Wydział Informatyki Politechniki Białostockiej Katedra Systemów Informacyjnych i Sieci Komputerowych Przedmiot obieralny: Zaawansowane bazy danych i hurtownie danych sem. 1, rok akademicki 2023/2024 STUDIA STACJONARNE 2 STOPNIA	**Sprawozdanie z 5 pracy domowej** Temat: **MongoDB** Data: 3.6.2024
Prowadząca: prof. dr hab. inż. Agnieszka Drużdżel	Grupa PS: 4 Autorzy: 1. Piotr Szumowski 2. Łukasz Janowicz

Praca domowa#5 MongoDB

Praca domowa została wykonana w python-ie przy użyciu biblioteki MongoClient.

```python
from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')
db = client['IMDB']
titleCollection = db['Title']
ratingCollection = db['Rating']
nameCollection = db['Name']
```

Stworzono instancje klienta, który łączy się z serwerem MongoDB.
Przypisano do zmiennej db połączenie z bazą danych IMDB.
Oraz przypisano do zmiennych 3 kolekcje Title, Rating oraz Name.

```python
from icecream import ic
```

Zaimportowanie biblioteki ułatwiającej debugowanie.

```python
def printDocList(docList):
    for doc in docList:
        print(doc)
```

Stworzenie prostej funkcji do wypisywania wielu dokumentów z listy.

Zadanie 1

Zadanie 1: Sprawdź liczbę dokumentów w kolekcjach Title/Rating/Name.

```
def zad1():
    titleCount = titleCollection.count_documents({})
    ratingCount = ratingCollection.count_documents({})
    nameCount = nameCollection.count_documents({})
    ic(titleCount)
    ic(ratingCount)
    ic(nameCount)
```

```
ic| titleCount: 8957802
ic| ratingCount: 1441992
ic| nameCount: 9396000
```

Zadanie 2

Zadanie 2: Wybierz 4 pierwsze dokumenty z kolekcji Title, które były wyprodukowane w roku 2020, są z kategorii filmów Romance, ich czas trwania jest większy niż 90 minut, ale nie przekracza 120 minut. Zwracane dokumenty powinny zawierać tytuł, rok produkcji, kategorię oraz czas trwania. Dane uporządkuj rosnąco wg tytułu filmu. Sprawdź również, ile dokumentów zwróciłoby zapytanie po wyłączeniu ograniczenia w postaci 4 pierwszych dokumentów. Wyszukując łańcuchy skorzystaj z operatora $regex.

```
def zad2():
    zad2Query = {'startYear': 2020, 'genres': {'$regex': 'Romance', '$options': 'i'}, 'runtimeMinutes': {'$gt': 90, '$lte': 120}}
    zad2Display = {'_id': 0, 'primaryTitle': 1, 'startYear': 1, 'genres': 1, 'runtimeMinutes': 1}
    zad2Results = titleCollection.find(zad2Query, zad2Display).sort('primaryTitle', 1).limit(4)
    printDocList(zad2Results)
    zad2Count = titleCollection.count_documents(zad2Query)
    ic(zad2Count)
```

```
{'primaryTitle': 419, 'startYear': 2020, 'runtimeMinutes': 111, 'genres': 'Drama,Romance,Thriller'}
{'primaryTitle': '#s_uchilishcha', 'startYear': 2020, 'runtimeMinutes': 95, 'genres': 'Drama,Romance'}
{'primaryTitle': '2 Hearts', 'startYear': 2020, 'runtimeMinutes': 101, 'genres': 'Drama,Romance'}
{'primaryTitle': '23 Walks', 'startYear': 2020, 'runtimeMinutes': 102, 'genres': 'Comedy,Drama,Romance'}
ic| zad2Count: 429
```

Zadanie 3

Zadanie 3: Sprawdź ile filmów różnego typu (pole titleType) było wyprodukowanych w roku 2000. Wynik zapytania powinien zwracać nazwę typu oraz liczbę filmów.

```
def zad3():
    zad3Query = [{'$match': {'startYear': 2000}},
                 {'$group': {'_id': '$titleType', 'count': {'$sum': 1}}},
                 {'$project': {'_id': 0, 'titleType': '$_id', 'count': 1}}]
    zad3Result = titleCollection.aggregate(zad3Query)
    printDocList(zad3Result)
```

```
{'count': 64631, 'titleType': 'tvEpisode'}
{'count': 583, 'titleType': 'tvSpecial'}
{'count': 110, 'titleType': 'tvShort'}
{'count': 2201, 'titleType': 'tvMovie'}
{'count': 326, 'titleType': 'tvMiniSeries'}
{'count': 4546, 'titleType': 'movie'}
{'count': 2605, 'titleType': 'tvSeries'}
{'count': 4752, 'titleType': 'video'}
{'count': 641, 'titleType': 'videoGame'}
{'count': 4381, 'titleType': 'short'}
```

Zadanie 4

Zadanie 4: W oparciu o kolekcje Title oraz Rating sprawdź średnią ocenę filmów dokumentalnych wprodukowanych w latach 2010-2012. Wyświetl tytuł filmu, rok produkcji oraz jego średnią ocenę. Dane uporządkuj malejąco wg średniej oceny.

- **sprawdź, ile takich dokumentów zwróci zapytanie**
- **wyświetl tylko 5 pierwszych dokumentów spełniających powyższe warunki**

```
def zad4a():
    zad4Query = {'genres': 'Documentary', 'startYear': {'$gte': 2010, '$lte': 2012}}
    zad4aResult = titleCollection.count_documents(zad4Query)
    ic(zad4aResult)
```

```
ic| zad4aResult: 38642
```

```python
def zad4b():
    zad4bQuery = [{'$match': {'genres': 'Documentary', 'startYear': {'$gte': 2010, '$lte': 2012}}},
                  {'$lookup': {'from': 'Rating', 'localField': 'tconst', 'foreignField': 'tconst', 'as': 'joinRating'}},
                  {'$project': {'_id': 0, 'primaryTitle': 1, 'startYear': 1, 'averageRating': '$joinRating.averageRating'}},
                  {'$sort': {'averageRating': -1}},
                  {'$limit': 5}]
    zad4bResult = list(titleCollection.aggregate(zad4bQuery))
    printDocList(zad4bResult)
```

```
{'primaryTitle': 'Tripping on Hallucinogenic Frogs', 'startYear': 2012, 'averageRating': [10.0]}
{'primaryTitle': 'Episode #8.62', 'startYear': 2010, 'averageRating': [10.0]}
{'primaryTitle': 'Episode #9.22', 'startYear': 2010, 'averageRating': [10.0]}
{'primaryTitle': 'Episode #9.21', 'startYear': 2010, 'averageRating': [10.0]}
{'primaryTitle': 'Episode #9.18', 'startYear': 2010, 'averageRating': [10.0]}
```

Zadanie 5

Zadanie 5: Utwórz indeks tekstowy dla pola primaryName w kolekcji Name. Następnie używając tego indeksu znajdź dokumenty opisujące osoby o nazwisku Fonda oraz Coppola. Przy wyszukiwaniu włącz opcję, która będzie uwzględniać wielkie/małe litery.

A. **Ile dokumentów zwraca zapytanie?**
B. **Wyświetl 5 pierwszych dokumentów (dokument powinien zwracać dwa pola: primaryName, primaryProfession).**

```python
def zad5():
    # nameCollection.drop_index('primaryName_text')
    nameCollection.create_index([('primaryName', 'text')])
    zad5Query = {'$text': {'$search': 'Fonda Coppola', '$caseSensitive': True}}
    zad5aResult = nameCollection.count_documents(zad5Query)
    ic(zad5aResult)
    zad5bResult = nameCollection.find(zad5Query, {'_id': 0, 'primaryName': 1, 'primaryProfession': 1}).limit(5)
    printDocList(zad5bResult)
```

```
ic| zad5aResult: 443
```

```
324\tMika Lyyra\t\\N\t\\N\tactor\ttt1578147\nnm3905325\tRichard Webb\t\\N\t\\N\tmiscellaneous
 tt2231461,tt2106361,tt8946378\nnm3905329\tRobert Spano\t\\N\t\\N\tactor\ttt1653203\nnm390533
 Chou\t\\N\t\\N\tmiscellaneous,producer\ttt1653663,tt1442064\nnm3905334\tBaxton Baylor\t\\N\t
 Atkins\t\\N\t\\N\t\\N\ttt0489594,tt0352048\nnm3905338\tJosue Laboucane\t\\N\t\\N\tactor\ttt1
 Mason\t\\N\t\\N\t\\N\ttt0215445\nnm3905344\tNoriko Suzuki\t\\N\t\\N\tanimation_department\tt
 Dickhuth\t\\N\t\\N\tactor\ttt1649987\nnm3905348\tOlivier Laroche\t\\N\t\\N\tactor\ttt0790602
 tt7686376\nnm3905353\tJacqueline Carson\t\\N\t\\N\tproducer\ttt1653066\nnm3905354\tKatalina
 Kondoh\t\\N\t\\N\t\\N\t\\N\nnm3905359\tMariano Gobbi\t1982\t\\N\tvisual_effects,stunts\ttt23
 Weathers\t\\N\t\\N\twriter,director,cinematographer\ttt2947042,tt1653087\nnm3905363\tPhilipp
 Passarelli\t\\N\t\\N\tactress\ttt1648032\nnm3905366\tMike Newton\t\\N\t\\N\t\\N\ttt0412116\n
 Hood\t\\N\t\\N\tactress\ttt1654075\nnm3905371\tShôko Yamamoto\t\\N\t\\N\tcamera_department\t
 Fasano\t\\N\t\\N\tactor\ttt1261065\nnm3905376\tMario Curiel\t\\N\t\\N\tdirector,writer,cinem
 Millet\t\\N\t\\N\tactor\ttt1790864\nnm3905381\tCarles Esquembre\t\\N\t\\N\tmiscellaneous\ttt
 script_department,sound_department\ttt1650483,tt2248948,tt1717582\nnm3905385\tLizzie Jacobs\
 Machado\t\\N\t\\N\tmusic_department\ttt0367975\nnm3905389\tKonstadina Stamelakou\t\\N\t\\N\t
 tt1717158\nnm3905392\tConrad Petraborg\t\\N\t\\N\tactor\ttt1652402\nnm3905393\tElles Nijkamp
```

```
Stouraiti\t\\N\t\\N\tactress\ttt28015596,tt1661225\nnm3923109\tZdrave Kostovski\t\\N\t\\N\tactor\ttt1604095,tt6803184,tt7767
tt1671719\nnm3923112\tPippa Forysthe\t\\N\t\\N\t\\N\t\\N\nnm3923113\tPerry Pazer\t\\N\t\\N\tactor\ttt0163929,tt1757689\nnm39
music_department\ttt1760967,tt2070597,tt0296402,tt14196028\nnm3923116\tEla Kopernikówna\t\\N\t\\N\tactress\ttt0046841\nnm392
Roy\t\\N\t\\N\tmiscellaneous\ttt2535882,tt1330058,tt1303901\nnm3923120\tRico Biriah\t\\N\t\\N\tmiscellaneous,publicist\ttt23
miscellaneous\ttt1657895\nnm3923125\tGeorge Thow\t1908\t1987\tmiscellaneous\ttt0047749\nnm3923126\tMiye Bennett\t\\N\t\\N\tp
Kim\t\\N\t\\N\tart_department\ttt1653696\nnm3923129\tJason Jay Edwards\t\\N\t\\N\tvisual_effects,producer\ttt0437086,tt20153
tt1650390\nnm3923132\tAleksandr Anoprikov\t\\N\t\\N\tactor,cinematographer\ttt4921370,tt4008272,tt14956152,tt9047564\nnm3923
Salus\t\\N\t\\N\tactress\ttt1650390\nnm3923135\tChien-Ming Chen\t\\N\t\\N\tmake_up_department\ttt1394317\nnm3923136\tJoaquin
Carter-Brown\t\\N\t\\N\tactor\ttt1351177\nnm3923139\tJoe Rizzo\t\\N\t\\N\t\\N\t\\N\nnm3923140\tChristina Domke\t\\N\t\\N\t\\
tt8216084\nnm3923143\tBlaskapelle der Freiwilligen Feuerwehr Butzbach\t\\N\t\\N\tmusic_department\ttt1661315\nnm3923144\tPau
Mestre\t\\N\t\\N\t\\N\t\\N\nnm3923148\tRoyston Snith\t\\N\t\\N\tart_department\ttt1131729\nnm3923149\tMatt Klein\t\\N\t\\N\t
Ho\t\\N\t\\N\twriter\ttt0092759\nnm3923152\tG. Adamidis\t\\N\t\\N\tactor\ttt1661224\nnm3923153\tAlan Mitchell\t\\N\t\\N\tact
camera_department\ttt2246869,tt2376386,tt1687903,tt4778350\nnm3923156\tMilan Srdic\t\\N\t\\N\tproducer\ttt0361447,tt13171304
tt0475591,tt1602585,tt4220190\nnm3923159\tAsta Musvicaite\t\\N\t\\N\tproducer,miscellaneous\ttt2777906,tt3339624,tt1657887\n
Reed\t\\N\t\\N\tactress\ttt1414055\nnm3923163\tDanny Farmer\t\\N\t\\N\t\\N\ttt0407423,tt0320802,tt1627023\nnm3923164\tLuis E
 Bittick\t\\N\t\\N\tactor,miscellaneous\ttt1694559,tt1737214\nnm3923168\tCourtney Smith\t\\N\t\\N\t\\N\ttt1016283\nnm3923169
tt0324888\nnm3923171\tBarb Merlotti\t\\N\t\\N\tart_director,art_department,production_designer\ttt2024404,tt2059193,tt162804
script_department\ttt1484102,tt2009580\nnm3923175\tKlaus Kieswald\t\\N\t\\N\tactor\ttt1661315\nnm3923176\tVadim Ermilov\t\\N
tt1442562\nnm3923179\tOrchester der Frankfurter Oper\t\\N\t\\N\tmusic_department\ttt1661315\nnm3923180\tXudong Du\t\\N\t\\N\
Li\t\\N\t\\N\tactress\ttt1368472\nnm3923183\tKaren Hall\t\\N\t\\N\t\\N\t\\N\nnm3923184\tBetty Carney\t\\N\t\\N\tactress\ttt1
tt5490310,tt0945531\nnm3923188\tBert Ferntheil\t\\N\t\\N\tsoundtrack\t\\N\nnm3923189\tMarc Zyla\t\\N\t\\N\t\\N\ttt1024827\nn
tt2038277\nnm3923192\tBrandon Crockett\t\\N\t\\N\tactor,producer,writer\ttt2551442,tt1670389,tt4391938,tt2443766\nnm3923193\
Leaf\t\\N\t\\N\t\\N\t\\N\nnm3923199\tGeneviève Saglio\t\\N\t\\N\tproduction_manager\ttt1661220\nnm3923200\tAndreas Tsatsaris
McCloughlin\t\\N\t\\N\tmiscellaneous\ttt0086189\nnm3923203\tNancy Martin\t\\N\t\\N\tproduction_designer\ttt0298833\nnm392320
Leiner\t\\N\t\\N\tcamera_department,visual_effects,miscellaneous\ttt1318514,tt6968614,tt5090568,tt6977338\nnm3923207\tDaniel
Malmqvist\t\\N\t\\N\tvisual_effects\ttt1034314,tt3616916,tt12844910,tt1450321\nnm3923211\tPaul Smith\t\\N\t\\N\tsound_depart
tt2066129\nnm3923214\tMike Paul\t\\N\t\\N\t\\N\t\\N\nnm3923215\tMichael Broman\t\\N\t\\N\tart_department\ttt0159270\nnm39232
tt8787366,tt3238990\nnm3923218\tHerr Pranger\t\\N\t\\N\tactor\ttt0928065\nnm3923219\tGeorge Dye\t\\N\t\\N\t\\N\t\\N\nnm39232
MacLean\t\\N\t\\N\tproducer,miscellaneous\ttt0798817,tt1189444\nnm3923225\tShannon Elizabeth\t\\N\t\\N\tset_decorator\ttt010
tt1114680\nnm3923230\tGraylen Cameron\t\\N\t\\N\tactor\ttt1392197\nnm3923231\tFernando j. Ruiz\t\\N\t\\N\tproducer\ttt204007
tt1139668,tt1833673,tt1152836\nnm3923235\tNicole George Flammia\t\\N\t\\N\t\\N\t\\N\nnm3923238\tChris Gomez\t\\N\t\\N\t\\N\t
cinematographer\ttt29926374,tt29926487,tt2193253,tt29926585\nnm3923242\tJessica Smith "J.Smith', 'primaryProfession': '\\N'}
{'primaryName': 'Victor Coppola', 'primaryProfession': 'camera_department,miscellaneous'}
```

Mimo iż wynik wydaje się być błędny to sprawdziłem przy MongoDB Shell z poleceniem:

```
IMDB> db.Name.find(
          { $text: { $search: "Fonda Coppola", $caseSensitive: true } },
          { primaryName: 1, primaryProfession: 1, _id: 0 }
      ).limit(5)
```

i zwraca również taki sam wynik, który sugerowałby popsutą strukturę danych:

```
>_MONGOSH
      'nm3860320\tChristopher Baer\t\\N\t\\N\tcamera_department\ttt1529240,tt0424297,tt1620444,tt0382295\n' +
      'nm3860321\tRoger Flatt\t\\N\t\\N\tactor\ttt1629396\n' +
      'nm3860322\tBill Hinds\t\\N\t\\N\tmusic_department\ttt0087925\n' +
      'nm3860323\tDebbie Garcia\t\\N\t\\N\tactress\ttt5454552,tt15887102,tt8723904,tt27925273\n' +
      'nm3860324\tBill Young\t\\N\t\\N\teditor,editorial_department\ttt1723634\n' +
      'nm3860325\tAshish S. Kulkarni\t\\N\t\\N\tproducer\ttt0353115,tt1629042,tt4625706\n' +
      'nm3860326\tPascale Chadenat\t\\N\t\\N\tcomposer\ttt0085639\n' +
      'nm3860327\tWolfgang Heilmann\t\\N\t\\N\tactor\ttt1629236\n' +
      'nm3860328\tEmily Timmer\t\\N\t\\N\tactress,camera_department\ttt1629412\n' +
      'nm3860329\tCourtland Thomas\t\\N\t\\N\tproducer,actor,camera_department\ttt1629447,tt2094962,tt3674938\n' +
      'nm3860330\tYavone Reese\t1981\t\\N\t\\N\t\\N\n' +
      'nm3860331\tDoug Radford\t\\N\t\\N\tcamera_department\ttt1661820,tt1069153\n' +
      'nm3860332\tKotaro Matsubara\t\\N\t\\N\tanimation_department,visual_effects,miscellaneous\ttt6741278,tt0802988,tt12415670\n' +
      'nm3860333\tConnie Hughes\t\\N\t\\N\tactress,music_department\ttt1621426,tt29417444\n' +
      'nm3860334\tJessie Sylvestre\t\\N\t\\N\tmiscellaneous\ttt6516738,tt1537186\n' +
      'nm3860335\tEmery Downes\t\\N\t\\N\tactor\ttt0115285\n' +
      'nm3860336\tTiana Möller\t\\N\t\\N\tart_department,art_director\ttt2112136,tt8435892,tt5700626,tt1830792\n' +
      'nm3860337\tVernon Wade Adams\t\\N\t\\N\tactor\ttt1629447\n' +
      'nm3860339\tIvo Vossen\t\\N\t\\N\ttransportation_department\ttt1529240\n' +
      'nm3860340\tEugene Ruffolo\t\\N\t\\N\tmusic_department,soundtrack\ttt0859163,tt0376136,tt1129445,tt0280350\n' +
      'nm3860341\tKazuo Harada\t\\N\t\\N\tactor\ttt9816816,tt6535334,tt1638339,tt0174699\n' +
      'nm3860342\tGerry Prince Young\t\\N\t\\N\twriter\ttt0052451\n' +
      'nm3860344\tWesley Sylvestre\t\\N\t\\N\t\\N\ttt1537186\n' +
      'nm3860345\tChalmers Davis\t\\N\t\\N\tmusic_department\ttt0087925\n' +
      'nm3860346\tMohamed Made Neto\t\\N\t\\N\ttransportation_department\ttt0420548\n' +
      'nm3860347\tKay Hurst\t\\N\t\\N\t\\N\t\\N\n' +
      'nm3860348\tMelissa Sussman\t\\N\t\\N\tactor,casting_department\ttt1464552,tt4563364,tt2301224\n' +
      'nm3860349\tDeborah Banker\t\\N\t\\N\tart_department\ttt0087925\n' +
      'nm3860350\tBrenda Archer\t\\N\t\\N\t\\N\ttt0462125\n' +
      'nm3860351\tRon Rose\t\\N\t\\N\tsound_department\ttt1936759,tt4789864,tt2301076,tt2288050\n' +
      'nm3860352\tM. Serano\t\\N\t\\N\tcostume_department\ttt0061646\n' +
      'nm3860355\tRichard Peterson\t\\N\t\\N\tsound_department\ttt0087925\n' +
      'nm3860356\tEmily Richardson\t\\N\t\\N\tdirector,camera_department\ttt3619706,tt1629342,tt9134476,tt9853502\n' +
      'nm3860359\tStefani L. Weiss\t1961\t\\N\tcamera_department,producer,make_up_department\ttt1756418,tt0285335\n' +
      'nm3860360\tFatima Driver\t\\N\t\\N\tmiscellaneous\ttt1129445\n' +
      'nm3860361\tNorbert Tissier\t\\N\t\\N\tvisual_effects\ttt1951265,tt1255953,tt6924650,tt0938330\n' +
      'nm3860362\tKeith Brion\t\\N\t\\N\tmusic_department,soundtrack\ttt1129445,tt0094731,tt0144550,tt14086890\n' +
      'nm3860363\tSamantha Laidlaw\t\\N\t\\N\tmiscellaneous,producer,assistant_director\ttt2126355,tt4052882,tt1809398,tt2058107\n' +
      'nm3860364\tGünter Häusler\t\\N\t\\N\tactor\ttt1629255\n' +
      'nm3860365\tMegan Mandy\t\\N\t\\N\tactress\ttt1627913\n' +
      'nm3860366\tMatt Creed\t\\N\t\\N\tactor,director,writer\ttt2707804,tt29808270,tt4941064,tt1484521\n' +
      'nm3860367\tYael Majors\t\\N\t\\N\tvisual_effects\ttt7713068,tt0107290,tt1392190,tt0831387\n' +
      'nm3860368\tMartin Marotta\t\\N\t\\N\tvisual_effects\ttt0420548\n' +
      'nm3860369\tDaniel Sky\t\\N\t\\N\tcomposer,soun'... 3540559 more characters,
    primaryProfession: '\\N'
  }
  {
    primaryName: 'Victor Coppola',
    primaryProfession: 'camera_department,miscellaneous'
  }
IMDB>
```

Sprawdziłem również działanie funkcji dla osoby Fonda i wtedy zwraca prawidłowo wyglądający wynik.

```
def zad5():
    # nameCollection.drop_index('primaryName_text')
    nameCollection.create_index([('primaryName', 'text')])
    zad5Query = {'$text': {'$search': 'Fonda', '$caseSensitive': True}}
    zad5aResult = nameCollection.count_documents(zad5Query)
    ic(zad5aResult)
    zad5bResult = nameCollection.find(zad5Query, {'_id': 0, 'primaryName': 1, 'primaryProfession': 1}).limit(5)
    printDocList(zad5bResult)
```

```
{'primaryName': 'Asia Fonda', 'primaryProfession': '\\N'}
{'primaryName': 'Gary Fonda', 'primaryProfession': 'actor'}
{'primaryName': 'Fonda Dsouza', 'primaryProfession': 'assistant_director,production_manager'}
{'primaryName': 'William Fonda', 'primaryProfession': 'cinematographer,camera_department,editor'}
{'primaryName': 'Fonda Berosini', 'primaryProfession': 'executive,producer'}
ic| zad5aResult: 86
```

Przy wywołaniu funkcji dla samej osoby Coppola również zwraca dziwny wynik.

Wniosek: Nieprawidłowo wyglądający wynik z wieloma tysiącami linii danych wynika zatem najprawdopodobniej z błędnej struktury danych dla Coppola np. brakującej danej w danym polu lub błędnych znakach uniemożliwiających prawidłowe przetworzenie danych dla MongoDB.

Zadanie 6

Zadanie 6: Utwórz indeks z porządkiem malejącym dla pola birthYear (kolekcja Name). Następnie wyświetl listę indeksów w kolekcji Name. Ile indeksów posiada ta kolekcja?

```
def zad6():
    nameCollection.create_index([('birthYear', -1)])
    indexes = list(nameCollection.list_indexes())
    for index in indexes:
        ic(index)
    indexCount = len(indexes)
    ic(indexCount)
```

```
ic| index: SON([('v', 2), ('key', SON([('_id', 1)])), ('name', '_id_')])
ic| index: SON([('v', 2), ('key', SON([('_fts', 'text'), ('_ftsx', 1)])), ('name', 'primaryName_text')
ic| index: SON([('v', 2), ('key', SON([('birthYear', -1)])), ('name', 'birthYear_-1')])
ic| indexCount: 3
```

```
('weights', SON([('primaryName', 1)])), ('default_language', 'english'), ('language_override', 'language'), ('textIndexVersion', 3)]
```

Ponieważ obrazek z wynikiem nie mieścił się w sprawozdaniu, to został podzielony na 2 części, lewą i prawą. 2 obrazek pokazuje dalszą część 2 linijki.
Wynik tekstowy wygląda tak:
ic| index: SON([('v', 2), ('key', SON([('_id', 1)])), ('name', '_id_')])
ic| index: SON([('v', 2), ('key', SON([('_fts', 'text'), ('_ftsx', 1)])), ('name', 'primaryName_text'), ('weights', SON([('primaryName', 1)])), ('default_language', 'english'), ('language_override', 'language'), ('textIndexVersion', 3)])
ic| index: SON([('v', 2), ('key', SON([('birthYear', -1)])), ('name', 'birthYear_-1')])
ic| indexCount: 3

Zadanie 7

Zadanie 7: Dla każdego filmu (kolekcja Title), który ma najwyższą średnią ocenę (10.0), dodaj pole max z wartością równą 1. W poleceniu skorzystaj z kolekcji Rating, która zawiera informacje o średniej ocenie filmu.

```python
def zad7():
    updatedMoviesIndexes = []
    highestRatedMovies = ratingCollection.find({'averageRating': 10.0}, {'_id': 0, 'tconst': 1})
    for movie in highestRatedMovies:
        result = titleCollection.update_one({'tconst': movie['tconst']}, {'$set': {'max': 1}})
        if result.modified_count > 0:
            updatedMoviesIndexes.append(movie['tconst'])

    ic(len(updatedMoviesIndexes))
    updatedMovies = titleCollection.find({'tconst': {'$in': updatedMoviesIndexes[:10]}})
    printDocList(updatedMovies)
```

```
ic| len(updatedMoviesIndexes): 5019
{'_id': ObjectId('665a5b8749b96058128f6f69'), 'tconst': 'tt0127236', 'titleType': 'video', 'primaryTitle': 'Renegades 2', 'originalTitle': 'Renegades 2',
{'_id': ObjectId('665a5b8749b96058128fad55'), 'tconst': 'tt0143979', 'titleType': 'short', 'primaryTitle': 'A View of Bosnia', 'originalTitle': 'A View of
{'_id': ObjectId('665a5b8749b96058128fea08'), 'tconst': 'tt0160316', 'titleType': 'movie', 'primaryTitle': 'Girls Loving Girls', 'originalTitle': 'Girls L
{'_id': ObjectId('665a5b8849b9605812903dda'), 'tconst': 'tt0183268', 'titleType': 'video', 'primaryTitle': 'Innocent Bi-Standers', 'originalTitle': 'Innoc
{'_id': ObjectId('665a5b8849b96058129043e9'), 'tconst': 'tt0184926', 'titleType': 'video', 'primaryTitle': 'Super Video Vixens 7: Christy Canyon', 'origin
{'_id': ObjectId('665a5b8849b96058129061aa'), 'tconst': 'tt0193147', 'titleType': 'video', 'primaryTitle': 'Execu-Comp #169 - Classic Super Sex', 'origina
{'_id': ObjectId('665a5b8949b9605812911f56'), 'tconst': 'tt0244729', 'titleType': 'tvMovie', 'primaryTitle': 'Prince of Wales: Kings in Waiting', 'origina
{'_id': ObjectId('665a5b8a49b9605812917f9d'), 'tconst': 'tt0281732', 'titleType': 'short', 'primaryTitle': 'Closed for Business', 'originalTitle': 'Closed
{'_id': ObjectId('665a5b8a49b9605812918c67'), 'tconst': 'tt0285180', 'titleType': 'short', 'primaryTitle': 'Heroes', 'originalTitle': 'Heroes', 'isAdult':
{'_id': ObjectId('665a5b8a49b960581291c52c'), 'tconst': 'tt0300386', 'titleType': 'video', 'primaryTitle': 'Las putas de New York', 'originalTitle': 'Las
```

```
', 'isAdult': 1, 'startYear': 1995, 'endYear': '\\N', 'runtimeMinutes': 91, 'genres': 'Adult', 'max': 1}
 of Bosnia', 'isAdult': 0, 'startYear': 1993, 'endYear': '\\N', 'runtimeMinutes': 16, 'genres': 'Documentary,Short', 'max': 1}
s Loving Girls', 'isAdult': 1, 'startYear': 1996, 'endYear': '\\N', 'runtimeMinutes': 60, 'genres': 'Adult', 'max': 1}
nocent Bi-Standers', 'isAdult': 1, 'startYear': 1989, 'endYear': '\\N', 'runtimeMinutes': '\\N', 'genres': 'Adult', 'max': 1}
ginalTitle': 'Super Video Vixens 7: Christy Canyon', 'isAdult': 1, 'startYear': 1993, 'endYear': '\\N', 'runtimeMinutes': '\\N', 'genres': 'Adult', 'max':
inalTitle': 'Execu-Comp #169 - Classic Super Sex', 'isAdult': 1, 'startYear': 1991, 'endYear': '\\N', 'runtimeMinutes': '\\N', 'genres': 'Adult', 'max': 1}
inalTitle': 'Prince of Wales: Kings in Waiting', 'isAdult': 0, 'startYear': 2000, 'endYear': '\\N', 'runtimeMinutes': '\\N', 'genres': 'Documentary', 'max'
sed for Business', 'isAdult': 0, 'startYear': 1997, 'endYear': '\\N', 'runtimeMinutes': 4, 'genres': 'Short', 'max': 1}
t': 0, 'startYear': 2000, 'endYear': '\\N', 'runtimeMinutes': '\\N', 'genres': 'Short', 'max': 1}
as putas de New York', 'isAdult': 1, 'startYear': 1999, 'endYear': '\\N', 'runtimeMinutes': 65, 'genres': 'Adult', 'max': 1}
```

Screen wynikowy został podzielony na 2 części, lewą i prawą dla przejrzystości.

Zadanie 8

Zadanie 8: W oparciu o kolekcje Title oraz Rating sprawdź średnią ocenę dowolnego filmu o określonym tytule oraz roku produkcji. Zapytanie powinno zwrócić nazwę filmu, rok produkcji oraz średnią ocenę.

```
def zad8(title, year):
    movie = titleCollection.find_one({'primaryTitle': title, 'startYear': year})
    if movie:
        rating = ratingCollection.find_one({'tconst': movie['tconst']})
        if rating:
            averageRating = rating['averageRating']
            zad8Result = {'primaryTitle': title, 'startYear': year, 'averageRating': averageRating}
            ic(zad8Result)
```

```
ic| zad8Result: {'averageRating': 8.1, 'primaryTitle': 'Blade Runner', 'startYear': 1982}
```

Wynik funkcji wywołanej z parametrami title = ‘Blade Runner’ i year = 1982.

Zadanie 9

Zadanie 9: Do filmu Blade Runner z roku 1982 dodaj pole rating, które będzie tablicą dokumentów z polami: averageRating oraz numVotes. Wartości pól w zagnieżdżonym dokumencie powinny posiadać odpowiednie wartości pobrane z kolekcji Rating.

```python
def zad9():
    movie = titleCollection.find_one({'primaryTitle': 'Blade Runner', 'startYear': 1982})
    if movie:
        rating = ratingCollection.find_one({'tconst': movie['tconst']})
        if rating:
            average_rating = rating['averageRating']
            num_votes = rating['numVotes']
            titleCollection.update_one({'_id': movie['_id']}, {'$set': {'rating': [{'averageRating': average_rating, 'numVotes': num_votes}]}})
            print("Pole rating zostało pomyślnie dodane do dokumentu.")
            movieAfterUpdate = titleCollection.find_one({'primaryTitle': 'Blade Runner', 'startYear': 1982})
            ic(movieAfterUpdate)
        else:
            print("Ocena dla filmu 'Blade Runner' z roku 1982 nie została znaleziona.")
    else:
        print("Film 'Blade Runner' z roku 1982 nie został znaleziony.")
```

```
Pole rating zostało pomyślnie dodane do dokumentu.
ic| movieAfterUpdate: {'_id': ObjectId('665a5b8549b96058128ecb7b'),
                       'endYear': '\\N',
                       'genres': 'Action,Drama,Sci-Fi',
                       'isAdult': 0,
                       'originalTitle': 'Blade Runner',
                       'primaryTitle': 'Blade Runner',
                       'rating': [{'averageRating': 8.1, 'numVotes': 825268}],
                       'runtimeMinutes': 117,
                       'startYear': 1982,
                       'tconst': 'tt0083658',
                       'titleType': 'movie'}
```

Zadanie 10

Zadanie 10: Zmodyfikuj pole rating w dokumencie z Zadania 9, dodając jeszcze jeden dokument z polami averageRating oraz numVotes oraz z wartościami: 10 oraz 55555.

```python
def zad10():
    movie = titleCollection.find_one({'primaryTitle': 'Blade Runner', 'startYear': 1982})
    if movie:
        titleCollection.update_one({'_id': movie['_id']}, {'$push': {'rating': {'averageRating': 10, 'numVotes': 55555}}})
        print("Nowy dokument został pomyślnie dodany do pola rating.")
        movieAfterUpdate = titleCollection.find_one({'primaryTitle': 'Blade Runner', 'startYear': 1982})
        ic(movieAfterUpdate)
    else:
        ic("Film 'Blade Runner' z roku 1982 nie został znaleziony.")
```

```
Nowy dokument został pomyślnie dodany do pola rating.
ic| movieAfterUpdate: {'_id': ObjectId('665a5b8549b96058128ecb7b'),
                       'endYear': '\\N',
                       'genres': 'Action,Drama,Sci-Fi',
                       'isAdult': 0,
                       'originalTitle': 'Blade Runner',
                       'primaryTitle': 'Blade Runner',
                       'rating': [{'averageRating': 8.1, 'numVotes': 825268},
                                  {'averageRating': 10, 'numVotes': 55555}],
                       'runtimeMinutes': 117,
                       'startYear': 1982,
                       'tconst': 'tt0083658',
                       'titleType': 'movie'}
```

Zadanie 11

Zadanie 11: Usuń pole rating dodane do filmu Blade Runner w Zadaniu 10.

```python
def zad11():
    movie = titleCollection.find_one({'primaryTitle': 'Blade Runner', 'startYear': 1982})
    if movie:
        titleCollection.update_one({'_id': movie['_id']}, {'$unset': {'rating': ''}})
        print("Pole rating zostało pomyślnie usunięte z dokumentu.")
        movieAfterUpdate = titleCollection.find_one({'primaryTitle': 'Blade Runner', 'startYear': 1982})
        ic(movieAfterUpdate)
    else:
        print("Film 'Blade Runner' z roku 1982 nie został znaleziony.")
```

```
Pole rating zostało pomyślnie usunięte z dokumentu.
ic| movieAfterUpdate: {'_id': ObjectId('665a5b8549b96058128ecb7b'),
                       'endYear': '\\N',
                       'genres': 'Action,Drama,Sci-Fi',
                       'isAdult': 0,
                       'originalTitle': 'Blade Runner',
                       'primaryTitle': 'Blade Runner',
                       'runtimeMinutes': 117,
                       'startYear': 1982,
                       'tconst': 'tt0083658',
                       'titleType': 'movie'}
```

Zadanie 12

Zadanie 12: Do filmu Pan Taduesz z 1999 roku dodaj pole avgRating z wartością 9.1. Jeśli nie ma takiego filmu, polecenie powinno zadziałać jak upsert.

```python
def zad12():
    # titleCollection.delete_one({'primaryTitle': 'Pan Tadeusz', 'startYear': 1999})
    movie = titleCollection.find_one({'primaryTitle': 'Pan Tadeusz', 'startYear': 1999})
    if not movie:
        titleCollection.update_one({'primaryTitle': 'Pan Tadeusz', 'startYear': 1999}, {'$set': {'avgRating': 9.1}}, upsert=True)
        print("Film 'Pan Tadeusz' z 1999 roku został pomyślnie dodany z polem avgRating.")
        movieAfterUpdate = titleCollection.find_one({'primaryTitle': 'Pan Tadeusz', 'startYear': 1999})
        ic(movieAfterUpdate)
    else:
        titleCollection.update_one({'_id': movie['_id']}, {'$set': {'avgRating': 9.1}})
        print("Pole avgRating zostało dodane/zaaktualizowane do filmu 'Pan Tadeusz' z 1999 roku.")
```

```
Film 'Pan Tadeusz' z 1999 roku został pomyślnie dodany z polem avgRating.
ic| movieAfterUpdate: {'_id': ObjectId('665f81f7722c3ed835093bd1'),
                       'avgRating': 9.1,
                       'primaryTitle': 'Pan Tadeusz',
                       'startYear': 1999}
```

Wywołanie funkcji jeśli nie ma rekordu z tytułem ‘Pan Tadeusz’ w roku 1999.

```
Pole avgRating zostało dodane/zaaktualizowane do filmu 'Pan Tadeusz' z 1999 roku.
```

Wywołanie funkcji jeśli istnieje rekord z tytułem ‘Pan Tadeusz’ w roku 1999.

Zadanie 13

Zadanie 13: Z kolekcji Title usuń dokumenty, które reprezentują filmy wyprodukowane przed 1989 rokiem. Ile takich dokumentów zostało usuniętych?

```
def zad13():
    zad13Result = titleCollection.delete_many({'startYear': {'$lt': 1989}})
    ic(zad13Result.deleted_count)
```

```
ic| zad13Result.deleted_count: 1144203
```

Wywołanie funkcji z zadań:

```
zad1()
zad2()
zad3()
zad4a()
zad4b()  # DZIALA, ALE WYKONUJE SIĘ PONAD 3H
zad5()  # Wynik wyjątkowo dziwny, ale taki sam przy wykonaniu w MongoDB Shell.
        # Fonda ma normalne wyniki, ale Coppola ma coś popsute dane
zad6()
zad7()  # DZIALA, ALE WYKONUJE SIĘ PONAD 3H
zad8("Blade Runner", 1982)
zad9()
zad10()
zad11()
zad12()
zad13()
```