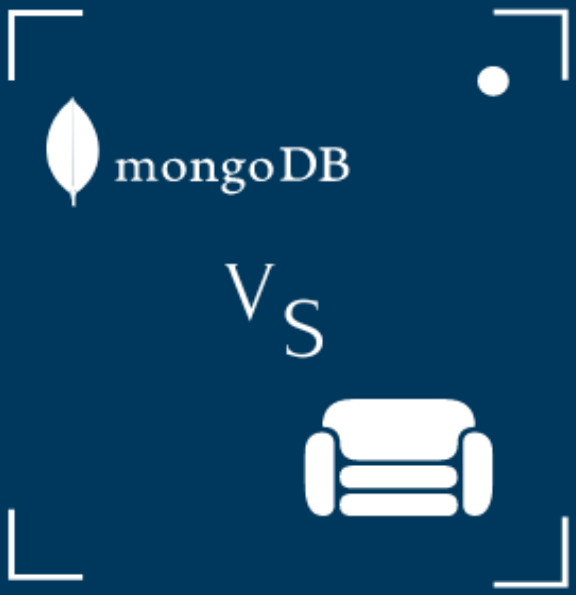
Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Споредба помеѓу претставници на Document бази на податоци



Предмет: Тимски проект

Професор: Слободан Калајџиски

Студенти: Сандра Стојановска, 183045
Елена Ридова, 183034
Филип Ставров, 183054
Едвин Лековиќ, 183131

Датум: 18.03.2022

Содржина

Вовед

Како голем тренд во денешната индустрија за менаџмент со податоци, се сретнуваат неструктурираните бази на податоци. Ова се должи на брзорастечката популарност на Big Data т.е. големите податоци, коишто водат до замена на традиционалните системи за менаџирање со бази на податоци, каде што имаме структурирани податоци со предефинирана шема. Оттука, се појавува прашањето како да се справиме со полуструктурирани или неструктурирани податоци со кои секојдневно се сретнуваме на Вебот, а решението се наоѓа токму во неструктурираните бази на податоци.

Негативната страна на релационите бази на податоци е што го следат т.н. ACID концепт, којшто носи свои ограничувања од страна на брзина и скалирање кога станува збор за работа со големи количества на податоци. Подемот на неструктурираните бази на податоци беше стимул за истражување на еден од нивните типови, односно document-based базите. Овој тип на бази е фокусиран на методите за складирање и пристап кои се оптимизирани за работа со документи наспроти редици или записи како што сме навикнати во системите за менаџирање со релациони бази на податоци.

Во фокусот на нашето истражување се двете најпопуларни бази од овој тип, MongoDB и CouchDB. Целта беше да направиме компаративна анализа на перформансите на овие две бази преку тестирање со извршување на исти прашалници врз исто податочно множество.

Методологија

И двете бази, MongoDB и CouchDB, овозможуваат креирање и надградување на програми без потреба да следат некоја главна шема. Менаџирање со содржини и справување со податоци во мобилни апликации се две од полињата каде што употреба на ваков тип на бази е соодветна.

Една од причините за моменталната популарност на MongoDB е нејзината флексибилна шема што ѝ дозволува лесно да еволуира, да се скалира и да ги поддржува сите функциии на современите бази на податоци, на начин којшто е едноставен за работа на програмерите.

И CouchDB не е многу поразлична, но е создадена со поголема слика во позадина. Нејзините компоненти може да се искористат како градбени единки кои решаваат проблеми на малку поразличен начин, за поголеми и покомплексни системи.

Пред да навлеземе во техничките детали на овие две нерелациони бази на податоци, прво ќе го разгледаме податочното множество коешто го обработувавме. Ова податочно множество е составено од JSON објекти за филмови издадени заклучно до 2017 година. Множеството има големина од 45000 записи, кои имаат 20 атрибути (колони), при што за истите извршивме неколку претпроцесирачки методи со цел подобро да го разбереме множеството.

Како што спомнавме и погоре, множеството со кое се одлучивме да работиме е множеството за филмови. Ваквиот избор го направивме поради “богатата” содржина на истото, односно големиот број на карактеристики (колони), кој ќе овозможи креирање на поинтересни и покомплексни прашалници.

Пред процесот на пишување на прашалниците, клучно е да се добие добра претстава за множеството, односно да се разберат неговите особености, како и врските кои постојат помеѓу различните карактеристики. Поради таа причина, ние се одлучивме да го користиме Python и големиот број на библиотеки кои истиот ги нуди за анализа и обработка на

податоците, со цел да направиме детален преглед на множеството, како и негово претпроцесирање. Самиот процес на претпроцесирање ни овозможи да ги увидиме карактеристиките кои содржат голем број на невалидни вредности, кои не се од полза за пишувањето на прашалниците. Исто така, успеавме да ги видиме и релациите и степените на поврзаност помеѓу различните карактеристики, како и да навлеземе подетално во секој од типовите на карактеристиките, со цел да направиме конверзија од еден во друг тип, доколку истата е потребна.

Вчитување на множеството и разбирање на податочните типови на секоја од неговите карактеристики

Прв чекор од ваквата анализа е вчитување на множеството, користејќи го pandas модулот во Python. Ова ни овозможува да направиме увид во структурата на множеството и да добиеме претстава за карактеристиките со кои работиме. Дел од множеството е прикажан на сликата број 1, која следува во продолжение.

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	popularity
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	tt0114709	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	21.9469
1	False	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8844	tt0113497	en	Jumanji	When siblings Judy and Peter discover an encha...	17.0155
2	False	{'id': 119050, 'name': 'Grumpy Old Men Collect...	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...	NaN	15602	tt0113228	en	Grumpier Old Men	A family wedding reignites the ancient feud be...	11.7129

Слика 1. “Приказ на дел од содржината на податочното множество за филмови”

Следно, одлучивме да добиеме информации за податочниот тип на секоја од карактеристиките. Ова е од големо значење, бидејќи дел од карактеристиките се претставени како текстуални полиња (string-ови), а всушност претставуваат комплексни објекти, или нумерички вредности. Со овој чекор успеавме да увидиме кои од дадените карактеристики (колони) ќе имаат потреба од конверзија на податочниот тип, со цел истите

да можат да бидат пристапувани при пишувањето на прашалниците. Приказ на податочните типови е даден на сликата број 2.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 24 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   adult                  45466 non-null  object
 1   belongs_to_collection  4494 non-null   object
 2   budget                 45466 non-null  object
 3   genres                 45466 non-null  object
 4   homepage               7782 non-null   object
 5   id                     45466 non-null  object
 6   imdb_id                45449 non-null  object
 7   original_language      45455 non-null  object
 8   original_title         45466 non-null  object
 9   overview               44512 non-null  object
 10  popularity             45461 non-null  object
 11  poster_path            45080 non-null  object
 12  production_companies   45463 non-null  object
 13  production_countries   45463 non-null  object
 14  release_date           45379 non-null  object
 15  revenue                45460 non-null  float64
 16  runtime                45203 non-null  float64
 17  spoken_languages       45460 non-null  object
 18  status                 45379 non-null  object
 19  tagline                20412 non-null  object
 20  title                  45460 non-null  object
 21  video                  45460 non-null  object
 22  vote_average           45460 non-null  float64
 23  vote_count             45460 non-null  float64
dtypes: float64(4), object(20)
memory usage: 8.3+ MB
```

Слика 2. “Податочен тип на секоја од колоните”

Она што можеме да го забележиме на сликата број 2 е дека колоните popularity и budget, иако содржат нумерички вредности, како податочен тип имаат објект. Користејќи дел од алатките кои ни ги нуди pandas модулот, направивме конверзија на податочните типови на овие две колони во нумерички вредности, попрецизно float вредности.

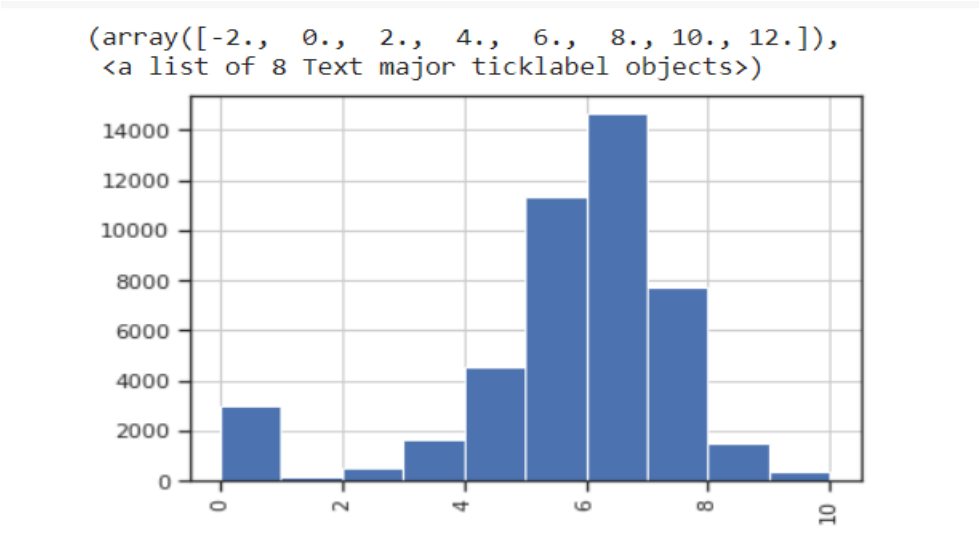
Иако дел од колоните како податочен тип имаат објект, сепак во нивната внатрешна структура тие содржат листи, или други објекти, кои се прикажани како текстуални полиња, а не како листи/објекти. Поради ова, направиме конверзија на секој внатрешен елемент од целокупниот објект, со цел истиот да може да биде пристапен и манипулиран, што не би било возможно доколку тој е стринг. Дел од колоните кај кои ова е случај се genres,

production_companies, production_countries и spoken_languages. Конкретен пример од ваквата конверзија е прикажан на слика број 3, каде можеме да видиме дека првично добиваме текстуално поле (string), а потоа листа.

```
[ ]  data['production_countries'][0]

     '[{'iso_3166_1': 'US', 'name': 'United States of America'}]'

[ ]  data['production_companies'][0]

     list
```

Слика 3. “Конверзија на податочен тип”

Следно, креиравме неколку хистограми, кои ни овозможија да ги видиме опсезите на дадени карактеристики кои се нумерички, како и да добиеме претстава за нивната распределба.



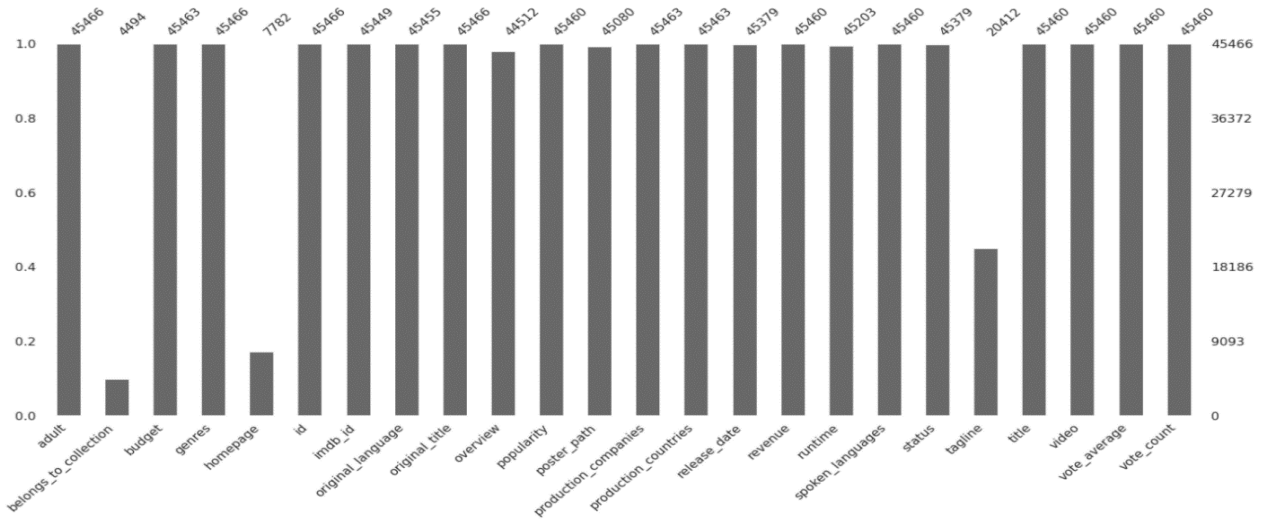
Хистограм за колоната **vote-average**.

Слика 4. “Хистограм за колоната vote-average”

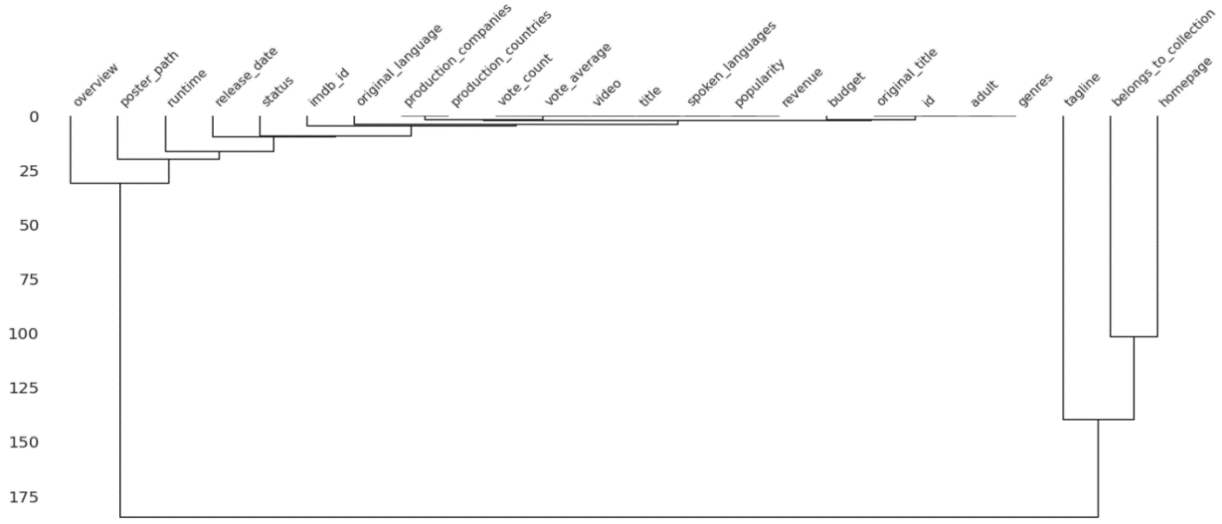
На сликата број 4, можеме да видиме дека колоната vote-average има вредности во опсегот од 0, до 10. Ова ни кажува дека најголем дел од филмовите имаат просечен број на гласови со вредност 7, додека пак, многу малку филмови добиле просечен број на гласови 1 или 10.

Анализа на вредностите кои недостасуваат во дадените колони на множеството

Чест случај при обработка и работа со вакви големи множества е појавата на невалидни вредности. Од таа причина, следен чекор во процесот на претпроцесирање на податоците е да ги пронајдеме ваквите невалидни вредности. Направивме детален приказ на невалидните вредности, како и на вредностите кои недостасуваат по колони/карактеристики. Ова ни овозможи да увидиме кои од колоните имаат голем број на вредности кои недостасуваат, односно се неупотребливи и не можат да се искористат во процесот на пишување на прашалници. Тоа доведе до одлука да ги отстраниме ваквите колони од множеството. Во продолжение следуваат неколку визуелизации, креирани со помош на различни методи за откривање на невалидни и вредности кои недостасуваат.



Слика 5. “Дијаграм за приказ на вредности кои недостасуваат”



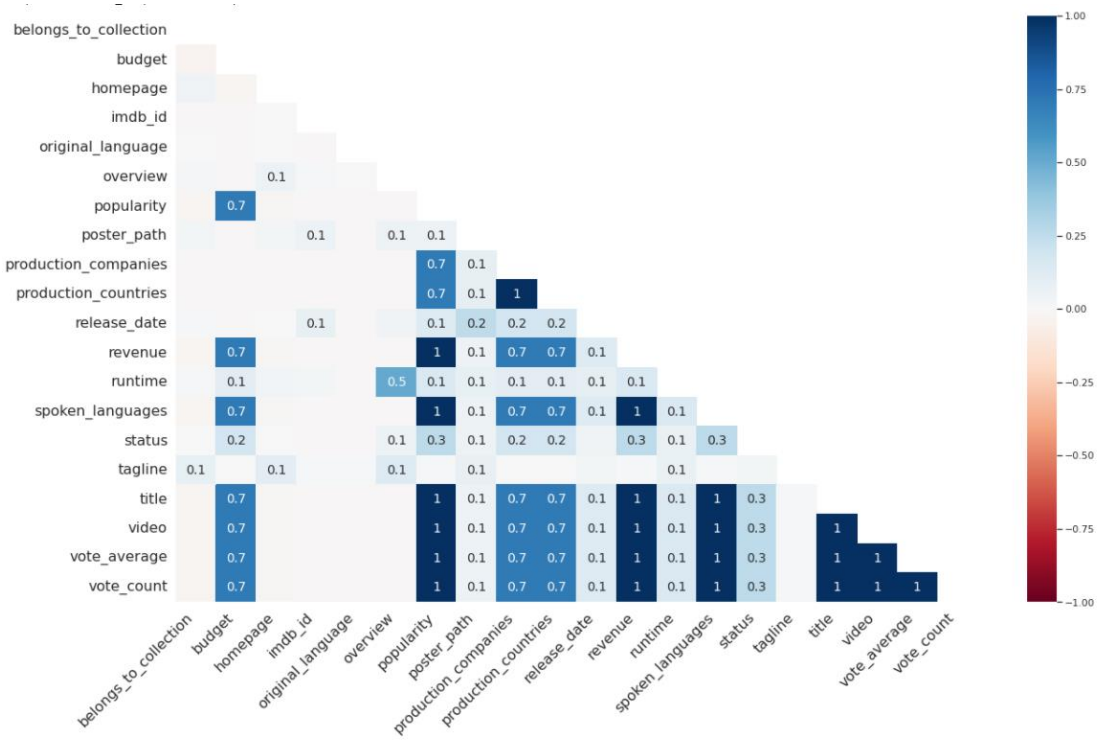
Слика 6. “Дендограм за приказ на вредности кои недостасуваат”

	Br. vrednosti sto nedostasuvaat	Vo procenti
adult	0	0.000000
belongs_to_collection	40972	90.115691
budget	3	0.006598
genres	0	0.000000
homepage	37684	82.883913
id	0	0.000000
imdb_id	17	0.037391
original_language	11	0.024194
original_title	0	0.000000
overview	954	2.098271
popularity	6	0.013197
poster_path	386	0.848986
production_companies	3	0.006598
production_countries	3	0.006598
release_date	87	0.191352
revenue	6	0.013197
runtime	263	0.578454
spoken_languages	6	0.013197
status	87	0.191352
tagline	25054	55.104914
title	6	0.013197
video	6	0.013197
vote_average	6	0.013197
vote_count	6	0.013197

Слика 7. “Процентуален приказ на вредности кои недостасуваат”

Од сликите 5, 6 и 7 можеме да забележиме дека дел од колоните имаат голем процент на вредности кои недостасуваат. Помеѓу нив се издвојуваат колоните “belongs-to-collection”, “homepage” и “tagline”. Пред да направиме отстранување на овие колони од множеството, решивме да направиме анализа на нивната поврзаност со останатите колони, т.е. да

увидиме колкаво е влијанието на присуството/отсуството на овие колони врз останатите колони од множеството. Детален приказ на оваа визуелизација е даден на слика 8.



Слика 8. “Дијаграм за корелација помеѓу присуство и отсуство на вредностите на колоните”

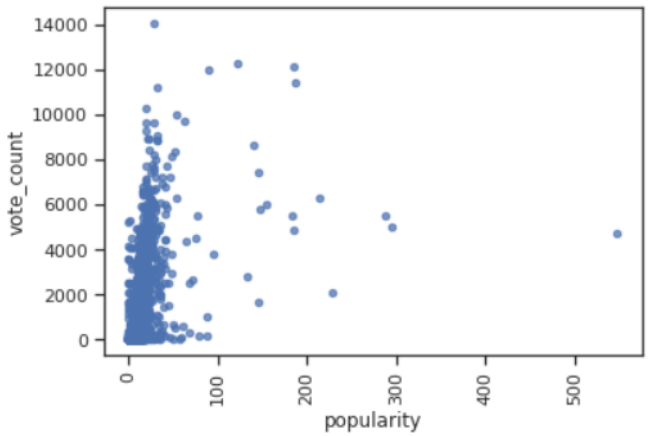
Од сликата број 8 можеме да заклучиме дека вредностите на колоните “belongs-to-collection”, “homepage” и “tagline” речиси и немаат никакво влијание врз вредностите на останатите колони од множеството, односно нивното ниво на корелација е многу мало. Поради тоа, одлучивме да ги отстраниме овие три колони од множеството, бидејќи како што споменавме и претходно, истите не можат да се искористат при пишувањето на прашалници. Во продолжение следува приказ на множеството по отстранувањето на овие три колони.

	adult	budget	genres	id	imdb_id	original_language	original_title	overview	popularity	poster_path	production_companies	production_countries	release_date	revenue	runtime	spoken_languages
0	False	30000000.0	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	862	tt0114709	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	21.946943	/rhlRbceoE9lR4veEXuwCC2wARtG.jpg	[{'name': 'Pixar Animation Studios', 'id': 3}]	[{'iso_3166_1': 'US', 'name': 'United States o...	1995-10-30	373554033.0	81.0	[{'iso_639_1': 'en', 'name': 'English'}]
1	False	65000000.0	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	8844	tt0113497	en	Jumanji	When siblings Judy and Peter discover an encha...	17.015539	/vzmL6fP7aPKNKPRTFnZmiUfciyV.jpg	[{'name': 'TriStar Pictures', 'id': 559}, {'na...	[{'iso_3166_1': 'US', 'name': 'United States o...	1995-12-15	262797249.0	104.0	[{'iso_639_1': 'en', 'name': 'English'}, {'iso...
2	False	0.0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...	15602	tt0113228	en	Grumpier Old Men	A family wedding reignites the ancient feud be...	11.712900	/6ksm1sjKMFLbO7UY2i6G1ju9SML.jpg	[{'name': 'Warner Bros.', 'id': 6194}, {'name'...	[{'iso_3166_1': 'US', 'name': 'United States o...	1995-12-22	0.0	101.0	[{'iso_639_1': 'en', 'name': 'English'}]
3	False	16000000.0	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	31357	tt0114885	en	Waiting to Exhale	Cheated on, mistreated and stepped on, the wom...	3.859495	/16XOMpEaLWkrcPqSQqhTmeJuqQl.jpg	[{'name': 'Twentieth Century Fox Film Corporat...	[{'iso_3166_1': 'US', 'name': 'United States o...	1995-12-22	81452156.0	127.0	[{'iso_639_1': 'en', 'name': 'English'}]
4	False	0.0	[{'id': 35, 'name': 'Comedy'}]	11862	tt0113041	en	Father of the Bride Part II	Just when George Banks has recovered from his ...	8.387519	/e64sOI48hQXyru7naBFyssKFxVd.jpg	[{'name': 'Sandollar Productions', 'id': 5842}...	[{'iso_3166_1': 'US', 'name': 'United States o...	1995-02-10	76578911.0	106.0	[{'iso_639_1': 'en', 'name': 'English'}]
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
45461	False	0.0	[{'id': 18, 'name': 'Drama'}, {'id': 10751, 'n...	439050	tt6209470	fa	رنگ خدا	Rising and falling between a man and woman.	0.072051	/jldsYfInld4tTWPx8es3uzsB1I8.jpg	[]	[{'iso_3166_1': 'IR', 'name': 'Iran'}]	NaN	0.0	90.0	[{'iso_639_1': 'fa', 'name': 'فارسی'}]

Слика 9. "Приказ на податочното множество по отстранување на гореспоменатите колони"

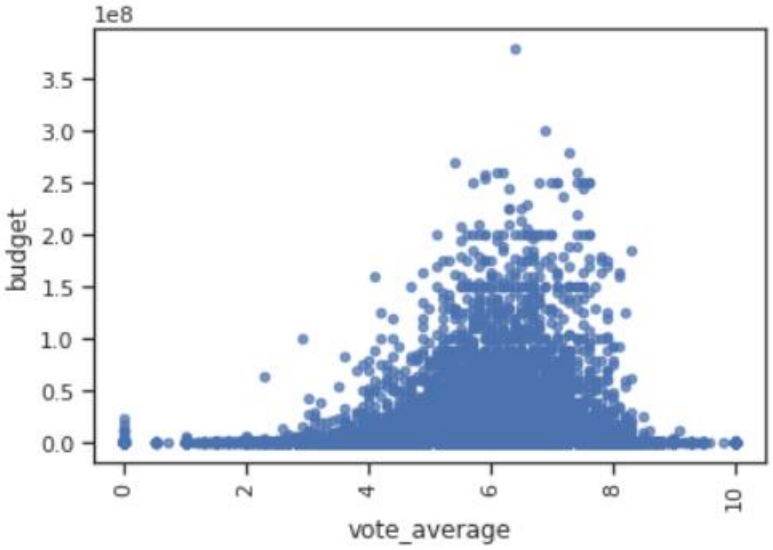
Анализа на врските и поврзаноста помеѓу секоја од колоните на податочното множество

Клучно за пишување на добри прашалници е добро познавање на карактеристиките и нивната меѓусебна поврзаност. Поради оваа причина креиравме неколку scatter-plots, кои ни овозможуваат да ја анализираме поврзаноста помеѓу две нумерички колони. Ова ни овозможува да напишеме посложени прашалници, чија основа ќе се темели на меѓусебните врски помеѓу различните карактеристики. Во продолжение следуваат неколку вакви дијаграми на поврзаност.



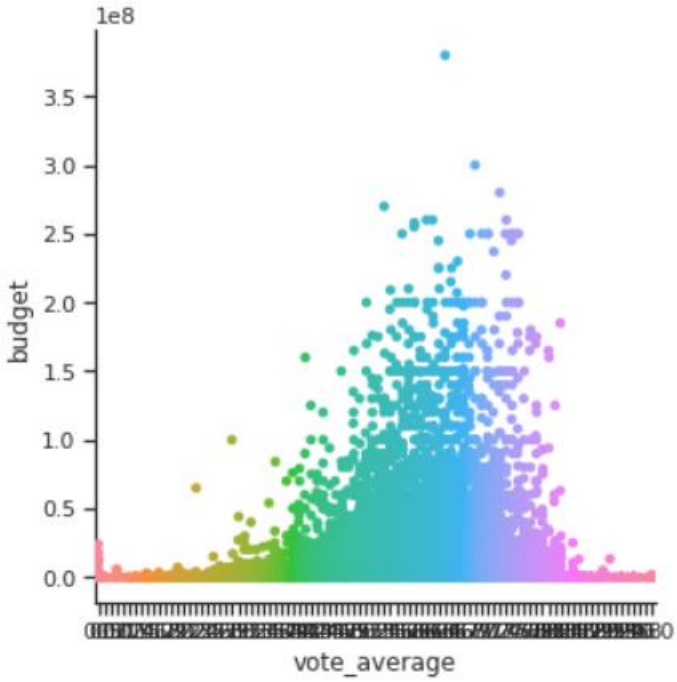
Слика 10. "Дијаграм на поврзаност за колоните popularity и vote-count"

На сликата број 10 ни е прикажан дијаграм на поврзаност за колоните popularity и vote-count. Од истиот можеме да забележиме дека помеѓу овие две карактеристики постои средна јачина на позитивна, линеарна поврзаност, односно, колку е поголема вредноста на vote-count за даден филм, толку ќе е поголема и вредноста за неговата популарност (иако, не за многу).



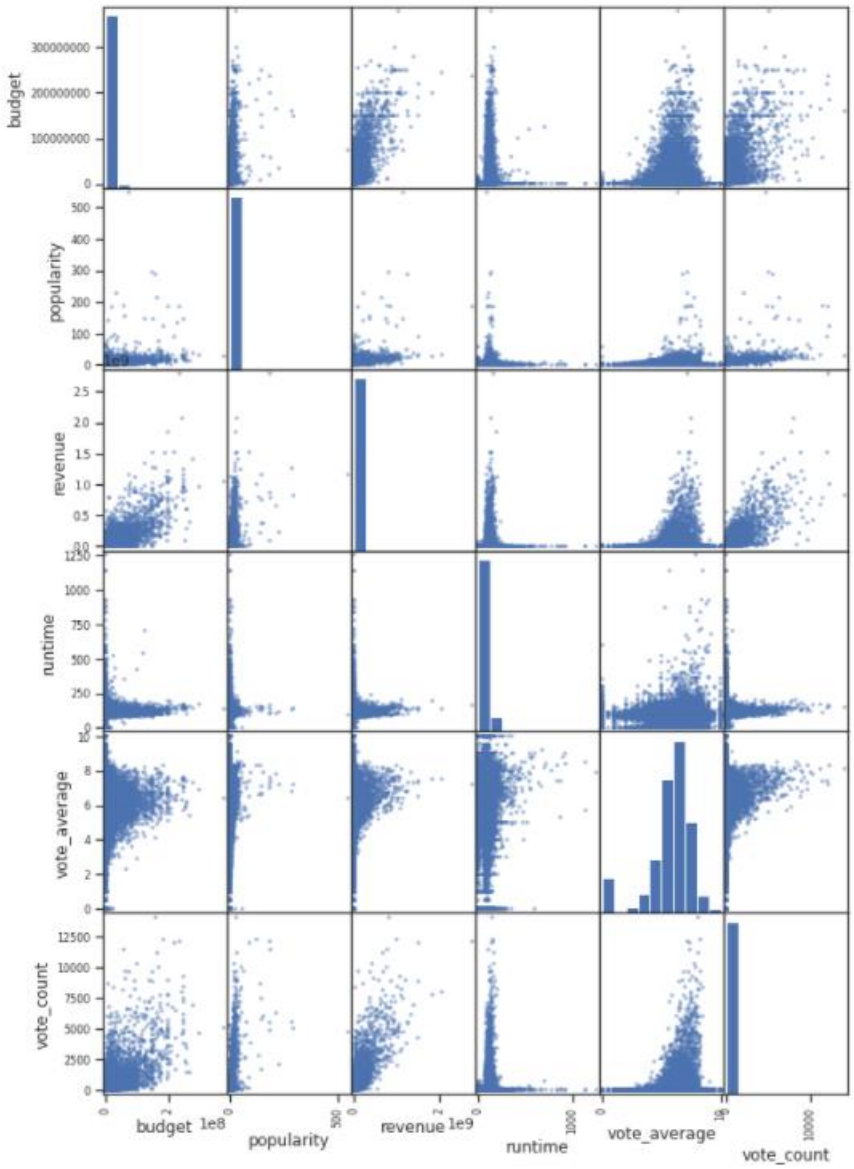
Слика 11. “Дијаграм на поврзаност за колоните vote-average и budget”

На сликата број 11 е прикажан дијаграм на поврзаност за колоните vote-average и budget, од кој можеме да забележиме дека овие две колони се меѓусебно поврзани. Нивната меѓусебна врска е силна, позитивна, но не е линеарна. Односно, овој дијаграм ни кажува дека колку повеќе се вложува во филмот, гледајќи од аспект на финансиски средства, поголема е веројатноста истиот да им се допадне на гледачите. Во продолжение следува уште еден дијаграм на поврзаност за овие две колони, кој ни дава дополнителни информации за различните кластери на податоци.



Слика 12. “Дијаграм на поврзаност за колоните vote-average и budget со кластери”

За детална анализа на врските кои постојат помеѓу секоја од нумеричките колони креиравме и scatter-matrix, т.е. матрица на корелација, која е прикажана на слика број 13. Оваа визуелизација ни помогна во процесот на пишување на корелирани прашања, односно прашања кои прават паралелно анализа на две или повеќе колони.



Слика 13. “Матрица на корелација помеѓу нумеричките карактеристики”

Од матрицата на корелација можеме да ги процениме врските кои постојат помеѓу секоја од нумеричките колони. Пример можеме да забележиме дека има тесна поврзаност помеѓу “revenue” и “budget”. Ова ни укажува на фактот дека филмовите за кои се издвојува поголем буџет, односно се вложува повеќе, резултираат со поголем профит. Ваквите врски помеѓу карактеристиките одлучивме да ги прикажеме и табеларно, односно да ја видиме во бројки поврзаноста помеѓу истите.

Correlation matrix

	budget	popularity	revenue	runtime	vote_average	vote_count
budget	1.00	0.45	0.77	0.13	0.07	0.68
popularity	0.45	1.00	0.51	0.13	0.15	0.56
revenue	0.77	0.51	1.00	0.10	0.08	0.81
runtime	0.13	0.13	0.10	1.00	0.16	0.11
vote_average	0.07	0.15	0.08	0.16	1.00	0.12
vote_count	0.68	0.56	0.81	0.11	0.12	1.00

Слика 14. “Табеларен приказ на поврзаноста помеѓу колоните”

Како што споменавме и погоре, колку е поголем буџетот кој ќе се вложи во процесот на креирање на еден филм, толку е поголем и профитот кој филмот ќе го донесе на продукцијата. Можеме да видиме дека поврзаноста помеѓу овие две карактеристики изнесува 0.77.

Исто така тесна поврзаност постои помеѓу “revenue” и “vote_count”, со дури 0.81. Ова ни кажува дека профитот на филмот е поголем, доколку истиот многу им се допаднал на гледачите, па тие гласале за него.

Исто така и буџетот има одредено ниво на поврзаност врз “vote_count”, кое изнесува 0.68. Според ова, можеме да претпоставиме дека филм во кој се вложени повеќе финансиски средства, е најверојатно филм кој има помодерни ефекти и сценографии, па веројатноста дека ќе им се допадне на гледачите е поголема.

Од друга страна пак, можеме да забележиме дека времетраењето на филмот не е пресудно за неговата популарност. Односно, даден филм може да биде и многу краток, но и многу долг, а сепак да постигне голема популарност и да ги освои симпатиите на гледачите.

По направената анализа на податоците, добивме добра претстава за генералната структура на множеството, како и идеи во врска со прашалниците кои можеме да ги пишуваме.

Следно, креиравме и подготвивме две посебни бази на податоци, користејќи ги претставниците на document-based базите на податоци MongoDB и CouchDB. Исто така, креиравме и два посебни модели со кои го направивме внесот на податоците во базата. Притоа, секој ред од погоре прикажаната табела беше внесен како посебен документ, користејќи го соодветниот модел за внес на податоците.

Импортирање и поврзување на двете бази соодветно, MongoDB и CouchDB

За креирање и поврзување на MongoDB користевме Python и MongoDB Atlas. Atlas е сервис базиран во облак за работа со бази на податоци, направен од истиот тим кој работел на MongoDB. Истиот го упростува менаџирањето со бази на податоци нудејќи ја потребната разновидност за градење на глобални апликации со добри перформанси базирани во облак. Од друга страна, речникот и листите како податочни типови кои ги нуди Python, го прават најдобар избор покрај JavaScript за манипулирање на JSON документи. Стандардниот MongoDB драјвер за Python, PyMongo е едноставен за користење и нуди интуитивно API за пристап до бази на податоци, колекции и документи.

Со користење на оваа комбинација, процесот на додавање на 45 000 записи во базата беше многу едноставен, при што дозволува и повеќе начини за пристап до истата. Затоа, по импортирањето, се поврзавме локално со базата и преминавме на извршување на претходно креираните прашалници во терминал.

За креирање и поврзување на CouchDB користевме Python и Project Fauxton. Project Fauxton претставува веб базиран интерфејс, кој го користевме за да ја дефинираме логиката на самите прашалници. Истиот е базиран на map-reduce концепт, кој нуди неколку можности за избор на reduce функции, како што се sum, count, stats, а покрај тоа нуди и можност за custom дефинирање на сопствен метод. Базиран е на JavaScript, кој е доста едноставен и нуди едноставна манипулација и “играње” со документите.

Процесот на импортирање на податоците беше овозможен од couchdb пакетот во Python, со чијашто помош на брз и едноставен начин успеавме да ги внесеме JSON документите во

базата. По импортирање на податоците во базата, со истата се поврзавме локално и започнавме со извршување на веќе креираните прашалници во терминал.

Модели на агрегација и имплементација

Со цел да искористиме различни модели за агрегација и да си поиграме со самата база, прашалниците ги направивме со растечка тежина, од полесни кон потешки. Притоа, во нив се сретнуваат следните агрегациски модели:

- Average (AVG): Просек на податочните вредности.
- Sum: Вкупна вредност од податочните вредности.
- Count: Број на записи.
- Maximum (Max): Максимална вредност на дадени податочни вредности.
- Minimum (Min): Минимална вредност на дадени податочни вредности.
- Group: Групирање на дадени податочни вредности според одреден критериум.
- Map-Reduce: За групирање на сите податоци базирани на клуч-вредност и reduce функција која се користи за изведување на операции на селектираните податоци.

Во продолжение може да се видат самите прашалници и нивната имплементација во двете бази.

Прашалник	MongoDB	CouchDB
Најди ја просечната популарност на сите филмови.	db.projectdata.aggregate([{ $group: { "_id" : null, "Average Popularity" : { $avg : "$popularity"}}}])	//map function (doc) { if (doc.id && doc.popularity) emit('average', doc.popularity); } //reduce function (keys, values, rereduce) { return sum(values)/values.length; }

Прикажи ги имињата на филмовите кои имаат поголема популарност од просечната и кои имаат буџет поголем од просечниот.	var average_pop = db.projectdata.aggregate([{ "$group": { "_id": "null", avg: { "$avg": "$popularity"} }}]).toArray()[0]["avg"]; var average_budget = db.projectdata.aggregate([{ "$group": { "_id": "null", avg: { "$avg": "$budget"} }}]).toArray()[0]["avg"]; db.projectdata.aggregate([{ $match : { 'popularity' : { $gt :average_pop}, 'budget' : { $gt: average_budget}}},{ $project :{ "_id" :0 , "original_title" :1 }}])	**//map** function (doc) { if(doc.original_title && doc.popularity && doc.budget == 0 || doc.budget){ emit("title", {"original_title":doc.original_title,"popularity":doc.popularity,"budget":doc.budget}); } } **//reduce** function (keys, values, rereduce) { if (rereduce) { let all = []; for(let val in values){ values[val].forEach(v => all.push(v)); } let popularityList = all.map(v => v.popularity); let budgetList = all.map(v => v.budget); const popularityAvg = popularityList.reduce((a,b) => a+b,0) / popularityList.length; const budgetAvg = budgetList.reduce((a,b) => a+b,0) / budgetList.length;

		return all.filter(v => v.popularity > popularityAvg && v.budget >= budgetAvg).map(v => v.original_title); } else { return values; } }
Провери дали филмот со најголем буџет е и филмот со најголема заработка.	var max_revenue = db.projectdata.aggregate([{ "$group": { "_id": "null", max: { "$max": "$revenue"} }}]).toArray()[0]["max"]; var max_budget = db.projectdata.aggregate([{ "$group": { "_id": "null", max: { "$max": "$budget"} }}]).toArray()[0]["max"]; db.projectdata.aggregate([{ $match: { $and: [{ revenue: "max_revenue" }, { views: "max_budget" }] } },]);	**//map** function (doc) { emit(doc.original_title, {"original_title":doc.original_title,"budget":doc.budget,"revenue":doc.revenue}); } **//reduce** function (keys, values, rereduce) { if (rereduce) { let all = []; for(let val in values){ values[val].forEach(v => all.push(v)); } let budgetList = all.map(v => v.budget); let revenueList = all.map(v => v.revenue); budgetList.sort(); revenueList.sort();

		const budgetMax = budgetList[budgetList.length-1]; const revenueMax = revenueList[revenueList.length-1]; all.filter(obj => obj.budget === budgetMax && obj.revenue === revenueMax); let objBudgetMax = all.filter(obj => obj.budget === budgetMax)[0]; let objRevenueMax = all.filter(obj => obj.revenue === revenueMax)[0]; return { "maxBudgetMovieTitle":objBudgetMax.original_title, "maxRevenueMovieTitle":objRevenueMax.original_title, "is it equal": objBudgetMax.original_title === objRevenueMax.original_title, }; } else { return values; } }
Најди го филмот со најголем број на гласови.	db.projectdata.find({ vote_count : { $ne:	function (doc) { emit(doc.vote_count, doc.original_title);

	""}}).sort({vote_count :-1 }).limit(1)	}
Најди го жанрот на најновиот филм.	db.projectdata.aggregate([{ $project : { "_id":0 , "original_title" : 1 , "genres.name" : 1, "release_date" : 1}},{$sort : {"release_date": -1}},{ $limit : 1}])	**//map** function (doc) { emit(doc.release_date, {"original_title":doc.original_title,"genres":doc.genres.map(g => g.name)}); }
Најди ги 10-те најпопуларни филмови за секој жанр.	db.projectdata.aggregate([{ "$sort": { "popularity": -1} }, { "$group": { "_id": "$genres.name", "results": { "$push": { "original_title": "$original_title", "popularity": "$popularity", "id": "$id" }}},{ "$project": { "results": { "$slice": ["$results", 10] } }}])	**//map** function (doc) { if(doc.genres.map(g => g.name).includes("Action")){ emit(doc.popularity, {"original_title":doc.original_title,"genres":doc.genres.map(g => g.name)}); } }
Најди го бројот на филмови по соодветен јазик.	var mapFunc = function(){ var original_languages = this.original_language; emit(original_languages,1); } var reduFunc= function(key,values){ var count = 0; var i=0; for(i in values){ count += values[i]; } return count; }	**//map** function (doc) { if(doc.original_language){ emit(doc.original_language, 1); } } **//reduce** _count predefined in couchDB

	db.projectdata.mapReduce(mapFunc, reduFunc, {out: "map_ex_1"}) db.map_ex_1.find()	
Најди ги првите 10 најспоменувани production_companies заедно со бројот на споменувања.	db.projectdata.aggregate([{ $group : { "_id" : "$production_companies.name", "no" : { "$sum" : 1}}},{ $sort :{"production_companies.name" : -1}},{ $limit : 10}])	**//map** function (doc) { let charCodeS = 'S'.charCodeAt(0); let charCodeU = 'U'.charCodeAt(0); for(const companies of doc.production_companies){ let firstLetter = companies.name.toUpperCase().charAt(0); if(firstLetter.charCodeAt(0) >= charCodeS && firstLetter.charCodeAt(0) <= charCodeU){ emit(companies.name,1); } } } **//reduce** _count predefined in couchDB
Најди го најпопуларниот филм во продукција на Волт Дизни и провери го бројот на јазици	db.projectdata.aggregate([{ $match: {'production_companies.name':{ $regex: 'Walt Disney Productions' }}},{ "$sort": { "popularity": -1} },{ $limit: 1 },{$group:{_id:{title:'$original_	**//map** function (doc) { for(const companies of doc.production_companies){ if(companies.name == 'Walt Disney Productions'){

на кој се појавува истиот.	title'}, spoken_languages:{$sum:1}}},])	emit(doc.popularity,{"original_title":doc.original_title,"spoken_languages":doc.spoken_languages.length}); break; } } } **//reduce** None
Најди го најкраткиот филм објавен пред 2000 год.	db.projectdata.aggregate([{ $project : {"_id" :0, "original_title":1,"release_date":1, "runtime" : 1}}, {$match : { "release_date" : { "$lt" : "2000-01-01"}, "runtime": {"$nin" : [null, 0]}}},{ $sort : { "runtime" :1}},{ $limit : 1}])	**//map** function (doc) { if(doc.release_date.substr(0,4) < '2000'){ emit(doc.runtime, {"original_title":doc.original_title}); } } **//reduce** none
Најди го англискиот филм (original_language) со најголем vote_count.	db.projectdata.explain("executionStats").find({original_language:"en"},{original_title:1,_id:0,vote_count:1}).sort({vote_count:-1}).limit(1)	function(doc) { if (doc.original_language =='en') { emit(doc.vote_count,doc.original_language); } } curl http://admin:admin@localhost:5984/m

		ovies/_design/MoviesInfo/_view/query11?descending=true&limit=1
Врати ги буџетите на филмовите кои имаа vote_average > 5 и се во продукција на 'Pixar Animation Studios'.	db.projectdata.explain("executionStats").find({$and:[{vote_average:{$gt:5}},{"production_companies.name":"Pixar Animation Studios"}]},{original_title:1,production_companies:1,_id:0,budget:1,vote_average:1})	function (doc) { for(const companies of doc.production_companies){ if(companies.name == 'Pixar Animation Studios' && doc.vote_average > 5){ emit(doc.vote_average,{"original_title": doc.original_title, "budget":doc.budget}); break; } } }
Спореди го бројот на објавени филмови во 1999 година и 2001 година и врати ја годината во која имаме повеќе филмови.	db.projectdata.explain("executionStats").aggregate({ $match: { $or: [{ $and: [{ release_date: { $gte: "1999-01-01" } }, { release_date: { $lt: "2000-01-01" } }] }, { $and: [{ release_date: { $gte: "2001-01-01" } }, { release_date: { $lt: "2002-01-01" } }] }] } }, { $group: { _id: { $substr: ["$release_date", 0, 4] }, totalMovies: { $sum: 1 } } }, {	curl -X GET http://admin:admin@localhost:5984/movies/_design/MoviesInfo/_view/query13?limit=1 -G -d group=true **//map** function (doc) { if (doc.release_date.split("-")[0] == "1999" || doc.release_date.split("-")[0] =="2001") { emit(doc.release_date.split("-")[0], 1); } }

	$sort: { "totalMovies": -1 } }, { $limit: 1 })	**//Reduce** _sum
Спореди кој е попопуларен филм помеѓу најдолгиот и најкраткиот филм и врати го името на истиот.	var min = db.projectdata.find({runtime:{ $ne:null},popularity:{$ne:null}} ,{original_title:1,runtime:1,po pularity:1,_id:0}).sort({runtime :1}).limit(1).toArray() var max = db.projectdata.find({runtime:{ $ne:null},popularity:{$ne:null}} ,{original_title:1,runtime:1,po pularity:1,_id:0}).sort({runtime :-1}).limit(1).toArray() const cmpArr = [] cmpArr.push(min[0],max[0]) cmpArr.reduce((a, b) => {if (Math.max(a.popularity, b.popularity) === a.popularity) {return a;} else {return b;}})	**//map** function (doc) { if(doc.runtime != null){ emit(doc.runtime,{"popularity":doc.pop ularity, "title":doc.title}); } } **//reduce** function (keys, values, rereduce) { var najkratok = values[values.length-1]; var najdolg = values[0] if(najdolg.popularity > najkratok.popularity){ return {"title":najdolg.title} }else{ return {"title":najkratok.title} } }
Дали филмот со најголем vote_count е филмот со	const vote_count_max = db.projectdata.find({},{original _title:1,vote_count:1,popularit y:1,_id:0}).sort({vote_count:-1}).limit(1).toArray()	**//map** function (doc) { emit(doc.original_title, {"original_title":doc.original_title,"vote

најголема популарност	const popularity_max = db.projectdata.find({},{original_title:1,vote_count:1,popularity:1,_id:0}).sort({popularity:-1}).limit(1).toArray() vote_count_max[0].original_title === popularity_max[0].original_title	_count":doc.vote_count,"popularity":doc.popularity}); } **//reduce** function (keys, values, rereduce) { if (rereduce) { let all = []; for(let val in values){ values[val].forEach(v => all.push(v)); } let voteCountList = all.map(v => v.vote_count); let popularityList = all.map(v => v.popularity); voteCountList.sort(); popularityList.sort(); const voteCountMax = voteCountList[voteCountList.length-1]; const popularityMax = popularityList[popularityList.length-1]; all.filter(obj => obj.vote_count === voteCountMax && obj.popularity === popularityMax); let objVoteCountMax = all.filter(obj => obj.vote_count === voteCountMax)[0]; let objPopularityMax = all.filter(obj => obj.popularity === popularityMax)[0]; return {

		"maxVoteCountMovieTitle":objVoteCountMax.original_title, "maxPopularityMovieTitle":objPopularityMax.original_title, "is it equal": objVoteCountMax.original_title === objPopularityMax.original_title, }; } else { return values; } }
Дали филмот кој има најмал буџет е и најмалку гласан.	const vote_average_min = db.projectdata.explain("executionStats").find({$and:[{vote_average:{$ne:null}},{vote_average: {$ne:0}},{budget:{$ne:null}},{budget: {$ne:0}}]},{original_title:1,vote_average:1,budget:1,_id:0}).sort({vote_average:1}).limit(1) const budget_min = db.projectdata.explain("executionStats").find({$and:[{vote_average:{$ne:null}},{vote_avera	**//map** function (doc) { emit(doc.original_title, {"original_title":doc.original_title,"budget":doc.budget,"vote_avg":doc.vote_average}); } **// reduce** function (keys, values, rereduce) { if (rereduce) { let all = []; for(let val in values){ values[val].forEach(v => all.push(v));

	ge: {$ne:0}},{budget:{$ne:null}},{budget: {$ne:0}}]},{original_title:1,vote_average:1,budget:1,_id:0}).sort({budget:1}).limit(1) vote_average_min[0].original_title === budget_min[0].original_title	} let budgetList = all.map(v => v.budget); let voteAvgList = all.map(v => v.vote_average); budgetList.sort(); revenueList.sort(); const budgetMin = budgetList[0]; const voteAvgMin = voteAvgList[0]; all.filter(obj => obj.budget === budgetMin && obj.vote_average === voteAvgMin); let objBudgetMin = all.filter(obj => obj.budget === budgetMin)[0]; let objVoteAvgMin = all.filter(obj => obj.revenue === voteAvgMin)[0]; return { "minBudgetMovieTitle":objBudgetMin.original_title, "minVoteAvgMovieTitle":objVoteAvgMin.original_title, "is it equal": objBudgetMin.original_title === objVoteAvgMin.original_title, }; } else { return values;

		} }
Најди ги продукциските компании кои имаат најголем вкупен vote_average.	db.projectdata.explain("executionStats").aggregate({ $match: { $and: [{ production_companies: { $ne: null } }, { production_companies: { $ne: "[]" } }, { production_companies: { $ne: [] } }] } }, { $group: { _id: { "production_companies": "$production_companies" }, totalVoteAverage: { $sum: "$vote_average" } } }, { $sort: { totalVoteAverage: -1 } }, { $limit: 1 })	**//map** function(doc) { doc.production_companies.forEach(function (pcompany) { emit(pcompany.name,doc.vote_average); }); } **//reduce** _stats curl -X GET http://admin:admin@localhost:5984/movies_sub/_design/movieSubInfo/_view/query17?limit=1 -G -d group=true
Најди просек на заработката на филмовите според жанрот.	db.projectdata.explain("executionStats").aggregate({ $match: { $and: [{ genres: { $ne: null } }, { genres: { $ne: "[]" } }, { genres: { $ne: [] } }] } }, { $group: { _id: { "genres": "$genres.name" }, avgRevenue: { $avg: "$revenue" } } })	**//map** function (doc) { if (doc.genres) { emit(doc.genres[0].name, doc.revenue); } } **//reduce** function (keys, values, rereduce) { return sum(values)/values.length; } Time: 487ms

		curl -X GET http://admin:admin@localhost:5984/movies/_design/moviesInfo/_view/query18?limit=1 -G -d group=true
Прикажи ги бројот на филмови објавени на ист датум.	db.projectdata.explain("executionStats").aggregate({ $match: { $and: [{ release_date: { $ne: null } }, { release_date: { $ne: "" } }] } }, { $group: { _id: { "release_date": "$release_date" }, num_movies: { $sum: 1 } } })	curl http://admin:admin@localhost:5984/movies/_design/testMovies/_view/query19?descending=true&group=true **//map** function (doc) { if(doc.release_date){ emit(doc.release_date, 1); } } **//reduce** _count predefined in couchDB
Најди го најдолгиот филм објавен по 2000 година и пред 2003 година.	db.projectdata.explain("executionStats").find({ $and: [{ release_date: { $gte: "2001-01-01" } }, { release_date: { $lt: "2003-01-01" } }] },{original_title:1,runtime:1,release_date:1,_id:0}).sort({runtime:-1}).limit(1)	curl http://admin:admin@localhost:5984/movies/_design/testMovies/_view/query20?descending=true&limit=1 function (doc) { if(doc.release_date.substr(0,4) > '2000' && doc.release_date.substr(0,4) < '2003'){ emit(doc.runtime, {"original_title":doc.original_title}); } }

Со цел да истражиме дополнителни методи на агрегација и да експериментираме со корелациите помеѓу атрибутите, ние се одлучивме да допишеме дополнителни прашалници. Во овие прашалници повеќе фокус посветивме на различните нивоа на агрегација и групирање на различните карактеристики на податочното множество. Притоа, направивме промена на двете неструктурирани бази на податоци помеѓу тимовите, со цел сите да добиеме искуство и да научиме да работиме како со MongoDB, така и со CouchDB. Ова ни овозможи на подлабоко ниво да ги увидиме сличностите и разликите кои овие две нестуктурирани бази на податоци ги имаат. Притоа, самиот процес на премин од едната база, кон другата беше доста лесен, бидејќи и двете подлежат на истите правила и принципи.

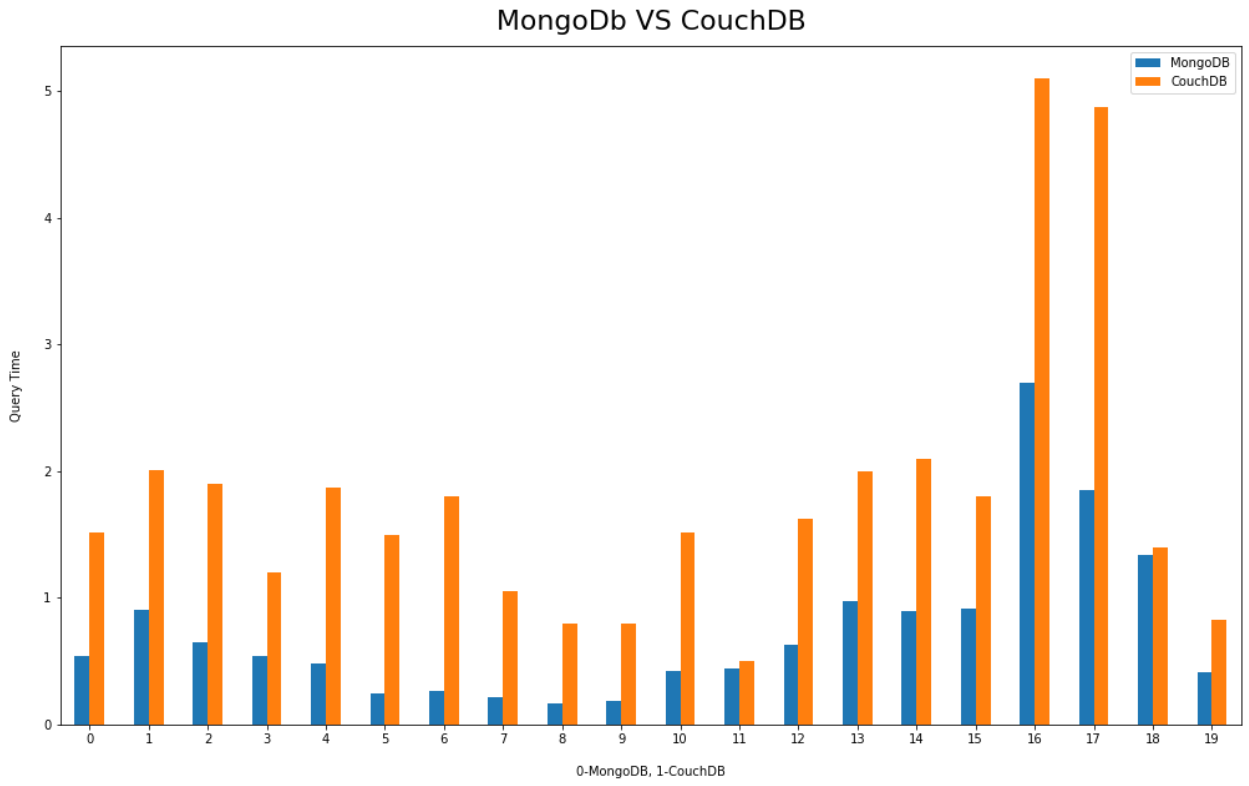
Процесот на импортирање на податоците е доста едноставен и кај двете бази на податоци, меѓутоа, пишувањето на прашалниците е во голема мера поедноставно и поинтуитивно кај MongoDB. Ова можеме да го заклучиме и од самата табела, прикажана погоре. Иако CouchDB нуди многу моќни начини на комбинирање и филтрирање на податоците, како и полесна скалабилност, сепак при пишување на прашалниците се соочуваме со многу препреки и предизвици, кои не се случај при користењето на MongoDB. MongoDB е синтаксички, како и логички многу поблиска до SQL базите на податоци, со кои поголем дел од луѓето се запознати.

Резултати

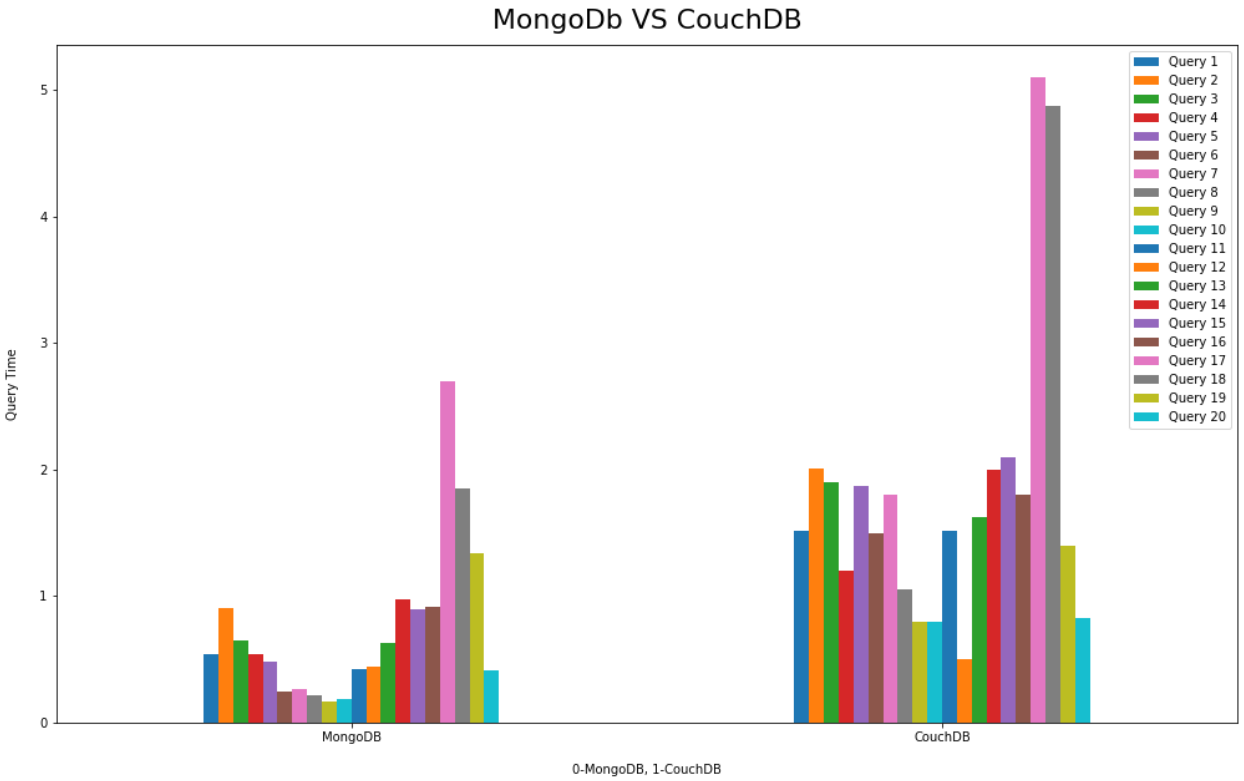
При извршувањето на прашалниците, паралелно извршувавме и мерења на времето потребно за добивање на резултат. Резултатите кои што ги добивме, се прикажани на графиконите во прилог.

	Query 1	Query 2	Query 3	Query 4	Query 5	Query 6	Query 7	Query 8	Query 9	Query 10	Query 11	Query 12	Query 13	Query 14	Query 15	Query 16	Query 17	Query 18	Query 19	Query 20
MongoDB	0.54	0.90	0.65	0.54	0.48	0.24	0.26	0.21	0.17	0.19	0.42	0.44	0.63	0.97	0.89	0.91	2.7	1.85	1.34	0.41
CouchDB	1.52	2.01	1.90	1.20	1.87	1.50	1.80	1.05	0.80	0.80	1.52	0.50	1.62	2.00	2.10	1.80	5.1	4.87	1.40	0.83

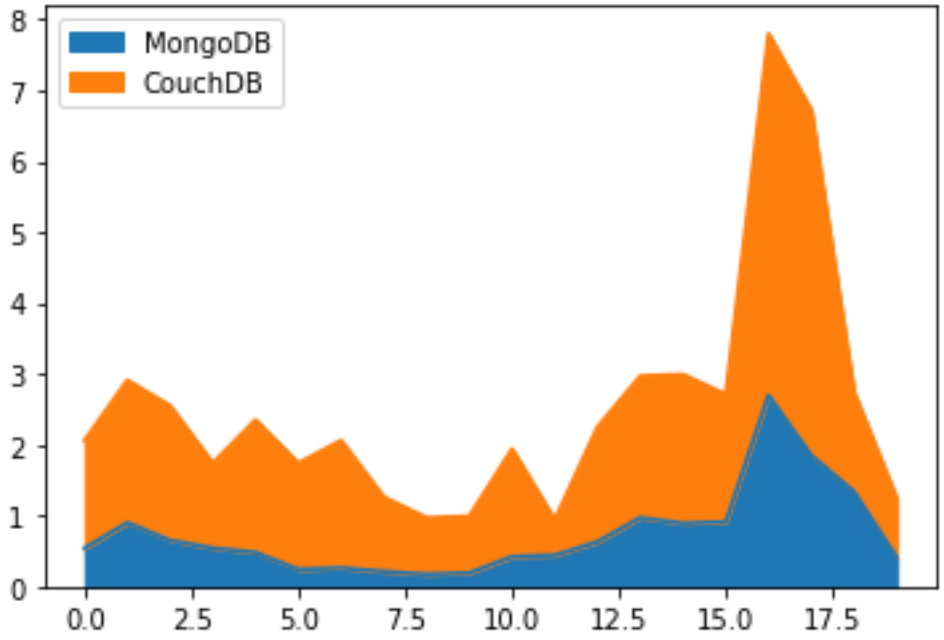
Слика 15. „Табеларен приказ на времињата на извршување на прашалниците“



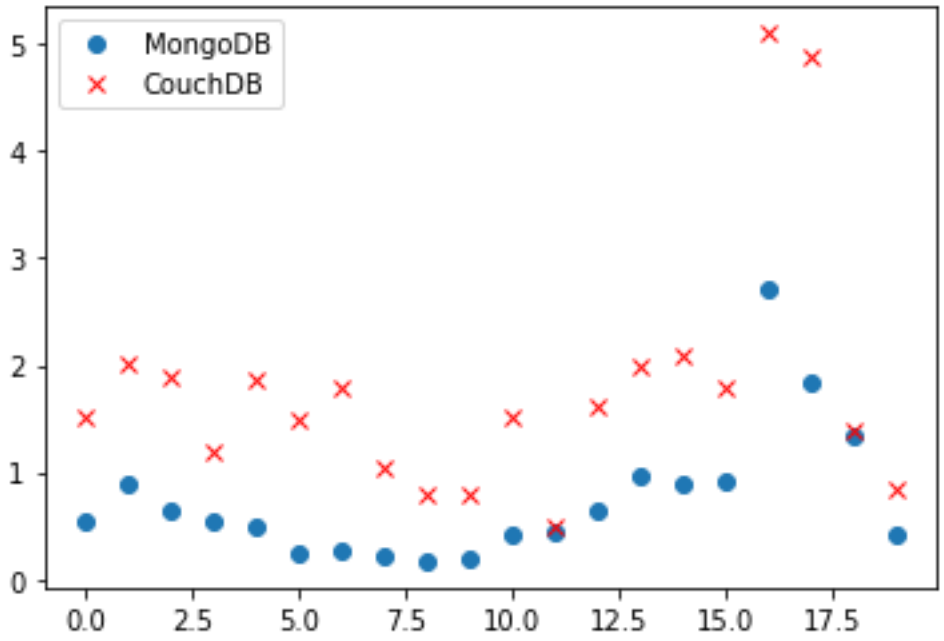
Слика 16. „Споредба на времињата на извршување на прашалниците“



Слика 17. „Споредба на времињата на извршување на прашалниците“

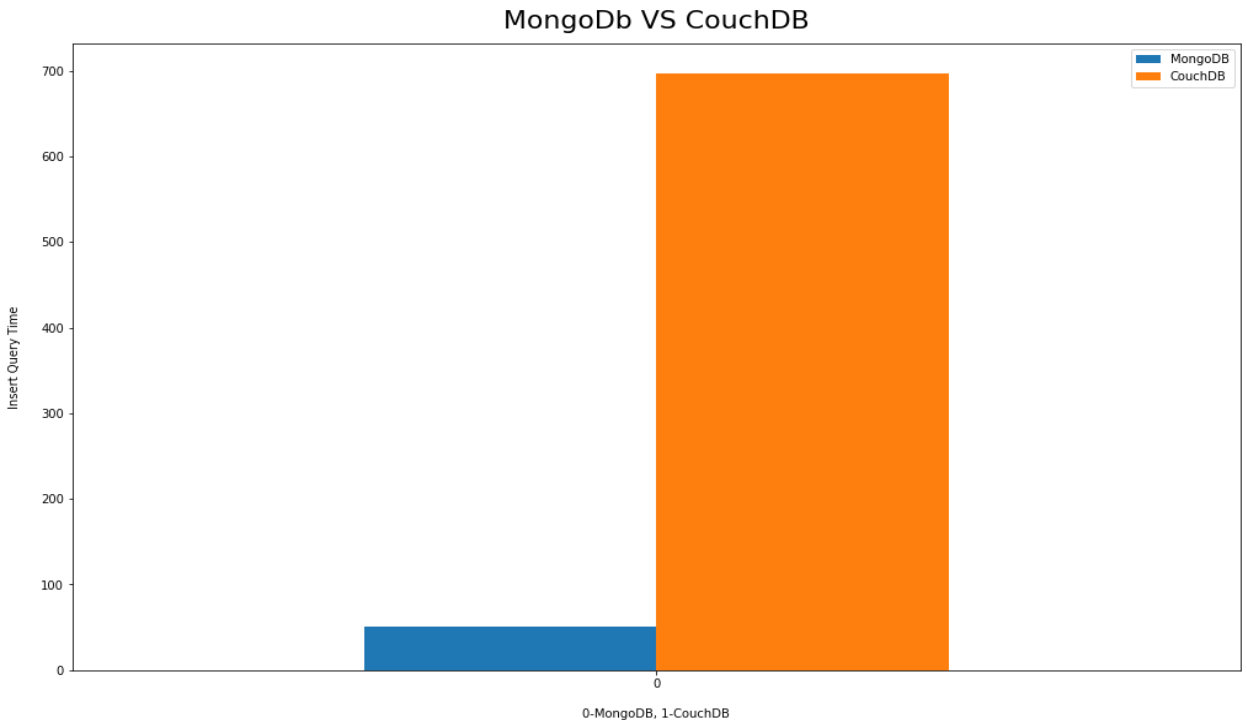


Слика 18. „Споредба на времињата на извршување на прашалниците“



Слика 19. „Споредба на времињата на извршување на прашалниците“

По направените визуелизации, а воедно и при самиот процес на извршување на прашалниците можевме да забележиме дека има доста голема разлика во потребното време на извршување на прашалниците на двете неструктурирани бази на податоци. Имено, MongoDB е евидентно подобар од гледна точка на брзина на извршување на различните операции. Притоа, со самото зголемување на нивото на агрегации кои ги користевме во прашалниците, временската разлика помеѓу потребните времиња на извршување кај двете неструктурирани бази на податоци стануваше се поголема. Односно, со секоја покомплицирана агрегација, времето на извршување на CouchDB растеше многу побрзо, во споредба со времето на извршување на MongoDB, кое многу побавно растеше.



Слика 20. „Споредба на времињата на извршување на додавање на документи во базите“

Покрај евидентно побрзото време на извршување на прашалниците во MongoDB, во споредба со CouchDB, од графикот прикажан на слика број 20 можеме да забележиме дека и времето на додавање на документите во овие две неструктурирани бази на податоци многу се разликува. Процесот на додавање на документите во CouchDB е дури единаесет пати побавен од истиот овој процес во MongoDB.

Заклучок

Со оглед на тоа што живееме во data-driven ера, ова истражување ни помогна да сфатиме колку е важен квалитетот, а не само квантитетот на податоците со кои работиме. Доколку не направевме претпроцесирање на самите податоци, немаше да можеме да креираме прашалници кои ќе извлечат значајни информации за самото множество, коишто носат одредена бизнис вредност во реалниот свет. Слободната форма на неструктурираните бази на податоци со себе носи и предизвици во поглед на разбирливоста на податоците, со коишто процесот на претпроцесирање добро се справува. Конкретно, ние успеавме да извлечеме одредени корелации од множеството на филмови со кое работевме, коишто пак значително ни помогнаа во следната фаза на формулирање на прашалниците.

Во поглед на имплементацијата на прашалниците во двете бази соодветно, се соочивме со одредени потешкотии, особено работејќи во CouchDB, што пак придонесе во увидување на разликите на двете бази. Така, MongoDB е поинтуитивна и поедноставна за користење, затоа што има синтакса доста поблиска до SQL, на којшто многумина се навикнати. Од друга страна, CouchDB можеби нуди многу поелегантни решенија, но е потребна перспектива во голема мера различна од традиционалните релациони бази на податоци.

Од аспект на перформанси на базите, прашалниците извршувани во MongoDB имаа многу пократко време на извршување во споредба со оние во CouchDB, што докажува дека не случајно MongoDB е моментално најкористената неструктурирана база на податоци. Но, сепак, интересно е да се забележи дека CouchDB може да се извршува на Apple iOS и Android уреди, нешто што MongoDB не го нуди. Така, доаѓаме до тоа дека секоја база има одредени предности коишто може да се соодветни за различни намени, во зависност од проектот којшто го развиваме.

Направената анализа прави невозможно да не се помисли на Големите податоци (Big Data) коишто стануваат сè повеќе актуелни во секоја сфера, а за кои релационите бази на податоци не се соодветни. Оттука, сите големи компании како Amazon, EBay, Netflix, Google,

Facebook коишто работат со големи количини на податоци користат неструктурирани бази на податоци, поради нивната скалабилност и разновидност. Одовде доаѓаме до две клучни дилеми, и тоа во врска со значењето на претпроцесирањето во процесот на извлекување на знаење од навидум неповрзани податоци, како и тоа дали неструктурираните бази на податоци целосно ќе ги заменат релационите во иднина.

Користена литература

- MongoDB official documentation
- Derek Banas: Mongo Tutorial
- Simplelearn: MongoDB Full Course
- MongoDB tutorial
- Python MongoDB
- The Battle of the NoSQL Databases - Comparing MongoDB and CouchDB
- CouchDB vs. MongoDB: What You Need to Know
- CouchDB official documentation
- Mark Grimes: CouchDB Tutorial
- Apache CouchDB
- CouchDB Tutorial
- CouchDB Fauxton
- Python CouchDB tutorial
- Difference between Couchdb and Mongodb
- Performance Evaluation of Query Response Time in The Document Stored NoSQL Database
- System Properties Comparison CouchDB vs. MongoDB
- CouchDB vs MongoDB - JavaTPoint
- Difference Between CouchDB vs MongoDB