

**INSTITUT SUPÉRIEUR D'ÉLECTRONIQUE  
DE PARIS (ISEP)**



**II.2414 ADVANCED DATABASES AND  
BIG DATA**

**BIG DATA PROJECT  
(Data Lake and Data Pipeline)**

**BUHARI ALIYU  
ABUBAKAR UMAR ELNAFATY**

**PROF. THIBAUT DE BROCA**

**JUNE 2023**

## **Synopsis**

The main goal of the 'Big Data project' is to create a simple end-to-end data architecture that spans the entire data lifecycle, encompassing data ingestion, transformation, and exposition. Within this project, you have the freedom to choose the specific data you wish to retrieve and determine the desired output that will generate value from the input data.

However, it is essential to adhere to a Datalake architecture, as it plays a crucial role in organizing the data effectively, ensuring a streamlined data pipeline, and facilitating data sharing among stakeholders. By following the Datalake architecture principles, we can achieve a well-structured and efficient data ecosystem

### **Theme for the Data**

The central focus of this project revolves around gathering movies, weather data, and forecast information from multiple sources. The primary goal is to conduct a comprehensive analysis of the current top-rated, upcoming, and trending movies, considering their popularity, vote counts, release dates, and weather conditions. By incorporating weather forecast data, this analysis aims to provide valuable insights for individuals planning to visit a cinema.

By visualizing the cinema data alongside weather forecasts, stakeholders such as consumers, researchers, statisticians, and data scientists can make more informed decisions about their moviegoing experiences. The integration of weather information allows users to consider factors like temperature, precipitation, and other weather-related conditions that can impact their cinema experience.

For example, if someone prefers to enjoy outdoor activities before or after watching a movie, they can check the weather forecast to determine the best time to visit the cinema. If it's expected to rain heavily, they might opt for an indoor movie theater instead of an outdoor cinema. On the other hand, if the weather forecast indicates clear skies and pleasant temperatures, they may choose an outdoor cinema for a unique and enjoyable experience.

Moreover, the combination of cinema data and weather forecasts enables users to apply filters and preferences based on their specific moviegoing desires. They can filter movies based on genre, release date, popularity, and user ratings, and further refine their choices by considering weather conditions. For instance, individuals who prefer to watch horror movies during rainy days can easily identify suitable options by cross-referencing the cinema data with the weather forecast.

Furthermore, this enhanced visualization empowers individuals to plan their cinema outings more effectively by taking into account both movie-related factors and

weather conditions. It provides a comprehensive understanding of the movie landscape and allows users to make informed decisions, ensuring an enjoyable and tailored cinema experience

## Data Sources

The retrieved data would be gathered in the data lake as raw data. Data sources include:

1. The Movie Database TMDB API
2. Weather Api

A community-built movie and television database is called The Movie Database (TMDB) API. The documentation for the API is available at <https://developer.themoviedb.org/docs/getting-started>, and it includes instructions on how to connect to our application and fetch data via the API. We have fetched 3 different dataset from the api, the top rated movies, the upcoming movies and trending movies in the cinema. We saved the data as a json file. While the weather api documentation can be access via <https://www.weatherapi.com/docs>, we fetched the current weather and forecast weather of Paris

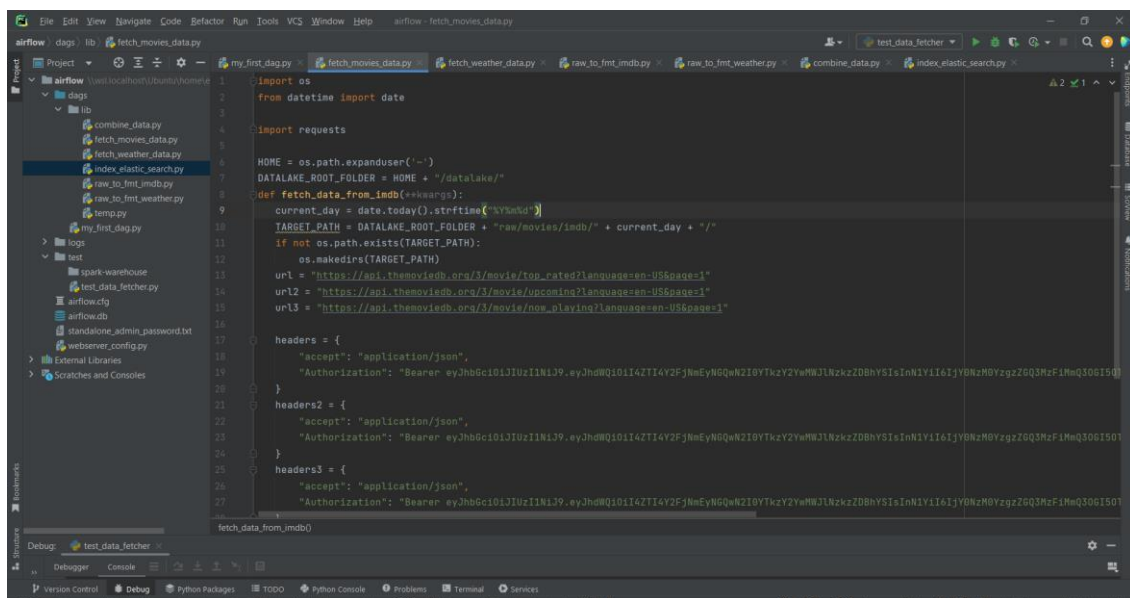


Figure 1 shows how to access the movie database API.

The Weather API serves as a valuable source of raw data for this project, providing real-time weather forecasts (for the next 7 days) and current weather information for Paris. By integrating this data into the project's data lake, stakeholders can explore the relationship between weather conditions and cinema trends, gaining insights into moviegoer behavior and enabling informed decision-making in Paris.

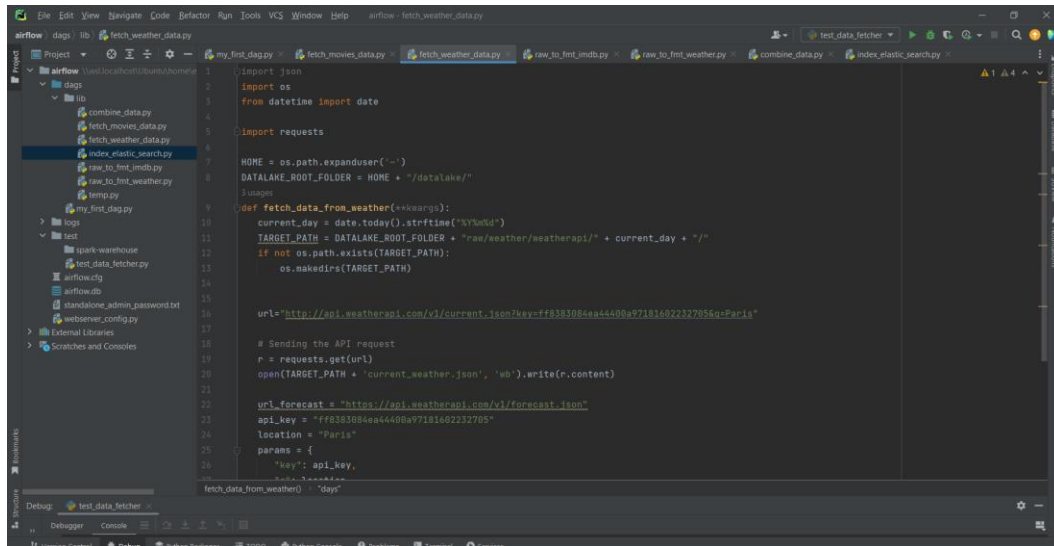


Figure 2: Connection to weather API

## Data Lake Organization

We designed our data lake with a structure that adheres to the following naming convention: Each route will look like this:

`/[{layer}]/[{group}]/[{TableName}]/[{date}]/filename`

**Layers** can include: - **raw** which is the first phase of the data and is straight from the source system, the data is prepared for use by the Datalake in the second stage, which is *formatted*, and *usage* is the final stage before

**Group:** we use *movies* to store movies file and *weather* to save weather files

**Table Name:** An item in this folder will always have this name and a similar schema, in raw we use *imdb* for the movies and *weatherapi* for weather.

**FileName:** top-rated\_movies.json, upcoming\_movies.json, trending\_movies.json, current\_weather.json and forecast.json are the file names for the raw data.

**Date:** should be formatted as YYYYMMDD, for instance, 20230611 (11th June 2022).

**For formatted file:** top-rated\_movies.snappy.parquet, trending\_movies.snappy.parquet, upcoming\_movies.snappy.parquet, current\_weather.snappy.parquet and forecast.snappy.parquet.

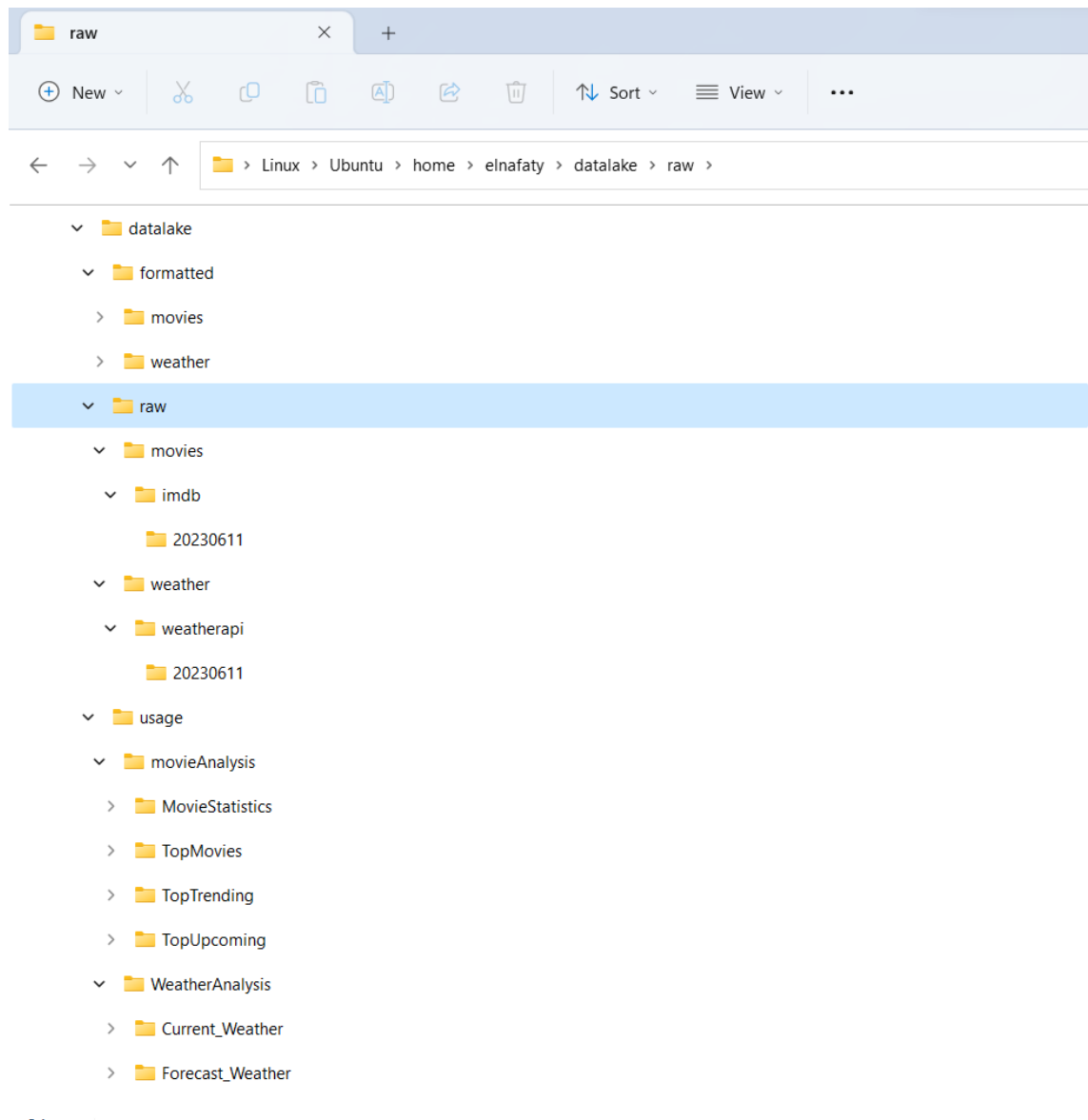


Figure 3: Datalake folder structure

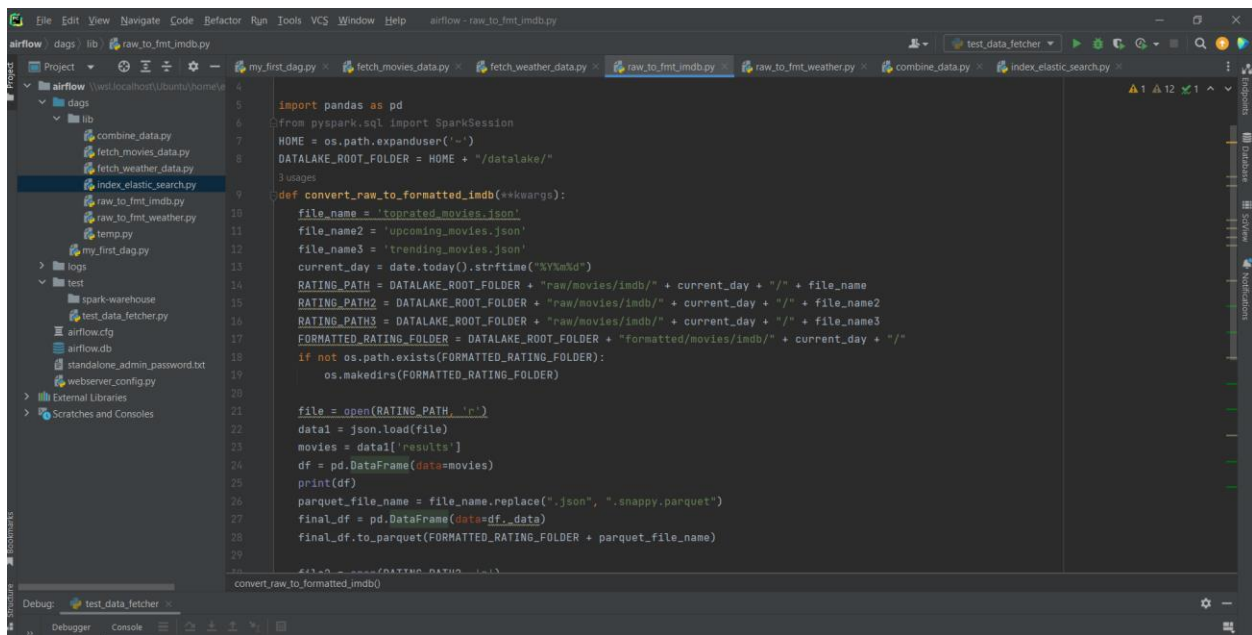
## Data Pipeline

The data pipeline plays a crucial role in transferring data from its source to a designated destination, such as a data warehouse. Throughout this journey, the data undergoes transformations and optimizations, ultimately arriving in a state that allows for analysis and the development of valuable business insights. The data pipeline encompasses the essential steps of aggregating, organizing, and moving data effectively.

## Data Pipeline Architecture

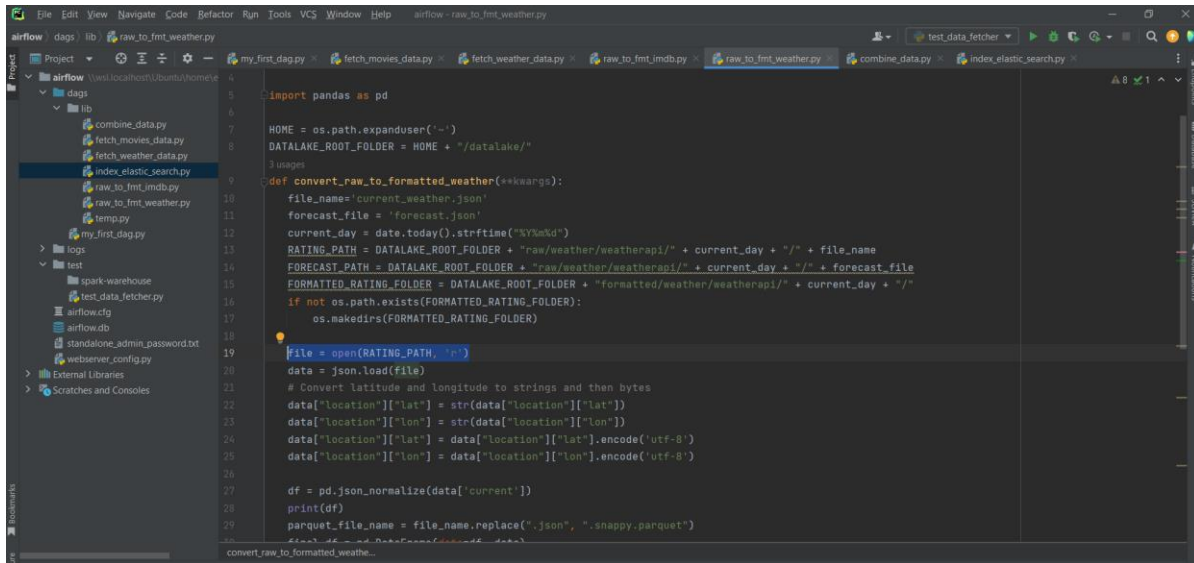
1. **Data Ingestion:** The source data is obtained through APIs. The data sources and ingestion process are detailed in Figure 1 and Figure 2 in the data source section. The collected data is then transferred to the raw folder in the data lake, where it can be further prepared for analysis. Python programming language is used for data ingestion in this project.
2. **Data Formatting:** In this step, the data is aggregated, cleansed, and manipulated to bring it to a standardized format suitable for analysis. Tools like Pandas are utilized to format the data like location, latitude and longitude for the weather. Figure 5 and Figure 6 illustrate the formatting process applied to data from different sources. The formatted data is then converted to the Parquet data format and compressed using snappy, which offers better processing and analysis capabilities.

By following these steps, the data is transformed and prepared in a way that optimizes its usability for subsequent analysis and visualization.



```
4 import pandas as pd
5 from pyspark.sql import SparkSession
6 HOME = os.path.expanduser('~')
7 DATALAKE_ROOT_FOLDER = HOME + "/datalake/"
8
9 def convert_raw_to_formatted_imdb(**kwargs):
10     file_name = 'toprated_movies.json'
11     file_name2 = 'upcoming_movies.json'
12     file_name3 = 'trending_movies.json'
13     current_day = date.today().strftime("%Y%m%d")
14     RATING_PATH = DATALAKE_ROOT_FOLDER + "raw/movies/imdb/" + current_day + "/" + file_name
15     RATING_PATH2 = DATALAKE_ROOT_FOLDER + "raw/movies/imdb/" + current_day + "/" + file_name2
16     RATING_PATH3 = DATALAKE_ROOT_FOLDER + "raw/movies/imdb/" + current_day + "/" + file_name3
17     FORMATTED_RATING_FOLDER = DATALAKE_ROOT_FOLDER + "formatted/movies/imdb/" + current_day + "/"
18     if not os.path.exists(FORMATTED_RATING_FOLDER):
19         os.makedirs(FORMATTED_RATING_FOLDER)
20
21     file = open(RATING_PATH, 'r')
22     data1 = json.load(file)
23     movies = data1['results']
24     df = pd.DataFrame(data=movies)
25     print(df)
26     parquet_file_name = file_name.replace(".json", ".snappy.parquet")
27     final_df = pd.DataFrame(data=df, data)
28     final_df.to_parquet(FORMATTED_RATING_FOLDER + parquet_file_name)
29
30 convert_raw_to_formatted_imdb()
```

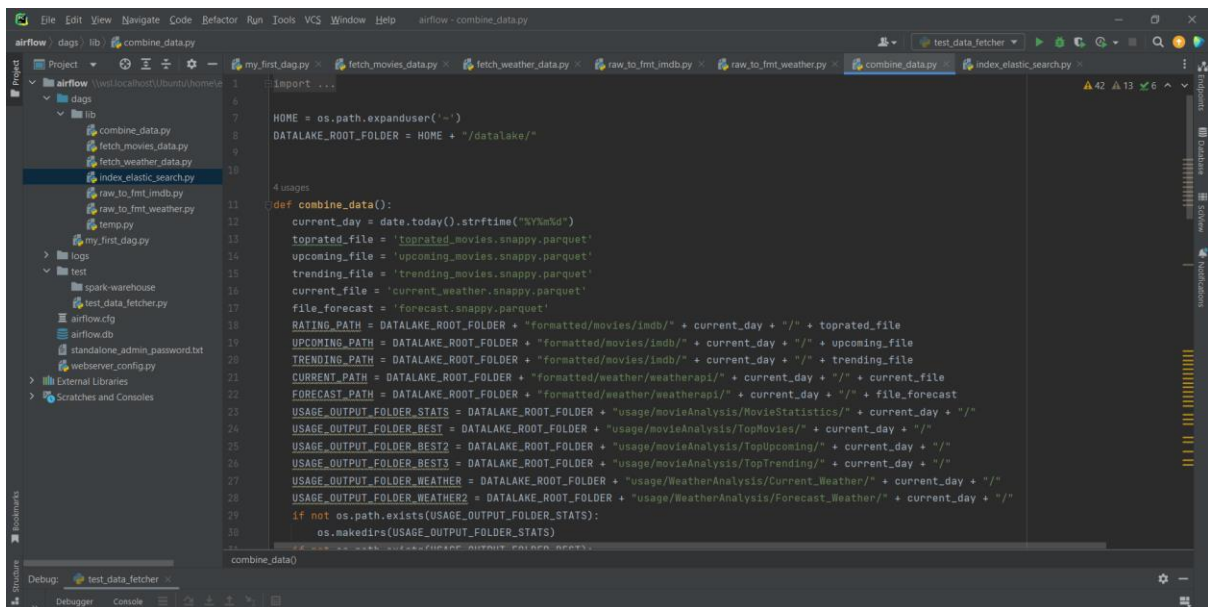
Figure 5: Formatting data from imdb with Panda



```
1 import pandas as pd
2
3 HOME = os.path.expanduser('~')
4 DATALAKE_ROOT_FOLDER = HOME + "/datalake/"
5
6
7 def convert_raw_to_formatted_weather(**kwargs):
8     file_name = 'current_weather.json'
9     forecast_file = 'forecast.json'
10     current_day = date.today().strftime("%Ym%d")
11     RATING_PATH = DATALAKE_ROOT_FOLDER + "raw/weather/weatherapi/" + current_day + "/" + file_name
12     FORECAST_PATH = DATALAKE_ROOT_FOLDER + "raw/weather/weatherapi/" + current_day + "/" + forecast_file
13     FORMATTED_RATING_FOLDER = DATALAKE_ROOT_FOLDER + "formatted/weather/weatherapi/" + current_day + "/"
14     if not os.path.exists(FORMATTED_RATING_FOLDER):
15         os.makedirs(FORMATTED_RATING_FOLDER)
16
17     file = open(RATING_PATH, 'r')
18     data = json.load(file)
19     # Convert latitude and longitude to strings and then bytes
20     data["location"]["lat"] = str(data["location"]["lat"])
21     data["location"]["lon"] = str(data["location"]["lon"])
22     data["location"]["lat"] = data["location"]["lat"].encode('utf-8')
23     data["location"]["lon"] = data["location"]["lon"].encode('utf-8')
24
25     df = pd.json_normalize(data['current'])
26     print(df)
27     parquet_file_name = file_name.replace(".json", ".snappy.parquet")
28     # Save the data to the formatted folder
29     df.to_parquet(FORMATTED_RATING_FOLDER + parquet_file_name)
```

Figure 6: Formatting weather data with Panda

- Data Combination:** Apache Spark is utilized in this step to combine the data from different data sources and performed sql analysis. This combined data becomes the foundation for further analysis. It is indexed and made accessible through the Elasticsearch and Kibana to visualize the data, we use the cloud version of the Elasticsearch and Kibana



```
1 import ...
2
3 HOME = os.path.expanduser('~')
4 DATALAKE_ROOT_FOLDER = HOME + "/datalake/"
5
6
7 def combine_data():
8     current_day = date.today().strftime("%Ym%d")
9     toprated_file = 'toprated_movies.snappy.parquet'
10     upcoming_file = 'upcoming_movies.snappy.parquet'
11     trending_file = 'trending_movies.snappy.parquet'
12     current_file = 'current_weather.snappy.parquet'
13     file_forecast = 'forecast.snappy.parquet'
14     RATING_PATH = DATALAKE_ROOT_FOLDER + "formatted/movies/imdb/" + current_day + "/" + toprated_file
15     UPCOMING_PATH = DATALAKE_ROOT_FOLDER + "formatted/movies/imdb/" + current_day + "/" + upcoming_file
16     TRENDING_PATH = DATALAKE_ROOT_FOLDER + "formatted/movies/imdb/" + current_day + "/" + trending_file
17     CURRENT_PATH = DATALAKE_ROOT_FOLDER + "formatted/weather/weatherapi/" + current_day + "/" + current_file
18     FORECAST_PATH = DATALAKE_ROOT_FOLDER + "formatted/weather/weatherapi/" + current_day + "/" + file_forecast
19     USAGE_OUTPUT_FOLDER_STATS = DATALAKE_ROOT_FOLDER + "usage/movieAnalysis/MovieStatistics/" + current_day + "/"
20     USAGE_OUTPUT_FOLDER_BEST1 = DATALAKE_ROOT_FOLDER + "usage/movieAnalysis/TopMovies/" + current_day + "/"
21     USAGE_OUTPUT_FOLDER_BEST2 = DATALAKE_ROOT_FOLDER + "usage/movieAnalysis/TopUpcoming/" + current_day + "/"
22     USAGE_OUTPUT_FOLDER_BEST3 = DATALAKE_ROOT_FOLDER + "usage/movieAnalysis/TopTrending/" + current_day + "/"
23     USAGE_OUTPUT_FOLDER_WEATHER = DATALAKE_ROOT_FOLDER + "usage/WeatherAnalysis/Current_Weather/" + current_day + "/"
24     USAGE_OUTPUT_FOLDER_WEATHER2 = DATALAKE_ROOT_FOLDER + "usage/WeatherAnalysis/Forecast_Weather/" + current_day + "/"
25     if not os.path.exists(USAGE_OUTPUT_FOLDER_STATS):
26         os.makedirs(USAGE_OUTPUT_FOLDER_STATS)
```

Figure 7: Data combination with Apache Spark



```

13 1
14 2
15 3
16 4
17 5
18 6
19 7
20 8
21 9
22 10
23 11
24 12
25 13
26 14
27 15
28 16
29 17
30 18
31 19
32 20
33 21
34 22
35 23
36 24
37 25
38 26
39 27
40 28
41 29
42 30
43 31
44 32
45 33
46 34
47 35
48 36
49 37
50 38
51 39
52 40
53 41
54 42
55 43
56 44
57 45
58 46
59 47
60 48
61 49
62 50
63 51
64 52
65 53
66 54
67 55
68 56
69 57
70 58
71 59
72 60
73 61
74 62
75 63
76 64
77 65
78 66
79 67
80 68
81 69
82 70
83 71
84 72
85 73
86 74
87 75
88 76
89 77
90 78
91 79
92 80
93 81
94 82
95 83
96 84
97 85
98 86
99 87
100 88
101 89
102 90
103 91
104 92
105 93
106 94
107 95
108 96
109 97
110 98
111 99
112 100
113 101
114 102
115 103
116 104
117 105
118 106
119 107
120 108
121 109
122 110
123 111
124 112
125 113
126 114
127 115
128 116
129 117
130 118
131 119
132 120
133 121
134 122
135 123
136 124
137 125
138 126
139 127
140 128
141 129
142 130
143 131
144 132
145 133
146 134
147 135
148 136
149 137
150 138
151 139
152 140
153 141
154 142
155 143
156 144
157 145
158 146
159 147
160 148
161 149
162 150
163 151
164 152
165 153
166 154
167 155
168 156
169 157
170 158
171 159
172 160
173 161
174 162
175 163
176 164
177 165
178 166
179 167
180 168
181 169
182 170
183 171
184 172
185 173
186 174
187 175
188 176
189 177
190 178
191 179
192 180
193 181
194 182
195 183
196 184
197 185
198 186
199 187
200 188
201 189
202 190
203 191
204 192
205 193
206 194
207 195
208 196
209 197
210 198
211 199
212 200
213 201
214 202
215 203
216 204
217 205
218 206
219 207
220 208
221 209
222 210
223 211
224 212
225 213
226 214
227 215
228 216
229 217
230 218
231 219
232 220
233 221
234 222
235 223
236 224
237 225
238 226
239 227
240 228
241 229
242 230
243 231
244 232
245 233
246 234
247 235
248 236
249 237
250 238
251 239
252 240
253 241
254 242
255 243
256 244
257 245
258 246
259 247
260 248
261 249
262 250
263 251
264 252
265 253
266 254
267 255
268 256
269 257
270 258
271 259
272 260
273 261
274 262
275 263
276 264
277 265
278 266
279 267
280 268
281 269
282 270
283 271
284 272
285 273
286 274
287 275
288 276
289 277
290 278
291 279
292 280
293 281
294 282
295 283
296 284
297 285
298 286
299 287
300 288
301 289
302 290
303 291
304 292
305 293
306 294
307 295
308 296
309 297
310 298
311 299
312 300
313 301
314 302
315 303
316 304
317 305
318 306
319 307
320 308
321 309
322 310
323 311
324 312
325 313
326 314
327 315
328 316
329 317
330 318
331 319
332 320
333 321
334 322
335 323
336 324
337 325
338 326
339 327
340 328
341 329
342 330
343 331
344 332
345 333
346 334
347 335
348 336
349 337
350 338
351 339
352 340
353 341
354 342
355 343
356 344
357 345
358 346
359 347
360 348
361 349
362 350
363 351
364 352
365 353
366 354
367 355
368 356
369 357
370 358
371 359
372 360
373 361
374 362
375 363
376 364
377 365
378 366
379 367
380 368
381 369
382 370
383 371
384 372
385 373
386 374
387 375
388 376
389 377
390 378
391 379
392 380
393 381
394 382
395 383
396 384
397 385
398 386
399 387
400 388
401 389
402 390
403 391
404 392
405 393
406 394
407 395
408 396
409 397
410 398
411 399
412 400
413 401
414 402
415 403
416 404
417 405
418 406
419 407
420 408
421 409
422 410
423 411
424 412
425 413
426 414
427 415
428 416
429 417
430 418
431 419
432 420
433 421
434 422
435 423
436 424
437 425
438 426
439 427
440 428
441 429
442 430
443 431
444 432
445 433
446 434
447 435
448 436
449 437
450 438
451 439
452 440
453 441
454 442
455 443
456 444
457 445
458 446
459 447
460 448
461 449
462 450
463 451
464 452
465 453
466 454
467 455
468 456
469 457
470 458
471 459
472 460
473 461
474 462
475 463
476 464
477 465
478 466
479 467
480 468
481 469
482 470
483 471
484 472
485 473
486 474
487 475
488 476
489 477
490 478
491 479
492 480
493 481
494 482
495 483
496 484
497 485
498 486
499 487
500 488
501 489
502 490
503 491
504 492
505 493
506 494
507 495
508 496
509 497
510 498
511 499
512 500
513 501
514 502
515 503
516 504
517 505
518 506
519 507
520 508
521 509
522 510
523 511
524 512
525 513
526 514
527 515
528 516
529 517
530 518
531 519
532 520
533 521
534 522
535 523
536 524
537 525
538 526
539 527
540 528
541 529
542 530
543 531
544 532
545 533
546 534
547 535
548 536
549 537
550 538
551 539
552 540
553 541
554 542
555 543
556 544
557 545
558 546
559 547
560 548
561 549
562 550
563 551
564 552
565 553
566 554
567 555
568 556
569 557
570 558
571 559
572 560
573 561
574 562
575 563
576 564
577 565
578 566
579 567
580 568
581 569
582 570
583 571
584 572
585 573
586 574
587 575
588 576
589 577
590 578
591 579
592 580
593 581
594 582
595 583
596 584
597 585
598 586
599 587
600 588
601 589
602 590
603 591
604 592
605 593
606 594
607 595
608 596
609 597
610 598
611 599
612 600
613 601
614 602
615 603
616 604
617 605
618 606
619 607
620 608
621 609
622 610
623 611
624 612
625 613
626 614
627 615
628 616
629 617
630 618
631 619
632 620
633 621
634 622
635 623
636 624
637 625
638 626
639 627
640 628
641 629
642 630
643 631
644 632
645 633
646 634
647 635
648 636
649 637
650 638
651 639
652 640
653 641
654 642
655 643
656 644
657 645
658 646
659 647
660 648
661 649
662 650
663 651
664 652
665 653
666 654
667 655
668 656
669 657
670 658
671 659
672 660
673 661
674 662
675 663
676 664
677 665
678 666
679 667
680 668
681 669
682 670
683 671
684 672
685 673
686 674
687 675
688 676
689 677
690 678
691 679
692 680
693 681
694 682
695 683
696 684
697 685
698 686
699 687
700 688
701 689
702 690
703 691
704 692
705 693
706 694
707 695
708 696
709 697
710 698
711 699
712 700
713 701
714 702
715 703
716 704
717 705
718 706
719 707
720 708
721 709
722 710
723 711
724 712
725 713
726 714
727 715
728 716
729 717
730 718
731 719
732 720
733 721
734 722
735 723
736 724
737 725
738 726
739 727
740 728
741 729
742 730
743 731
744 732
745 733
746 734
747 735
748 736
749 737
750 738
751 739
752 740
753 741
754 742
755 743
756 744
757 745
758 746
759 747
760 748
761 749
762 750
763 751
764 752
765 753
766 754
767 755
768 756
769 757
770 758
771 759
772 760
773 761
774 762
775 763
776 764
777 765
778 766
779 767
780 768
781 769
782 770
783 771
784 772
785 773
786 774
787 775
788 776
789 777
790 778
791 779
792 780
793 781
794 782
795 783
796 784
797 785
798 786
799 787
800 788
801 789
802 790
803 791
804 792
805 793
806 794
807 795
808 796
809 797
810 798
811 799
812 800
813 801
814 802
815 803
816 804
817 805
818 806
819 807
820 808
821 809
822 810
823 811
824 812
825 813
826 814
827 815
828 816
829 817
830 818
831 819
832 820
833 821
834 822
835 823
836 824
837 825
838 826
839 827
840 828
841 829
842 830
843 831
844 832
845 833
846 834
847 835
848 836
849 837
850 838
851 839
852 840
853 841
854 842
855 843
856 844
857 845
858 846
859 847
860 848
861 849
862 850
863 851
864 852
865 853
866 854
867 855
868 856
869 857
870 858
871 859
872 860
873 861
874 862
875 863
876 864
877 865
878 866
879 867
880 868
881 869
882 870
883 871
884 872
885 873
886 874
887 875
888 876
889 877
890 878
891 879
892 880
893 881
894 882
895 883
896 884
897 885
898 886
899 887
900 888
901 889
902 890
903 891
904 892
905 893
906 894
907 895
908 896
909 897
910 898
911 899
912 900
913 901
914 902
915 903
916 904
917 905
918 906
919 907
920 908
921 909
922 910
923 911
924 912
925 913
926 914
927 915
928 916
929 917
930 918
931 919
932 920
933 921
934 922
935 923
936 924
937 925
938 926
939 927
940 928
941 929
942 930
943 931
944 932
945 933
946 934
947 935
948 936
949 937
950 938
951 939
952 940
953 941
954 942
955 943
956 944
957 945
958 946
959 947
960 948
961 949
962 950
963 951
964 952
965 953
966 954
967 955
968 956
969 957
970 958
971 959
972 960
973 961
974 962
975 963
976 964
977 965
978 966
979 967
980 968
981 969
982 970
983 971
984 972
985 973
986 974
987 975
988 976
989 977
990 978
991 979
992 980
993 981
994 982
995 983
996 984
997 985
998 986
999 987
1000 988

```

Figure 8: Analysis

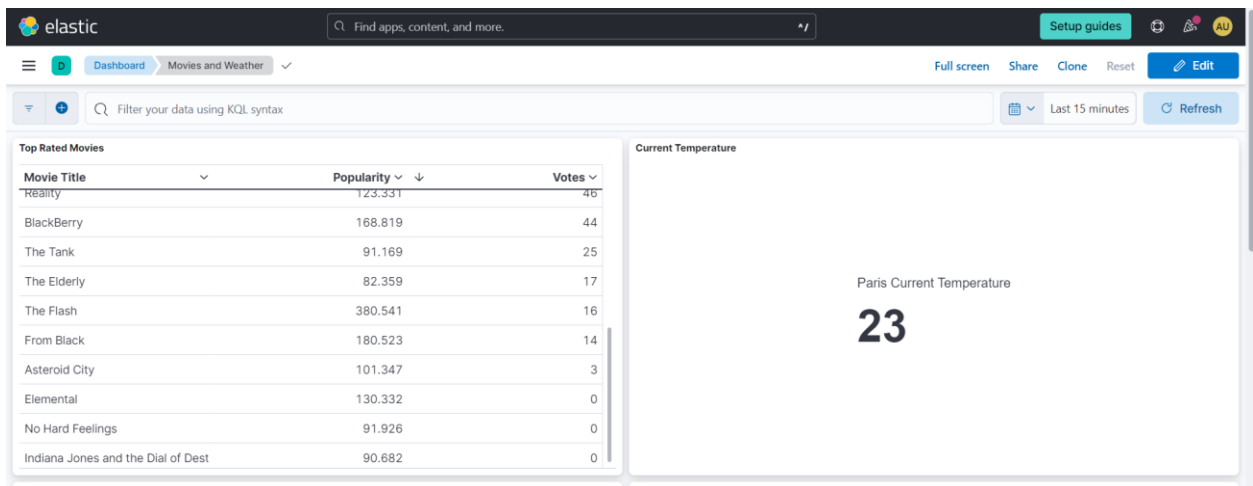
4. **Indexing:** In this project, the Elasticsearch Python library is utilized to facilitate the indexing process. The data is efficiently indexed and organized, enabling easy retrieval and exploration through Elasticsearch and visualization using Kibana.

```

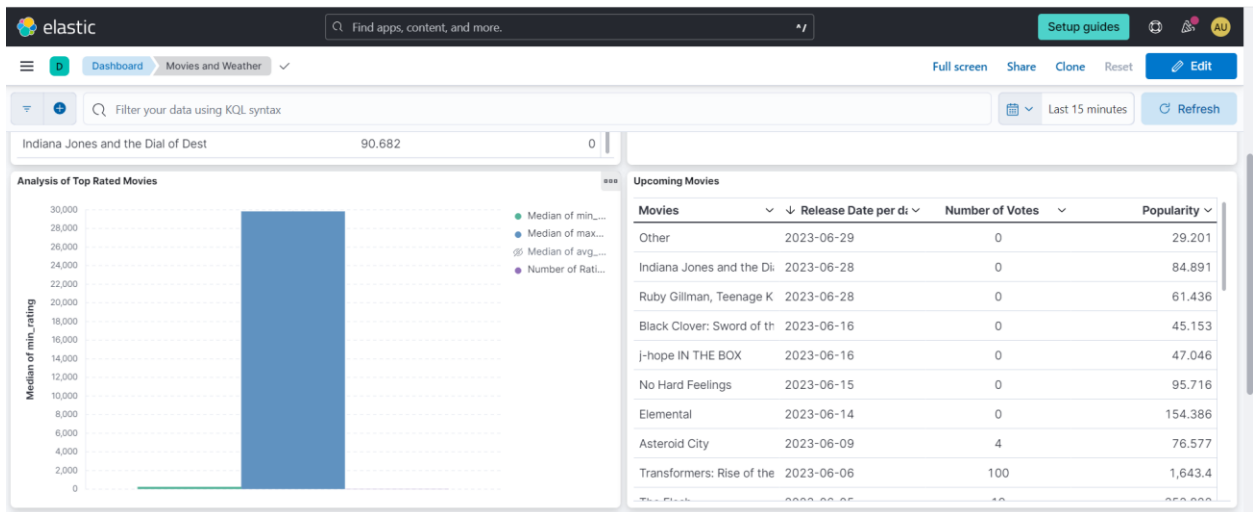
1 1
2 2
3 3
4 4
5 5
6 6
7 7
8 8
9 9
10 10
11 11
12 12
13 13
14 14
15 15
16 16
17 17
18 18
19 19
20 20
21 21
22 22
23 23
24 24
25 25
26 26
27 27
28 28
29 29
30 30
31 31
32 32
33 33
34 34
35 35
36 36
37 37
38 38
39 39
40 40
41 41
42 42
43 43
44 44
45 45
46 46
47 47
48 48
49 49
50 50
51 51
52 52
53 53
54 54
55 55
56 56
57 57
58 58
59 59
60 60
61 61
62 62
63 63
64 64
65 65
66 66
67 67
68 68
69 69
70 70
71 71
72 72
73 73
74 74
75 75
76 76
77 77
78 78
79 79
80 80
81 81
82 82
83 83
84 84
85 85
86 86
87 87
88 88
89 89
90 90
91 91
92 92
93 93
94 94
95 95
96 96
97 97
98 98
99 99
100 100
101 101
102 102
103 103
104 104
105 105
106 106
107 107
108 108
109 109
110 110
111 111
112 112
113 113
114 114
115 115
116 116
117 117
118 118
119 119
120 120
121 121
122 122
123 123
124 124
125 125
126 126
127 127
128 128
129 129
130 130
131 131
132 132
133 133
134 134
135 135
136 136
137 137
138 138
139 139
140 140
141 141
142 142
143 143
144 144
145 145
146 146
147 147
148 148
149 149
150 150
151 151
152 152
153 153
154 154
155 155
156 156
157 157
158 158
159 159
160 160
161 161
162 162
163 163
164 164
165 165
166 166
167 167
168 168
169 169
170 170
171 171
172 172
173 173
174 174
175 175
176 176
177 177
178 178
179 179
180 180
181 181
182 182
183 183
184 184
185 185
186 186
187 187
188 188
189 189
190 190
191 191
192 192
193 193
194 194
195 195
196 196
197 197
198 198
199 199
200 200
201 201
202 202
203 203
204 204
205 205
206 206
207 207
208 208
209 209
210 210
211 211
212 212
213 213
214 214
215 215
216 216
217 217
218 218
219 219
220 220
221 221
222 222
223 223
224 224
225 225
226 226
227 227
228 228
229 229
230 230
231 231
232 232
233 233
234 234
235 235
236 236
237 237
238 238
239 239
240 240
241 241
242 242
243 243
244 244
245 245
246 246
247 247
248 248
249 249
250 250
251 251
252 252
253 253
254 254
255 255
256 256
257 257
258 258
259 259
260 260
261 261
262 262
263 263
264 264
265 265
266 266
267 267
268 268
269 269
270 270
271 271
272 272
273 273
274 274
275 275
276 276
277 277
278 278
279 279
280 280
281 281
282 282
283 283
284 284
285 285
286 286
287 287
288 288
289 289
290 290
291 291
292 292
293 293
294 294
295 295
296 296
297 297
298 298
299 299
300 300
301 301
302 302
303 303
304 304
305 305
306 306
307 307
308 308
309 309
310 310
311 311
312 312
313 313
314 314
315 315
316 316
317 317
318 318
319 319
320 320
321 321
322 322
323 323
324 324
325 325
326 326
327 327
328 328
329 329
330 330
331 331
332 332
333 333
334 334
335 335
336 336
337 337
338 338
339 339
340 340
341 341
342 342
343 343
344 344
345 345
346 346
347 347
348 348
349 349
350 350
351 351
352 352
353 353
354 354
355 355
356 356
357 357
358 358
359 359
360 360
361 361
362 362
363 363
364 364
365 365
366 366
367 367
368 368
369 369
370 370
371 371
372 372
373 373
374 374
375 375
376 376
377 377
378 378
379 379
380 380
381 381
382 382
383 383
384 384
385 385
386 386
387 387
388 388
389 389
390 390
391 391
392 392
393 393
394 394
395 395
396 396
397 397
398 398
399 399
400 400
401 401
402 402
403 403
404 404
405 405
406 406
407 407
408 408
409 409
410 410
411 411
412 412
413 413
414 414
415 415
416 416
417 417
418 418
419 419
420 420
421 421
422 422
423 423
424 424
425 425
426 426
427 427
428 428
429 429
430 430
431 431
432 432
433 433
434 434
435 435
436 436
437 437
438 438
439 439
440 440
441 441
442 442
443 443
444 444
445 445
446 446
447 447
448 448
449 449
450 450
451 451
452 452
453 453
454 454
455 455
456 456
457 457
458 458
459 459
460 460
461 461
462 462
463 463
464 464
465 465
466 466
467 467
468 468
469 469
470 470
471 471
472 472
473 473
474 474
475 475
476 476
477 477
478 478
479 479
480 480
481 481
482 482
483 483
484 484
485 485
486 486
487 487
488 488
489 489
490 490
491 491
492 492
493 493
494 494
495 495
496 496
497 497
498 498
499 499
500 500
501 501
502 502
503 503
504 504
505 505
506 506
507 507
508 508
509 509
510 510
511 511
512 512
513 513
514 514
515 515
516 516
517 517
518 518
519 519
520 520
521 521
522 522
523 523
524 524
525 525
526 526
527 527
528 528
529 529
530 530
531 531
532 532
533 533
534 534
535 535
536 536
537 537
538 538
539 539
540 540
541 541
542 542
543 543
544 544
545 545
546 546
547 547
548 548
549 549
550 550
551 551
552 552
553 553
554 554
555 555
556 556
557 557
558 558
559 559
560 560
561 561
562 562
563 563
564 564
565 565
566 566
567 567
568 568
5
```



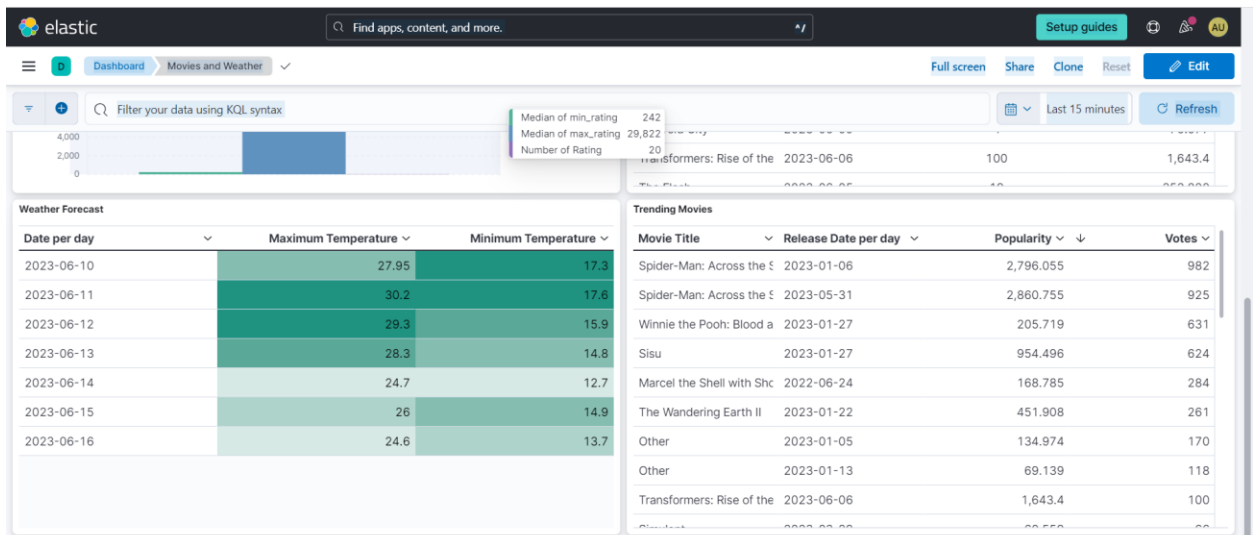




Top Rated Movies & Current Temperature



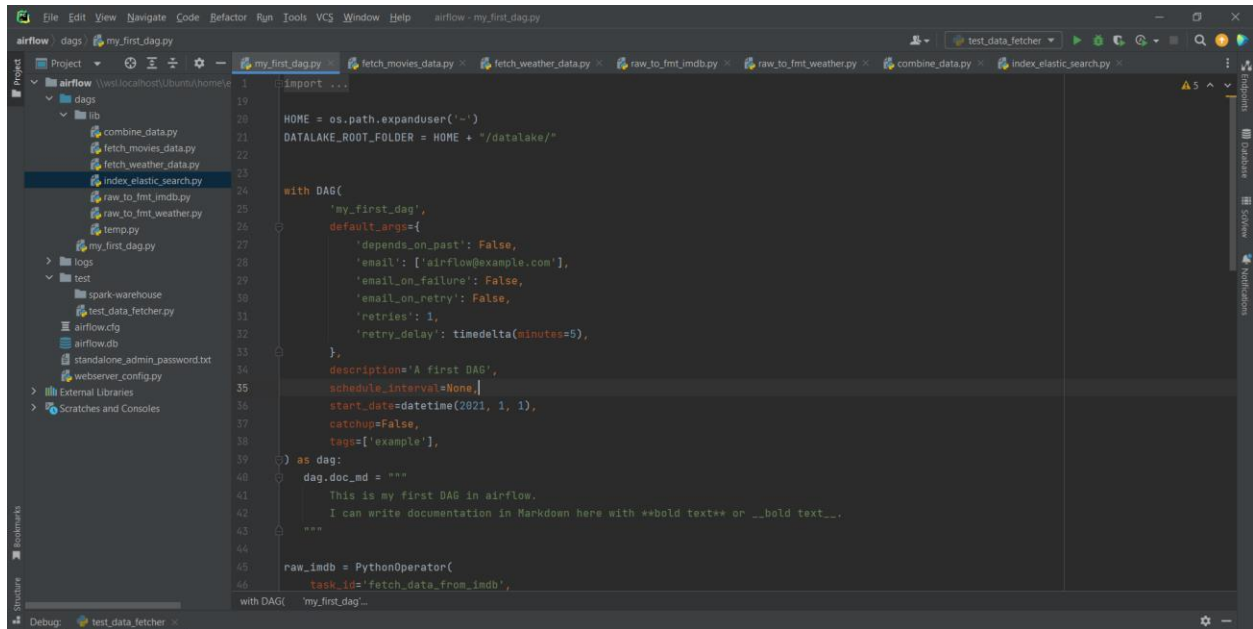
Analysis of Top Rated Movies & Upcoming Movies



Weather Forecast & Trending Movies

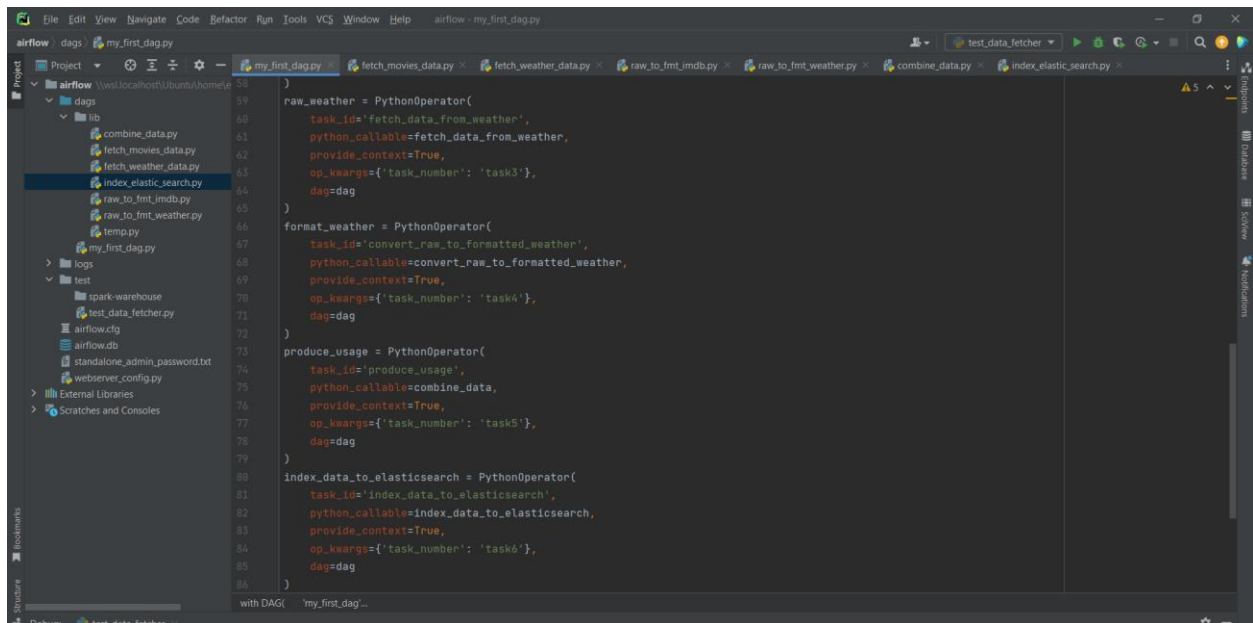
## Directed Acyclic Graphs - DAGs

We use Apache Airflow to create DAGs (Directed Acyclic Graphs) to manage workflow orchestration of our data. The tasks and dependencies are defined in python (Figure 10 and 11) and Airflow manages the scheduling and execution. The DAG that reflects our pipeline architecture is shown in Figure 12.



```
1 import os
2
3 HOME = os.path.expanduser('~')
4 DATALAKE_ROOT_FOLDER = HOME + "/datalake/"
5
6
7 with DAG(
8     'my_first_dag',
9     default_args={
10         'depends_on_past': False,
11         'email': ['airflow@example.com'],
12         'email_on_failure': False,
13         'email_on_retry': False,
14         'retries': 1,
15         'retry_delay': timedelta(minutes=5),
16     },
17     description='A first DAG',
18     schedule_interval=None,
19     start_date=datetime(2021, 1, 1),
20     catchup=False,
21     tags=['example'],
22 ) as dag:
23     dag.doc_md = """
24     This is my first DAG in airflow.
25     I can write documentation in Markdown here with **bold text** or __bold text___.
26     """
27
28     raw_imdb = PythonOperator(
29         task_id='fetch_data_from_imdb',
30         python_callable=fetch_data_from_imdb,
31     )
32
33     with DAG(
34         'my_first_dag'...
```

Figure 10: DAG Python Code i



```
58
59
60 raw_weather = PythonOperator(
61     task_id='fetch_data_from_weather',
62     python_callable=fetch_data_from_weather,
63     provide_context=True,
64     op_kwargs={'task_number': 'task3'},
65     dag=dag
66 )
67
68 format_weather = PythonOperator(
69     task_id='convert_raw_to_formatted_weather',
70     python_callable=convert_raw_to_formatted_weather,
71     provide_context=True,
72     op_kwargs={'task_number': 'task4'},
73     dag=dag
74 )
75
76 produce_usage = PythonOperator(
77     task_id='produce_usage',
78     python_callable=combine_data,
79     provide_context=True,
80     op_kwargs={'task_number': 'task5'},
81     dag=dag
82 )
83
84 index_data_to_elasticsearch = PythonOperator(
85     task_id='index_data_to_elasticsearch',
86     python_callable=index_data_to_elasticsearch,
87     provide_context=True,
88     op_kwargs={'task_number': 'task6'},
89     dag=dag
90 )
91
92 with DAG(
93     'my_first_dag'...
```

Figure 11: DAG Python Code ii

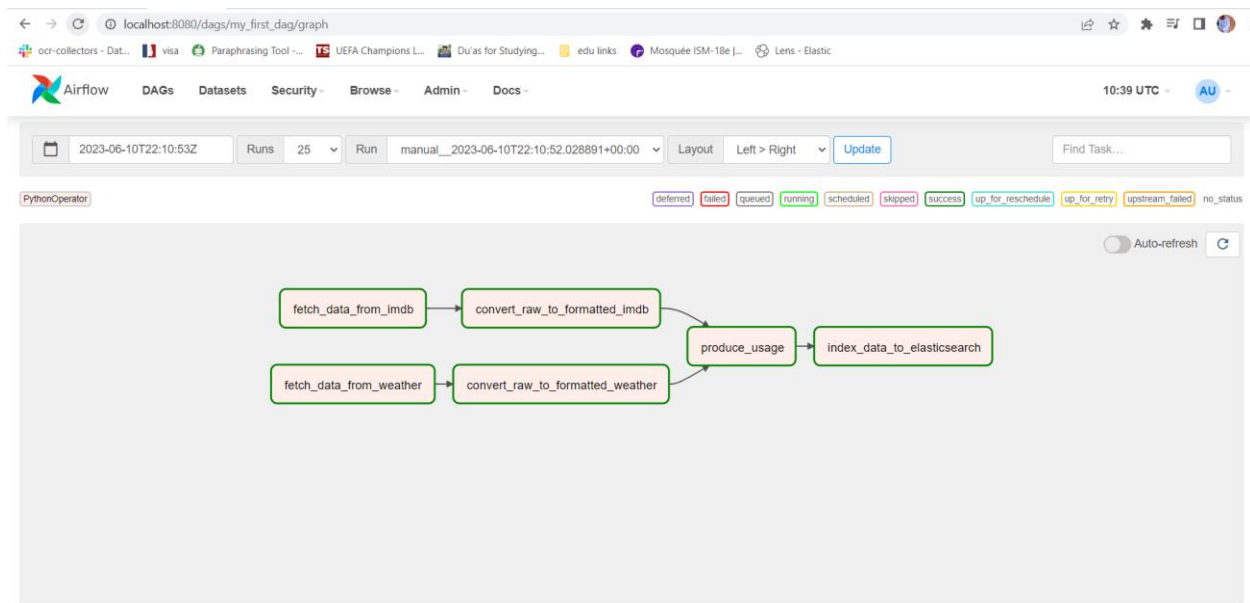


Figure 12 : DAG Airflow Graph View

## Conclusion

Throughout the implementation of this project, we have gained extensive knowledge and valuable experience in various technologies. Despite facing challenges along the way, the outcome has been highly successful, providing us with in-depth learnings and honing our skills.

The dynamic and comprehensive visualization of cinema and weather data offers significant benefits to consumers, researchers, statisticians, data scientists, and others, empowering them to make well-informed decisions, perform comparisons, and apply filters based on their specific needs. By considering trending movies, upcoming releases, top-rated films, user ratings, and weather forecasts for the upcoming week, stakeholders can leverage these insights for powerful recommendations.

This project has allowed us to broaden our understanding of key concepts such as big data, data lakes, data transformation, visualization, streaming, data analytics, and workflow. We have become proficient in utilizing various tools and technologies, including Apache Airflow, Apache Spark, cloud-based Elasticsearch and Kibana, Python, Panda, Parquet, and JSON, among others.