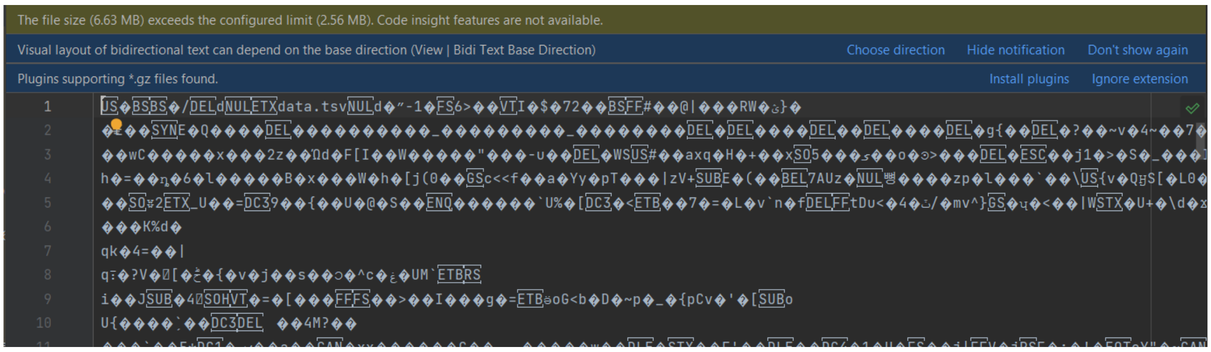
Similarly for the source 2, we have to write a code to get the data from api, the api is taken from imdb website, which shows ratings and numVotes of the movies.



```
url = 'https://datasets.imdbws.com/title.ratings.tsv.gz'


path =  DATALAKE_ROOT_FOLDER + "raw/imdb/" + current_day + "/"
os.makedirs(path, exist_ok=True)
r = requests.get(url, allow_redirects=True)
open(path + 'title.ratings.tsv.gz', 'wb').write(r.content)
```

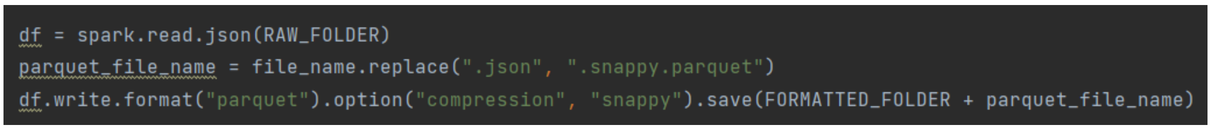
I get the data as tsv.gz file and I saved it in raw folder. You can see the data below.



```
The file size (6.63 MB) exceeds the configured limit (2.56 MB). Code insight features are not available.
Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction)    Choose direction    Hide notification    Don't show again
Plugins supporting *.gz files found.    Install plugins    Ignore extension
```

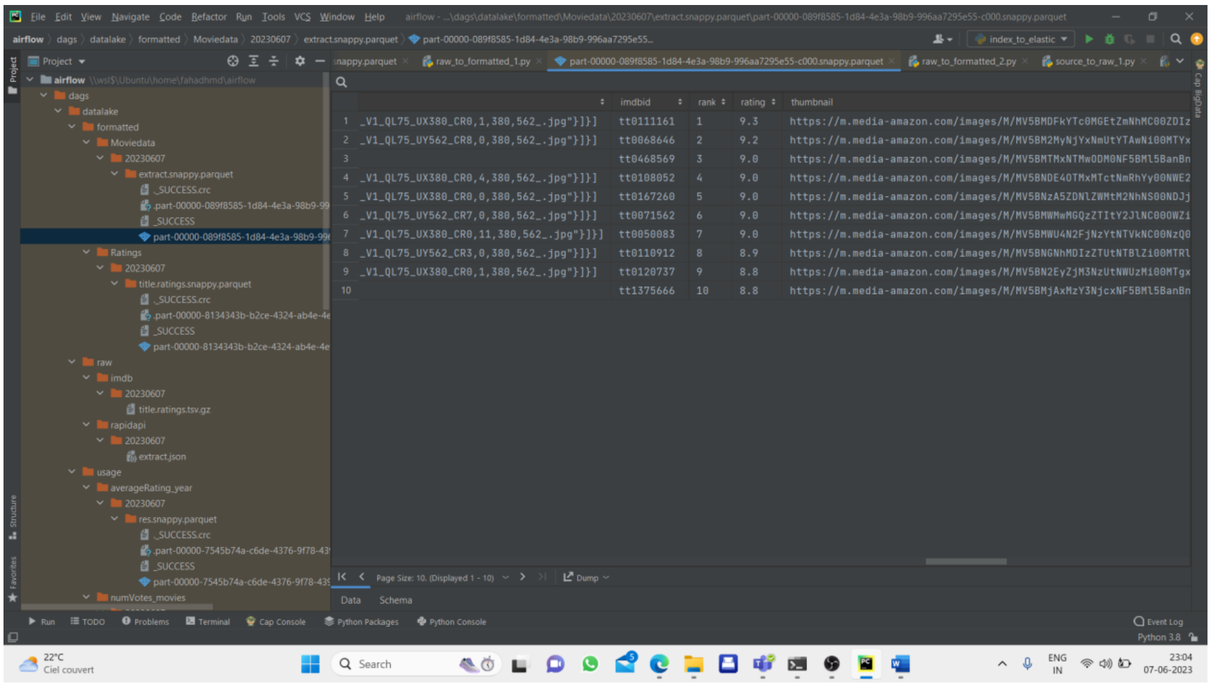
Raw_to_formatted_1:

Now the data is in json format, it is not suitable for analysis. So, we have to convert into parquet format. We used spark to convert the data into parquet format.



```
df = spark.read.json(RAW_FOLDER)
parquet_file_name = file_name.replace(".json", ".snappy.parquet")
df.write.format("parquet").option("compression", "snappy").save(FORMATTED_FOLDER + parquet_file_name)
```

After that I saved the data in the formatted folder. You can see the data below, which shows id, title, description of the movies, imdbid, rating, release year, director name, genre etc…



	imdbid	rank	rating	thumbnail
_V1_QL75_UX380_CR0,1,380,562_.jpg"}]}]	tt0111161	1	9.3	https://m.media-amazon.com/images/M/MV5BMDFkYTc0MGEtZmNhMC00ZDIz
_V1_QL75_UY562_CR8,0,380,562_.jpg"}]}]	tt0068646	2	9.2	https://m.media-amazon.com/images/M/MV5BM2MyNjYxNmUtYTAwNi00MTYx
	tt0468569	3	9.0	https://m.media-amazon.com/images/M/MV5BMTMxNTMwODM0NF5BMl5BanBn
_V1_QL75_UX380_CR0,4,380,562_.jpg"}]}]	tt0108052	4	9.0	https://m.media-amazon.com/images/M/MV5BNDE4OTMxMTctNmRhYy00NWE2
_V1_QL75_UX380_CR0,0,380,562_.jpg"}]}]	tt0167260	5	9.0	https://m.media-amazon.com/images/M/MV5BNzA5ZDNlZWMtM2NhNS00NDJj
_V1_QL75_UY562_CR7,0,380,562_.jpg"}]}]	tt0071562	6	9.0	https://m.media-amazon.com/images/M/MV5BMWMwMGQzZTItY2JlNC00OWZi
_V1_QL75_UX380_CR0,11,380,562_.jpg"}]}]	tt0050083	7	9.0	https://m.media-amazon.com/images/M/MV5BMWU4N2FjNzYtNTVkNC00NzQ0
_V1_QL75_UY562_CR3,0,380,562_.jpg"}]}]	tt0110912	8	8.9	https://m.media-amazon.com/images/M/MV5BNGNhMDIzZTUtNTBlZi00MTRl
_V1_QL75_UX380_CR0,1,380,562_.jpg"}]}]	tt0120737	9	8.8	https://m.media-amazon.com/images/M/MV5BN2EyZjM3NzUtNWUzMi00MTgx
	tt1375666	10	8.8	https://m.media-amazon.com/images/M/MV5BMjAxMzY3NjcxNF5BMl5BanBn

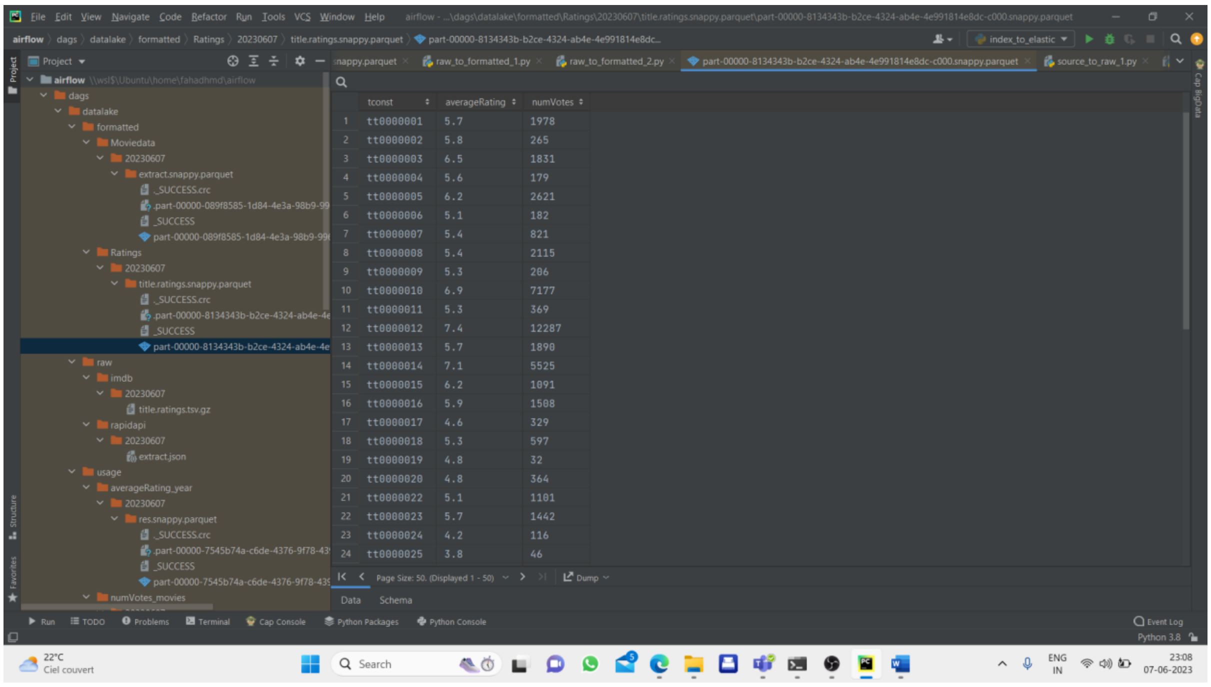
Raw_to_formatted_2:

Similarly we have to do this for raw_to_formatted_2, the data is in tsv.gz format, it is not suitable for analysis. So, we have to convert into parquet format. We used spark to convert the data into parquet format.

```
# Read TSV data using Spark
df = spark.read.option("header", True).option("delimiter", "\t").csv(RAW_FOLDER)

# Write data as Parquet with Snappy compression
parquet_file_name = file_name.replace(".tsv.gz", ".snappy.parquet")
df.write.format("parquet").option("compression", "snappy").save(FORMATTED_FOLDER + parquet_file_name)
```

And we saved the parquet data in formatted folder. You can see the data below,



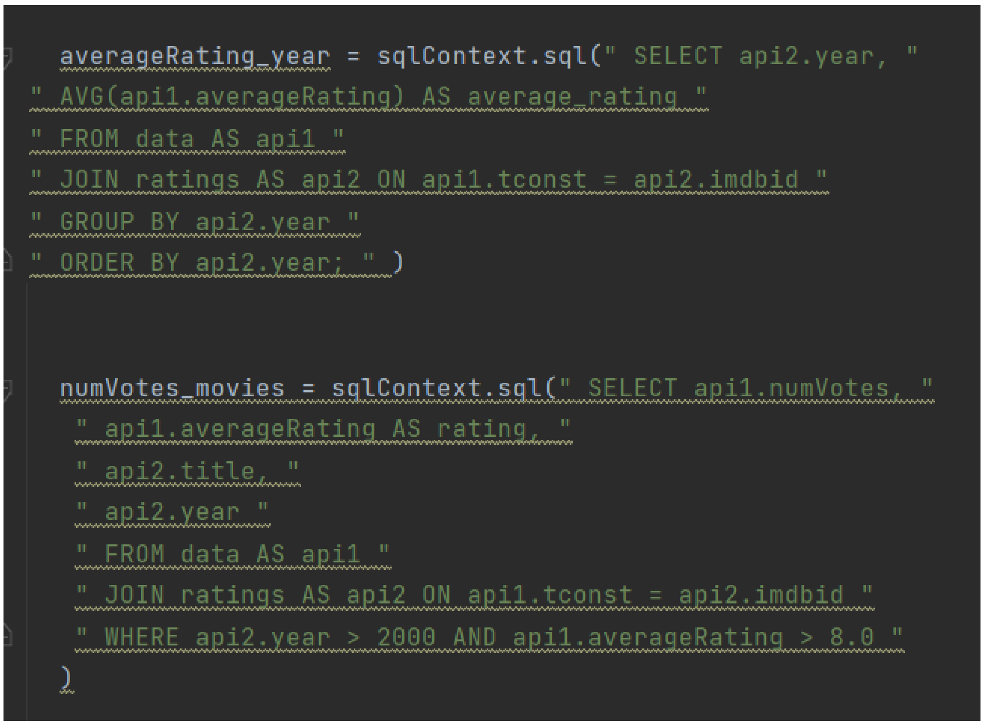
```
airflow - ...\dags\datalake\formatted\Ratings\20230607\title.ratings.snappy.parquet\part-00000-8134343b-b2ce-4324-ab4e-4e991814e8dc-c000.snappy.parquet
```

	tconst	averageRating	numVotes
1	tt0000001	5.7	1978
2	tt0000002	5.8	265
3	tt0000003	6.5	1831
4	tt0000004	5.6	179
5	tt0000005	6.2	2621
6	tt0000006	5.1	182
7	tt0000007	5.4	821
8	tt0000008	5.4	2115
9	tt0000009	5.3	206
10	tt0000010	6.9	7177
11	tt0000011	5.3	369
12	tt0000012	7.4	12287
13	tt0000013	5.7	1890
14	tt0000014	7.1	5525
15	tt0000015	6.2	1091
16	tt0000016	5.9	1508
17	tt0000017	4.6	329
18	tt0000018	5.3	597
19	tt0000019	4.8	32
20	tt0000020	4.8	364
21	tt0000022	5.1	1101
22	tt0000023	5.7	1442
23	tt0000024	4.2	116
24	tt0000025	3.8	46

Page Size: 50. (Displayed 1 - 50)

Produce_usage :

After converting this data to parquet, we have to combine this data. We used spark to combine this data, we have to write a SQL query to join the data by using (JOIN), here we joined the data using imdbid from both of the data.



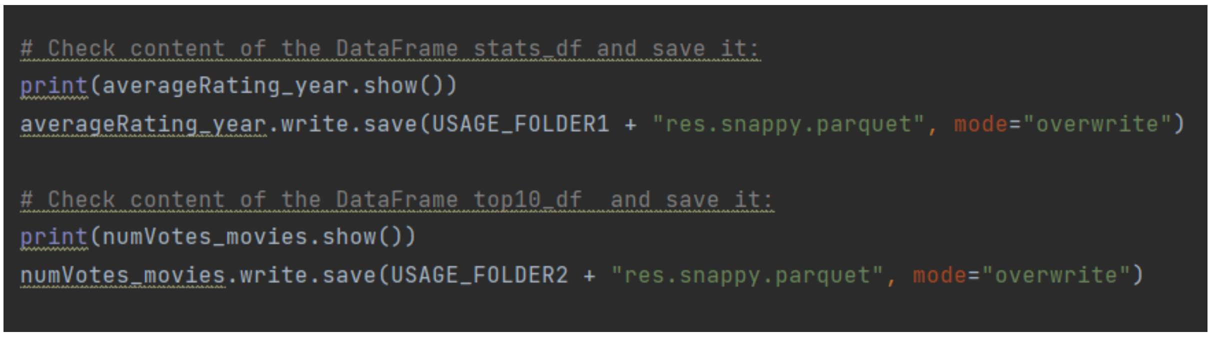
```
averageRating_year = sqlContext.sql(" SELECT api2.year, "
" AVG(api1.averageRating) AS average_rating "
" FROM data AS api1 "
" JOIN ratings AS api2 ON api1.tconst = api2.imdbid "
" GROUP BY api2.year "
" ORDER BY api2.year; " )


numVotes_movies = sqlContext.sql(" SELECT api1.numVotes, "
 " api1.averageRating AS rating, "
 " api2.title, "
 " api2.year "
 " FROM data AS api1 "
 " JOIN ratings AS api2 ON api1.tconst = api2.imdbid "
 " WHERE api2.year > 2000 AND api1.averageRating > 8.0 "
)
```

1st usage : we joined a sql query using imdbid from both of the formatted data and calculated average rating by each year.

2nd usage : Similarly, we joined a sql query using imdbid from both of the formatted data and we displayed the data that has above average rating 8 and the movies above the year 2000. It shows numVotes, rating, title and year.

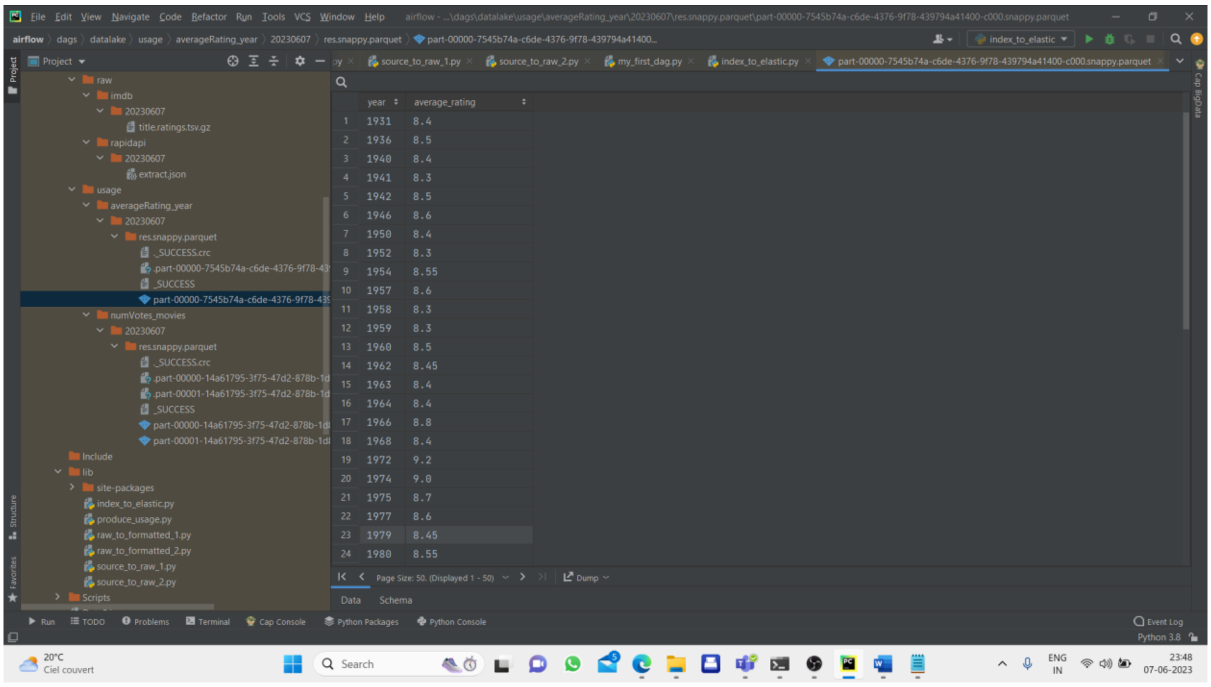
And then, we converted the data into parquet format and saved the data in usage folder.



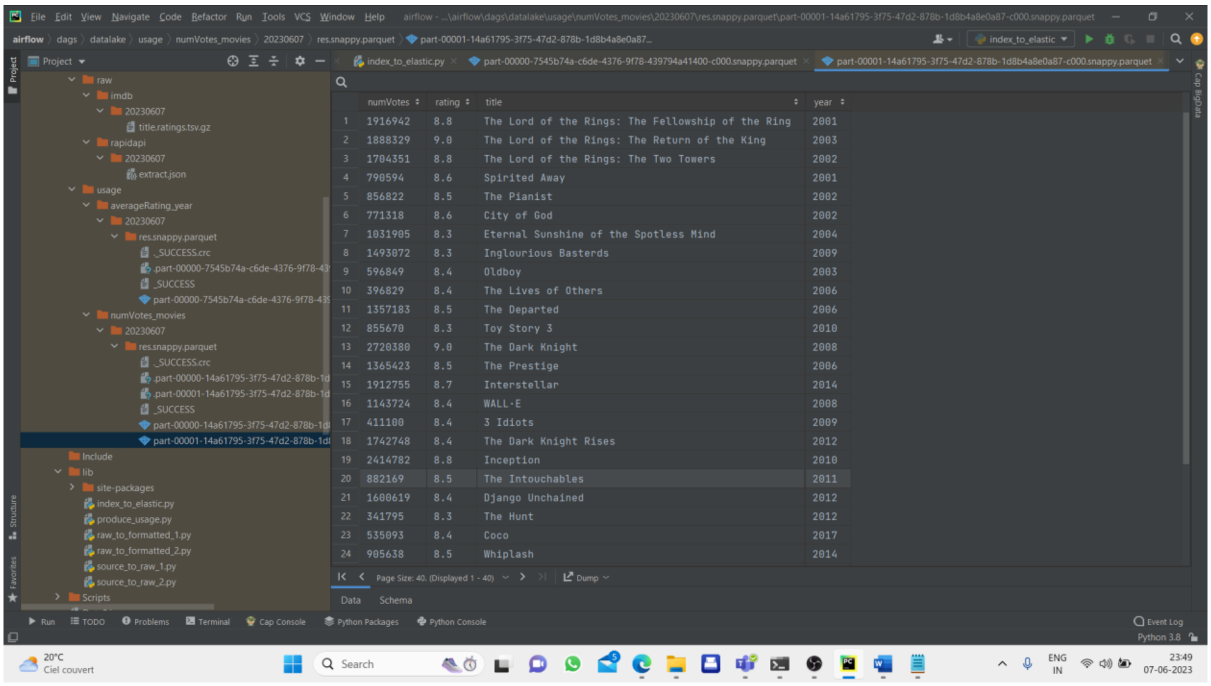
```
# Check content of the DataFrame stats_df and save it:
print(averageRating_year.show())
averageRating_year.write.save(USAGE_FOLDER1 + "res.snappy.parquet", mode="overwrite")

# Check content of the DataFrame top10_df  and save it:
print(numVotes_movies.show())
numVotes_movies.write.save(USAGE_FOLDER2 + "res.snappy.parquet", mode="overwrite")
```

You can see the usage data in parqueted format.

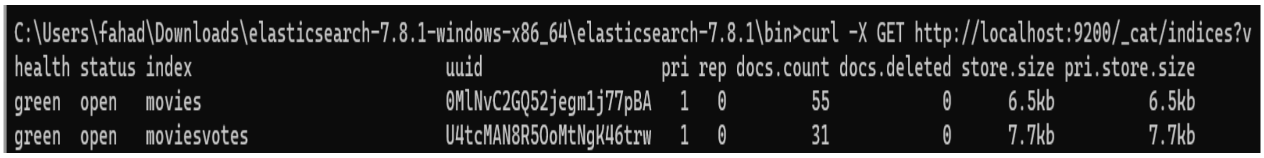

File Edit View Navigate Code Refactor Run Tools VCS Window Help
airflow - ...\dags\datalake\usage\averageRating_year\20230607\res.snappy.parquet\part-00000-7545b74a-c6de-4376-9f78-439794a41400-c000.snappy.parquet
airflow 〉 dags 〉 datalake 〉 usage 〉 averageRating_year 〉 20230607 〉 res.snappy.parquet 〉 part-00000-7545b74a-c6de-4376-9f78-439794a41400...
Project
source_to_raw_1.py
source_to_raw_2.py
my_first_dag.py
index_to_elastic.py
part-00000-7545b74a-c6de-4376-9f78-439794a41400-c000.snappy.parquet
raw
imdb
20230607
title.ratings.tsv.gz
rapidapi
20230607
extract.json
usage
averageRating_year
20230607
res.snappy.parquet
_SUCCESS.crc
part-00000-7545b74a-c6de-4376-9f78-43
_SUCCESS
part-00000-7545b74a-c6de-4376-9f78-43
numVotes_movies
20230607
res.snappy.parquet
_SUCCESS.crc
part-00000-14a61795-3f75-47d2-878b-1d
part-00001-14a61795-3f75-47d2-878b-1d
_SUCCESS
part-00000-14a61795-3f75-47d2-878b-1d
part-00001-14a61795-3f75-47d2-878b-1d
include
lib
site-packages
index_to_elastic.py
produce_usage.py
raw_to_formatted_1.py
raw_to_formatted_2.py
source_to_raw_1.py
source_to_raw_2.py
Scripts
Page Size: 50. (Displayed 1 - 50)
Dump
Data
Schema
Run
TODO
Problems
Terminal
Cap Console
Python Packages
Python Console
Event Log
Python 3.8
20°C
Ciel couvert
Search
ENG
IN
23:48
07-06-2023

	year	average_rating
1	1931	8.4
2	1936	8.5
3	1940	8.4
4	1941	8.3
5	1942	8.5
6	1946	8.6
7	1950	8.4
8	1952	8.3
9	1954	8.55
10	1957	8.6
11	1958	8.3
12	1959	8.3
13	1960	8.5
14	1962	8.45
15	1963	8.4
16	1964	8.4
17	1966	8.8
18	1968	8.4
19	1972	9.2
20	1974	9.0
21	1975	8.7
22	1977	8.6
23	1979	8.45
24	1980	8.55

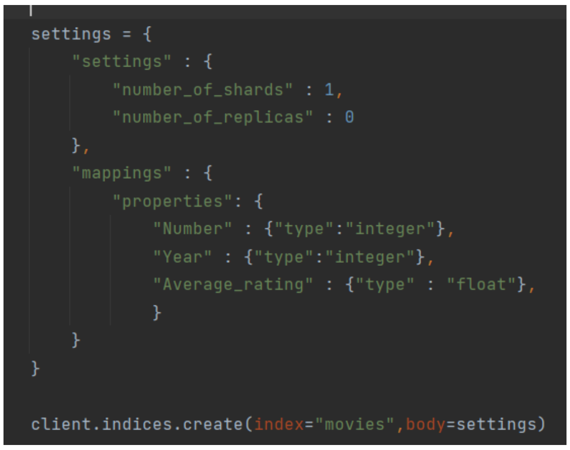

	numVotes	rating	title	year
1	1916942	8.8	The Lord of the Rings: The Fellowship of the Ring	2001
2	1888329	9.0	The Lord of the Rings: The Return of the King	2003
3	1704351	8.8	The Lord of the Rings: The Two Towers	2002
4	790594	8.6	Spirited Away	2001
5	856822	8.5	The Pianist	2002
6	771318	8.6	City of God	2002
7	1031905	8.3	Eternal Sunshine of the Spotless Mind	2004
8	1493072	8.3	Inglourious Basterds	2009
9	596849	8.4	Oldboy	2003
10	396829	8.4	The Lives of Others	2006
11	1357183	8.5	The Departed	2006
12	855670	8.3	Toy Story 3	2010
13	2720380	9.0	The Dark Knight	2008
14	1365423	8.5	The Prestige	2006
15	1912755	8.7	Interstellar	2014
16	1143724	8.4	WALL·E	2008
17	411100	8.4	3 Idiots	2009
18	1742748	8.4	The Dark Knight Rises	2012
19	2414782	8.8	Inception	2010
20	882169	8.5	The Intouchables	2011
21	1600619	8.4	Django Unchained	2012
22	341795	8.3	The Hunt	2012
23	535093	8.4	Coco	2017
24	905638	8.5	Whiplash	2014

Index_to_elastic:

After getting the final data from combing the data, We used kibana, elasticsearch to display the data in website. Initially we have to create a index in elasticsearch, we created index for both of the usages.

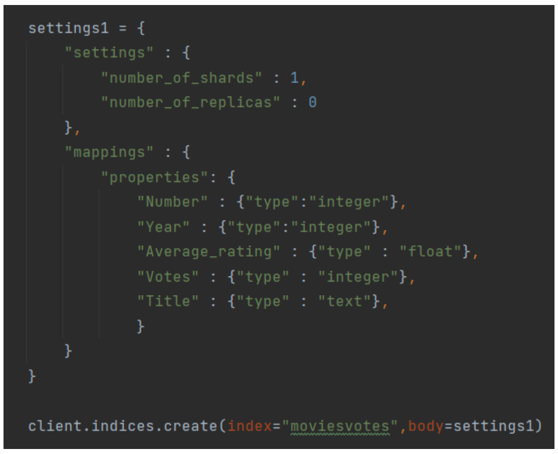

```
C:\Users\fahad\Downloads\elasticsearch-7.8.1-windows-x86_64\elasticsearch-7.8.1\bin>curl -X GET http://localhost:9200/_cat/indices?v
health status index                       uuid                   pri rep docs.count docs.deleted store.size pri.store.size
green  open   movies                      0MlNvC2GQ52jegm1j77pBA   1   0         55            0      6.5kb          6.5kb
green  open   moviesvotes                 U4tcMAN8R5OoMtNgK46trw   1   0         31            0      7.7kb          7.7kb
```

As you can see, we created two indexes, movies and moviesdata. And we updated the data in the elastic search.



```
settings = {
    "settings" : {
        "number_of_shards" : 1,
        "number_of_replicas" : 0
    },
    "mappings" : {
        "properties": {
            "Number" : {"type":"integer"},
            "Year" : {"type":"integer"},
            "Average_rating" : {"type" : "float"},
            }
    }
}

client.indices.create(index="movies",body=settings)
```

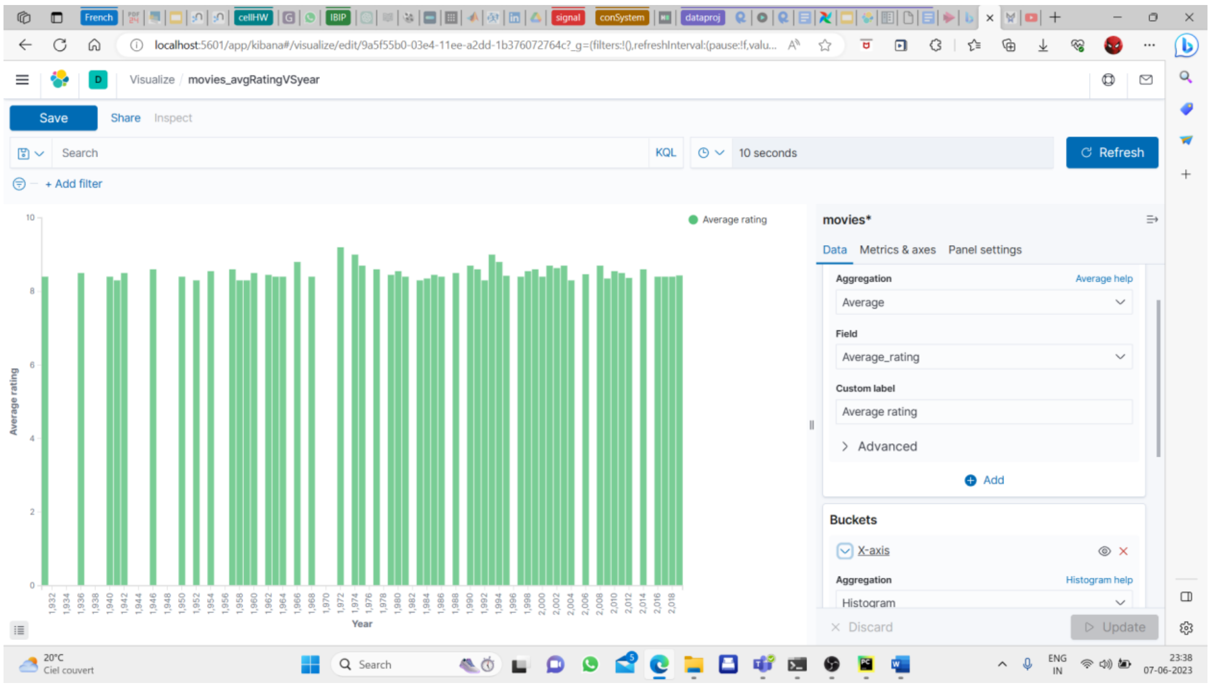


```
settings1 = {
    "settings" : {
        "number_of_shards" : 1,
        "number_of_replicas" : 0
    },
    "mappings" : {
        "properties": {
            "Number" : {"type":"integer"},
            "Year" : {"type":"integer"},
            "Average_rating" : {"type" : "float"},
            "Votes" : {"type" : "integer"},
            "Title" : {"type" : "text"},
            }
    }
}

client.indices.create(index="moviesvotes",body=settings1)
```

Elasticsearch and Kibana :

Usage 1 : To display the data in elasticsearch, we have to create a index and create a visualization, For the usage 1, we used vertical bar to display the data which shows average rating by year wise.



localhost:5601/app/kibana#/visualize/edit/9a5f55b0-03e4-11ee-a2dd-1b376072764c?_g=(filters:!(),refreshInterval:(pause:!f,valu...

Visualize / movies_avgRatingVSyear

Search KQL 10 seconds

movies*

Data Metrics & axes Panel settings

Aggregation: Average

Field: Average_rating

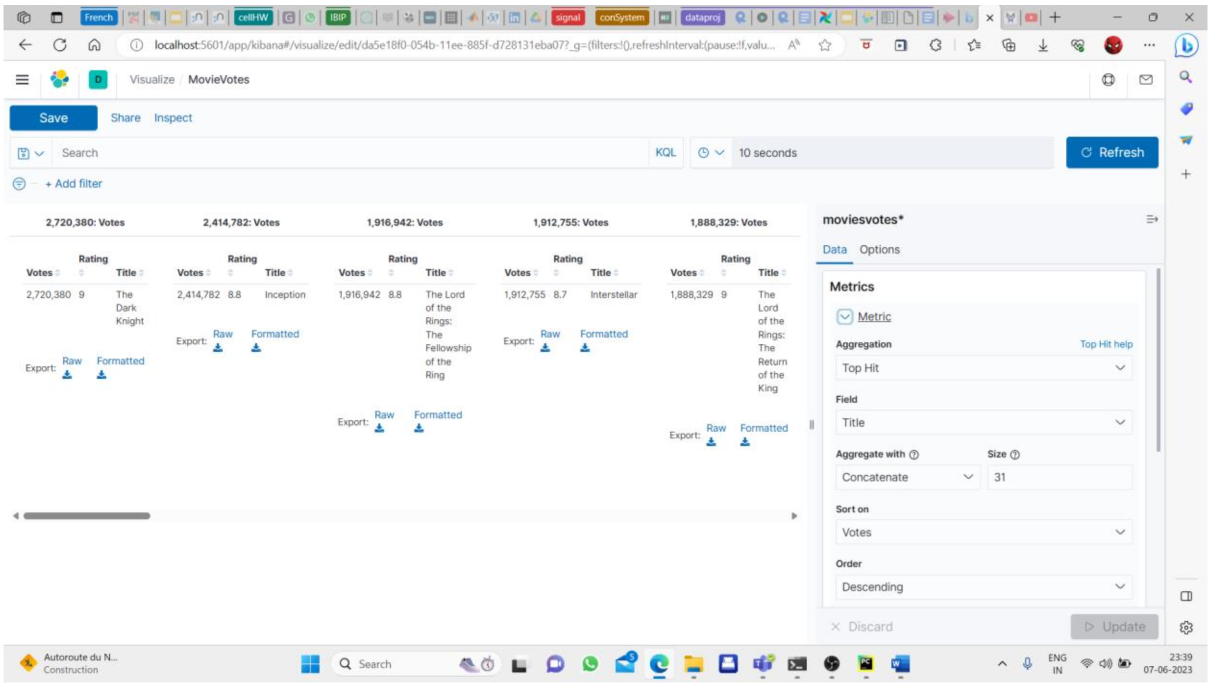
Custom label: Average rating

Buckets

X-axis

Aggregation: Histogram

Usage 2 : For the usage 2, we used data model, which displays the numvotes of the movies by descending order, we also displayed the rating with this result.


localhost:5601/app/kibana#/visualize/edit/da5e18f0-054b-11ee-885f-d728131eba07?_g=(filters:!(),refreshInterval:(pause:!f,valu...

Visualize / MovieVotes

Save Share Inspect

Search KQL 10 seconds Refresh

+ Add filter

2,720,380: Votes

Votes	Rating	Title
2,720,380	9	The Dark Knight

2,414,782: Votes

Votes	Rating	Title
2,414,782	8.8	Inception

1,916,942: Votes

Votes	Rating	Title
1,916,942	8.8	The Lord of the Rings: The Fellowship of the Ring

1,912,755: Votes

Votes	Rating	Title
1,912,755	8.7	Interstellar

1,888,329: Votes

Votes	Rating	Title
1,888,329	9	The Lord of the Rings: The Return of the King

Export: Raw Formatted

moviesvotes*

Data Options

Metrics

Metric

Aggregation: Top Hit

Field: Title

Aggregate with: Concatenate

Size: 31

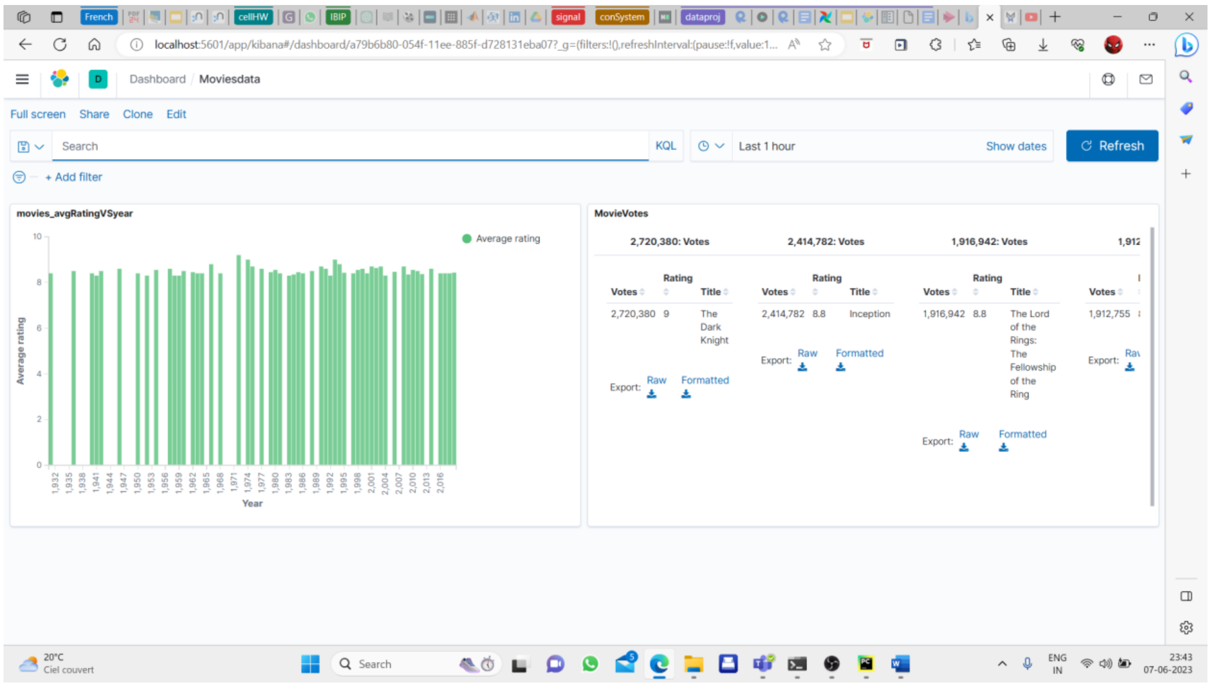
Sort on: Votes

Order: Descending

Discard Update

Elasticsearch Dashboard :

After getting this two data usages, we showed this data in the dashboard.



localhost:5601/app/kibana#/dashboard/a79b6b80-054f-11ee-885f-d728131eba07?_g=(filters:!(),refreshInterval:(pause:!f,value:1...

Dashboard / Moviesdata

Full screen Share Clone Edit

Search KQL Last 1 hour Show dates Refresh

+ Add filter

movies_avgRatingVSyear

MovieVotes

2,720,380: Votes

Votes	Rating	Title
2,720,380	9	The Dark Knight

2,414,782: Votes

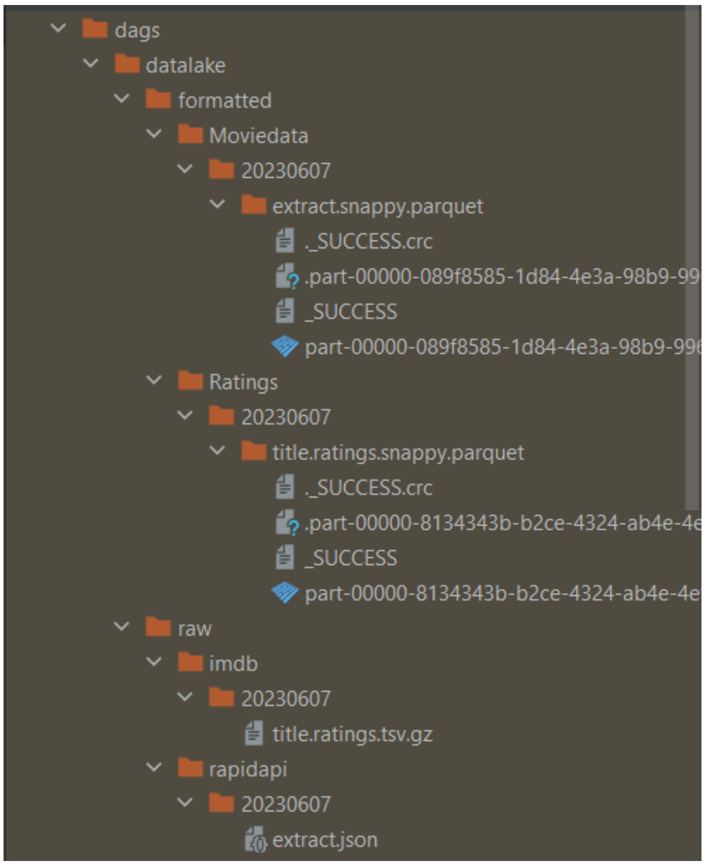
Votes	Rating	Title
2,414,782	8.8	Inception

1,916,942: Votes

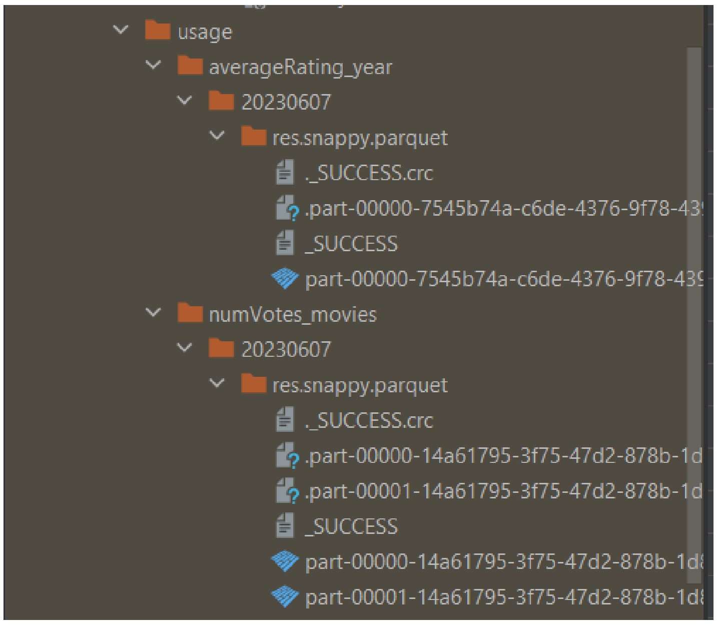
Votes	Rating	Title
1,916,942	8.8	The Lord of the Rings: The Fellowship of the Ring

Export: Raw Formatted

We have created specific folder to save the data.



```
dags
  datalake
    formatted
      Moviedata
        20230607
          extract.snappy.parquet
            ._SUCCESS.crc
            .part-00000-089f8585-1d84-4e3a-98b9-99
            _SUCCESS
            part-00000-089f8585-1d84-4e3a-98b9-99
      Ratings
        20230607
          title.ratings.snappy.parquet
            ._SUCCESS.crc
            .part-00000-8134343b-b2ce-4324-ab4e-4e
            _SUCCESS
            part-00000-8134343b-b2ce-4324-ab4e-4e
    raw
      imdb
        20230607
          title.ratings.tsv.gz
      rapidapi
        20230607
          extract.json
```



```
    usage
      averageRating_year
        20230607
          res.snappy.parquet
            ._SUCCESS.crc
            .part-00000-7545b74a-c6de-4376-9f78-43
            _SUCCESS
            part-00000-7545b74a-c6de-4376-9f78-43
      numVotes_movies
        20230607
          res.snappy.parquet
            ._SUCCESS.crc
            .part-00000-14a61795-3f75-47d2-878b-1d
            .part-00001-14a61795-3f75-47d2-878b-1d
            _SUCCESS
            part-00000-14a61795-3f75-47d2-878b-1d
            part-00001-14a61795-3f75-47d2-878b-1d
```