Dokumentowe bazy danych – MongoDB

Ćwiczenie 2 - zadanie do samodzielnego wykonania

Imię i nazwisko:

Adrian Żerebiec

Materiały:

Książki

Np.

- Shannon Bradshaw, Eoin Brazil, Kristina Chodorow, MongoDB: The Definitive Guide. Powerful and Scalable Data Storage, O'Reily 2019
- Alex Giamas, Mastering MongoDB 4.x., Pact 2019

Dokumentacja

- https://www.mongodb.com/docs/manual/reference/program/mongo/

MongoDB University Courses

- https://university.mongodb.com/courses/catalog
- MongoDB Basics
  - https://university.mongodb.com/courses/M001/about
- The MongoDB Aggregation Framework
  - https://university.mongodb.com/courses/M121/about
- Data Modeling
  - https://university.mongodb.com/courses/M320/about

Yelp Dataset

www.yelp.com - serwis społecznościowy – informacje o miejscach/lokalach

- restauracje, kluby, hotele itd. (*businesses*),
- użytkownicy piszą recenzje (*reviews*) o miejscach i wystawiają oceny oceny,
- użytkownicy odwiedzają te miejsca - "meldują się" (*check-in*)
- Przykładowy zbiór danych zawiera dane z 5 miast: Phoenix, Las Vegas, Madison, Waterloo i Edinburgh.

Kolekcje:

- **42,153** businesses
- **320,002** business attributes
- **31,617** check-in sets
- **252,898** users
- **955,999** edge social graph
- **403,210** tips
- **1,125,458** reviews

business

```
{
    'type': 'business',
    'business_id': (encrypted business id),
    'name': (business name),
    'neighborhoods': [(hood names)],
    'full_address': (localized address),
    'city': (city),
    'state': (state),
    'latitude': latitude,
    'longitude': longitude,
    'stars': (star rating, rounded to half-stars),
    'review_count': review count,
    'categories': [(localized category names)]
    'open': True / False (corresponds to closed, not business hours),
    'hours': {
        (day_of_week): {
            'open': (HH:MM),
            'close': (HH:MM)
        },
        ...
    },
    'attributes': {
        (attribute_name): (attribute_value),
        ...
    },
}
```

review

```
{
    'type': 'review',
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'stars': (star rating, rounded to half-stars),
    'text': (review text),
    'date': (date, formatted like '2012-03-14'),
    'votes': {(vote type): (count)},
}
```

user

```
{
    'type': 'user',
    'user_id': (encrypted user id),
    'name': (first name),
    'review_count': (review count),
    'average_stars': (floating point average, like 4.31),
    'votes': {(vote type): (count)},
    'friends': [(friend user_ids)],
    'elite': [(years_elite)],
    'yelping_since': (date, formatted like '2012-03'),
    'compliments': {
        (compliment_type): (num_compliments_of_this_type),
        ...
    },
    'fans': (num_fans),
}
```

check-in

```
{
    'type': 'checkin',
    'business_id': (encrypted business id),
    'checkin_info': {
        '0-0': (number of checkins from 00:00 to 01:00 on all Sundays),
        '1-0': (number of checkins from 01:00 to 02:00 on all Sundays),
        ...
        '14-4': (number of checkins from 14:00 to 15:00 on all Thursdays),
        ...
        '23-6': (number of checkins from 23:00 to 00:00 on all Saturdays)
    }, # if there was no checkin for a hour-day block it will not be in the dict
}
```

tip

```
{
    'type': 'tip',
    'text': (tip text),
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'date': (date, formatted like '2012-03-14'),
    'likes': (count),
}
```

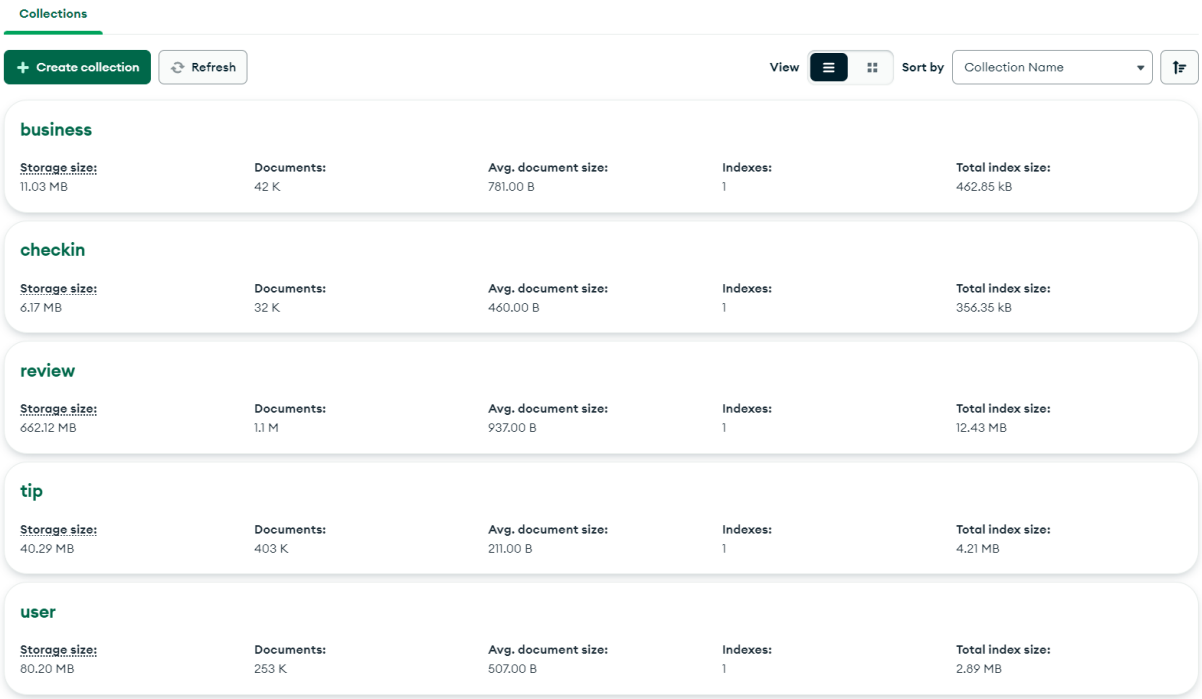
Zadania

1. Operacje wyszukiwania danych

Dla zbioru Yelp wykonaj następujące zapytania

W niektórych przypadkach może być potrzebne wykorzystanie mechanizmu Aggregation Pipeline https://www.mongodb.com/docs/manual/core/aggregation-pipeline/

Początkowo stworzyłem bazę YELP i dodałem potrzebne dane z pomocą MongoDB Compass. Z racji ograniczeń MongoDB Atlas baza była lokalna.



Poniższe zadania rozwiązywałem z wykorzystaniem rozszerzenia „MongoDB for VS Code v1.0.1” do programu Visual Studio Code. Z jego pomocą łatwo połączyłem się z bazą danych i dzięki tzw. Playground’owi pisałem polecenia dotyczące bazy danych YELP znajdującej się na mongodb://localhost:27017.

Wszystkie uzyskane wyniki zostały skopiowane do plików z rozszerzeniem .txt lub .json i są to wyniki uzyskane poprzez wywołanie funkcji. Pliki do zadania 1 znajdują się w katalogu „Zadanie1 – wyniki”.

a) Zwróć dane wszystkich restauracji (kolekcja businesss, pole *categories* musi zawierać wartość *Restaurants),* które są otwarte w poniedziałki (pole hours) i mają ocenę co najmniej 4 gwiazdki (pole *stars*). Zapytanie powinno zwracać: nazwę firmy, adres, kategorię, godziny otwarcia i gwiazdki. Posortuj wynik wg nazwy firmy.

…. Wyniki, zrzuty ekranów, kod, komentarz

Kod:

```
const zadanie1a = () => {
  const result = db.business.find(
    {
      'categories': 'Restaurants',
      'hours.Monday.open': { $exists: true },
      'stars': { $gte: 4 }
    },
    {
      'name': 1,
      'address': 1,
      'categories': 1,
      'hours.Monday': 1,
      'stars': 1,
      '_id': 0
    }
  ).sort({ 'name': 1 });

  // Tworzenie tablicy wyników
  const resultsArray = [];
  while (result.hasNext()) {
    const document = result.next();
    resultsArray.push(document);
  }

  // Zapisywanie wyników do pliku JSON
  const fs = require('fs');
  const fileName = 'Zadanie1a-wyniki.json';
  fs.writeFileSync(fileName, JSON.stringify(resultsArray, null, 2));
  console.log(`Wyniki zostały zapisane do pliku ${fileName}`);
};
```

Aby uzyskać wynik wybieramy tylko te dane których kategoria to restauracja, w poniedziałek istnieją godziny otwarcia, a ocena jest większa od 4. Dodatkowo sortujemy pod ‘name’.

Z racji iż wyników jest bardzo dużo prezentuję tylko kilka przykładów. Wszystkie uzyskane wyniki zapisałem do pliku wynik_zadanie1a.json a go przeniosłem do katalogu Zadanie 1 - wyniki.

Wyniki początkowe:


```
{
   "hours": {
     "Monday": {
       "close": "22:00",
       "open": "10:00"
     }
   },
   "categories": [
     "Food",
     "Desserts",
     "Coffee & Tea",
     "Indian",
     "Restaurants"
   ],
   "name": "10-to-10 In Delhi",
   "stars": 4.5
 },
 {
   "hours": {
     "Monday": {
       "close": "22:00",
       "open": "11:00"
     }
   },
   "categories": [
     "Vietnamese",
     "Asian Fusion",
     "French",
     "Restaurants"
   ],
   "name": "188 Restaurant",
   "stars": 4
 },
 {
   "hours": {
     "Monday": {
       "close": "05:00",
       "open": "18:00"
     }
   },
   "categories": [
     "Bars",
     "Asian Fusion",
     "Nightlife",
     "Lounges",
```

```
      "Karaoke",
      "Restaurants"
    ],
    "name": "21 Restaurant & Lounge",
    "stars": 4
  }
```


Wyniki końcowe:


```
  {
    "hours": {
      "Monday": {
        "close": "21:30",
        "open": "11:00"
      }
    },
    "categories": [
      "Italian",
      "Pizza",
      "Restaurants"
    ],
    "name": "il Capo Pizzeria",
    "stars": 4.5
  },
  {
    "hours": {
      "Monday": {
        "close": "21:00",
        "open": "06:00"
      }
    },
    "categories": [
      "Food",
      "Juice Bars & Smoothies",
      "Restaurants",
      "Cafes"
    ],
    "name": "s.e.e.d. cafe",
    "stars": 4.5
  },
  {
    "hours": {
      "Monday": {
        "close": "14:00",
        "open": "08:00"
      }
    },
    "categories": [
```

```
    "Sandwiches",
    "Restaurants"
  ],
  "name": "the Goodwich",
  "stars": 4.5
},
{
  "hours": {
    "Monday": {
      "close": "20:00",
      "open": "09:00"
    }
  },
  "categories": [
    "Vietnamese",
    "Restaurants"
  ],
  "name": "unPhogettable",
  "stars": 4
},
{
  "hours": {
    "Monday": {
      "close": "23:00",
      "open": "17:00"
    }
  },
  "categories": [
    "Spanish",
    "Restaurants"
  ],
  "name": "é by José Andrés",
  "stars": 4.5
}
```

b) Ile hoteli znajduje się w każdym mieście. (pole *categories* musi zawierać wartość *Hotels & Travel* lub *Hotels)*. Wynik powinien zawierać nazwę miasta, oraz liczbę hoteli. Posortuj wynik malejąco wg liczby hoteli.

…. Wyniki, zrzuty ekranów, kod, komentarz

Kod:

```
const zadanie1b = () => {
  const result = db.business.aggregate([
    {
      $match: {
        $or: [
          { categories: 'Hotels & Travel' },
          { categories: 'Hotels' }]
      }
    },
    {
      $group: {
        _id: '$city',
        count: { $sum: 1 }
      }
    },
    {
      $project: {
        city: '$_id',
        count: 1,
        _id: 0
      }
    },
    {
      $sort: {
        count: -1
      }
    }
  ]);
  while (result.hasNext()) {
    const document = result.next();
    console.log(`Miasto: ${document.city}, Liczba hoteli: ${document.count}`);
  }
};
```

Wybieramy tylko te miasta które mają odpowiednie kategorie. Grupujemy po miastach i sortujemy malejąco według licznika ‘count’.

Wyniki:

```
Miasto: Las Vegas, Liczba hoteli: 485
Miasto: Phoenix, Liczba hoteli: 250
Miasto: Edinburgh, Liczba hoteli: 161
Miasto: Scottsdale, Liczba hoteli: 122
Miasto: Madison, Liczba hoteli: 67
Miasto: Tempe, Liczba hoteli: 57
Miasto: Mesa, Liczba hoteli: 53
Miasto: Henderson, Liczba hoteli: 41
Miasto: Chandler, Liczba hoteli: 30
Miasto: Glendale, Liczba hoteli: 20
Miasto: North Las Vegas, Liczba hoteli: 12
Miasto: Peoria, Liczba hoteli: 12
Miasto: Surprise, Liczba hoteli: 10
Miasto: Goodyear, Liczba hoteli: 9
Miasto: Casa Grande, Liczba hoteli: 7
Miasto: Middleton, Liczba hoteli: 7
Miasto: Gila Bend, Liczba hoteli: 6
Miasto: Wickenburg, Liczba hoteli: 5
Miasto: Kitchener, Liczba hoteli: 5
Miasto: Avondale, Liczba hoteli: 5
Miasto: Waterloo, Liczba hoteli: 5
Miasto: Fountain Hills, Liczba hoteli: 4
Miasto: Gilbert, Liczba hoteli: 4
Miasto: Paradise Valley, Liczba hoteli: 3
Miasto: Tolleson, Liczba hoteli: 3
Miasto: Cave Creek, Liczba hoteli: 3
Miasto: Apache Junction, Liczba hoteli: 3
Miasto: Florence, Liczba hoteli: 3
Miasto: Lasswade, Liczba hoteli: 2
Miasto: Carefree, Liczba hoteli: 2
Miasto: Litchfield Park, Liczba hoteli: 2
Miasto: Fitchburg, Liczba hoteli: 2
Miasto: Queen Creek, Liczba hoteli: 2
Miasto: Cambridge, Liczba hoteli: 2
Miasto: South Queensferry, Liczba hoteli: 2
Miasto: Clark County, Liczba hoteli: 2
Miasto: Gold Canyon, Liczba hoteli: 2
Miasto: Anthem, Liczba hoteli: 2
Miasto: Goldfield, Liczba hoteli: 1
Miasto: DeForest, Liczba hoteli: 1
Miasto: Stanfield, Liczba hoteli: 1
Miasto: Youngtown, Liczba hoteli: 1
Miasto: Monona, Liczba hoteli: 1
Miasto: N Las Vegas, Liczba hoteli: 1
Miasto: Boulder City, Liczba hoteli: 1
Miasto: Mc Farland, Liczba hoteli: 1
Miasto: Tonto Basin, Liczba hoteli: 1
```

```
Miasto: Verona, Liczba hoteli: 1
Miasto: Buckeye, Liczba hoteli: 1
Miasto: Enterprise, Liczba hoteli: 1
Miasto: Tortilla Flat, Liczba hoteli: 1
Miasto: Bonnyrigg, Liczba hoteli: 1
Miasto: Coolidge, Liczba hoteli: 1
Miasto: Fort McDowell, Liczba hoteli: 1
Miasto: Laveen, Liczba hoteli: 1
Miasto: Maricopa, Liczba hoteli: 1
Miasto: Ratho, Liczba hoteli: 1
Miasto: Sun City, Liczba hoteli: 1
Miasto: Heidelberg, Liczba hoteli: 1
Miasto: Woolwich, Liczba hoteli: 1
Miasto: Deforest, Liczba hoteli: 1
Miasto: Waddell, Liczba hoteli: 1
Miasto: Inverkeithing, Liczba hoteli: 1
Miasto: Dalkeith, Liczba hoteli: 1
```

c) Ile każda firma otrzymała ocen/wskazówek (kolekcja *tip)* w 2012. Wynik powinien zawierać nazwę firmy oraz liczbę ocen/wskazówek Wynik posortuj według liczby wskazówek (*tip*).

…. Wyniki, zrzuty ekranów, kod, komentarz

Kod:

```
const zadanie1c = () => {
  const result = db.tip.aggregate([
    {
      $match:
      {
        'date':
        {
          $regex: /^2012/
        }
      }
    },
    {
      $group:
      {
        '_id': '$business_id',
        'tip_count':
        {
          $count: {}
        }
      }
    },
    {
      $lookup: {
        from: 'business',
```

```
        localField: '_id',
        foreignField: 'business_id',
        as: 'business'
      }
    },
    { $sort: { 'tip_count': -1 } },
    {
      $project: {
        'business_name': { $first: '$business.name' },
        'tip_count': 1
      }
    }
  ]);

  while (result.hasNext()) {
    const document = result.next();
    console.log('Firma:', document.business_name);
    console.log('Liczba ocen/wskazówek:', document.tip_count);
    console.log('---');}};
```

Wybieramy tylko te których data zaczyna się od 2012 grupujemy, zliczamy a na koniec sortujemy malejąco i zwracamy nazwę oraz liczbę ocen czyli tip_count.

Wynik:

Z racji, iż wyników jest dużo w sprawozdaniu zamieszczam te z początku

```
Firma:
McCarran International Airport
Liczba ocen/wskazówek:
1084
---
Firma:
Phoenix Sky Harbor International Airport
Liczba ocen/wskazówek:
622
---
Firma:
Earl of Sandwich
Liczba ocen/wskazówek:
430
---
Firma:
Las Vegas Athletic Club Southwest
Liczba ocen/wskazówek:
374
---
Firma:
The Cosmopolitan of Las Vegas
```

```
Liczba ocen/wskazówek:
351
---
Firma:
Wicked Spoon
Liczba ocen/wskazówek:
347
---
Firma:
Sushi House Goyemon
Liczba ocen/wskazówek:
258
---
Firma:
Pho Kim Long
Liczba ocen/wskazówek:
252
---
Firma:
Secret Pizza
Liczba ocen/wskazówek:
239
---
```

d) Recenzje mogą być oceniane przez innych użytkowników jako *cool, funny* lub *useful* (kolekcja *review*, pole *votes*, jedna recenzja może mieć kilka głosów w każdej kategorii). Napisz zapytanie, które zwraca dla każdej z tych kategorii, ile sumarycznie recenzji zostało oznaczonych przez te kategorie (np. recenzja ma kategorię *funny* jeśli co najmniej jedna osoba zagłosowała w ten sposób na daną recenzję)

…. Wyniki, zrzuty ekranów, kod, komentarz

Kod:

```
const zadanie1d = () => {
  const result = db.review.aggregate([
    {
      $group: {
        _id: null,
        cool_count: { $sum: { $cond: [{ $gt: ['$votes.cool', 0] }, 1, 0] } },
        funny_count: { $sum: { $cond: [{ $gt: ['$votes.funny', 0] }, 1, 0] }
},
        useful_count: { $sum: { $cond: [{ $gt: ['$votes.useful', 0] }, 1, 0] }
}
      }
    },
    {
      $project: {
        _id: 0
      }
    }
  ]);

  const output = result.toArray()[0];

  console.log('Liczba recenzji oznaczonych jako "cool":', output.cool_count);
  console.log('Liczba recenzji oznaczonych jako "funny":',
output.funny_count);
  console.log('Liczba recenzji oznaczonych jako "useful":',
output.useful_count);
};
```

Grupujemy i dla każdej z grup przeliczamy liczbę wystąpień, tam gdzie są większe od 0. Na koniec wykluczamy _id ze zbioru wynikowego.

Wyniki:

```
Liczba recenzji oznaczonych jako "cool":
346519
Liczba recenzji oznaczonych jako "funny":
269256
Liczba recenzji oznaczonych jako "useful":
549519
```

e) Zwróć dane wszystkich użytkowników (kolekcja *user*), którzy nie mają ani jednego pozytywnego głosu (pole *votes*) z kategorii (*funny lub useful*), wynik posortuj alfabetycznie według nazwy użytkownika.

…. Wyniki, zrzuty ekranów, kod, komentarz

Kod:

```
const zadanie1e = () => {
  const result = db.user.find(
    {
      $and: [
        { 'votes.funny': { $eq: 0 } },
        { 'votes.useful': { $eq: 0 } }
      ]
    }
  ).sort({ name: 1 });


  while (result.hasNext()) {
    const document = result.next();
    console.log('Użytkownik:', document.name);
    console.log('Yelping Since:', document.yelping_since);
    console.log('Review Count:', document.review_count);
    console.log('User ID:', document.user_id);
    console.log('Friends:', document.friends);
    console.log('Fans:', document.fans);
    console.log('Average Stars:', document.average_stars);
    console.log('Type:', document.type);
    console.log('Compliments:', document.compliments);
    console.log('Elite:', document.elite);

    if (document.votes) {
      console.log('Votes:');
      Object.keys(document.votes).forEach(category => {
        console.log(`  ${category}: ${document.votes[category]}`);
      });
    } else {
```

```
      console.log('Votes: None');
    }

    console.log('---');
  }
};
```

Wybieramy te gdzie wartości funny oraz useful są równe 0 a następnie sortujemy.

Wynik:

Z racji iż nazwy użytkownika są nieco dziwnie wprowadzone, stąd widzimy na pierwszym miejscu użytkownika „ Bernard” a na drugim „,Maria”, jednak jak widzimy dalej sortowanie odbywa się po nazwach

```
Użytkownik:
 Bernard
Yelping Since:
2009-08
Review Count:
1
User ID:
xP3SPgfgW2vc5Zj5uV8SEA
Friends:

Fans:
0
Average Stars:
5
Type:
user
Compliments:
[object Object]
Elite:

Votes:
  funny: 0
  useful: 0
  cool: 0
---
Użytkownik:
,Maria
Yelping Since:
2013-03
Review Count:
1
User ID:
Os5f3TNpM7_A8IDNEdPX2g
```

```
Friends:

Fans:
0
Average Stars:
5
Type:
user
Compliments:
[object Object]
Elite:

Votes:
  funny: 0
  useful: 0
  cool: 0
```

Inny fragment wyników:

```
Użytkownik:
wilson
Yelping Since:
2007-08
Review Count:
1
User ID:
pERmc2YYUbtuZkOxIABi4A
Friends:
AIsUl-LOictvrwMFO8Lzeg,8YBy2nz_5A-ciPidTCPo6Q
Fans:
0
Average Stars:
1
Type:
user
Compliments:
[object Object]
Elite:

Votes:
  funny: 0
  useful: 0
  cool: 0
---
Użytkownik:
yasmin
Yelping Since:
2009-06
```

```
Review Count:
1
User ID:
3rmkb3icxtqLCF3utv1-AA
Friends:

Fans:
0
Average Stars:
5
Type:
user
Compliments:
[object Object]
Elite:

Votes:
  funny: 0
  useful: 0
  cool: 0
---
Użytkownik:
yaze
Yelping Since:
2011-08
Review Count:
5
User ID:
kad5Df3fL5pASggozOG4lQ
Friends:

Fans:
0
Average Stars:
4.5
Type:
user
Compliments:
[object Object]
Elite:

Votes:
  funny: 0
  useful: 0
  cool: 0
```

f) Wyznacz, jaką średnia ocenę uzyskała każda firma na podstawie wszystkich recenzji (kolekcja *review*, pole *stars*). Ogranicz do firm, które uzyskały średnią powyżej 3 gwiazdek.

przypadek 1: Wynik powinien zawierać id firmy oraz średnią ocenę. Posortuj wynik wg id firmy.

przypadek 2: Wynik powinien zawierać nazwę firmy oraz średnią ocenę. Posortuj wynik wg nazwy firmy.

…. Wyniki, zrzuty ekranów, kod, komentarz

Przypadek 1:

Kod:

```
const zadanie1fa = () => {
  const result = db.review.aggregate([
    {
      $group: {
        _id: '$business_id',
        average_stars: { $avg: '$stars' }
      }
    },
    {
      $match: {
        average_stars: { $gt: 3 }
      }
    },
    {
      $sort: {
        _id: 1
      }
    },
    {
      $limit: 500 // Dodanie limitu 500 wyników
    }
  ]);

  console.log('Firmy z oceną powyżej 3 gwiazdek posortowane po ID:');
  while (result.hasNext()) {
    const document = result.next();

    console.log('ID firmy:', document._id);
    console.log('Średnia ocena:', document.average_stars);
    console.log('---');
  }
};
```

Grupujemy oraz przeliczamy wartości średnie, odrzucamy te których średnia jest mniejsza od 3 oraz sortujemy po _id. Jednocześnie dla szybkości obliczeń ustawiłem limit na 500, gdyż zbiór danych jest bardzo duży.

Wynik:

```
Firmy z oceną powyżej 3 gwiazdek posortowane po ID:
ID firmy:
--1emggGHgoG6ipd_RMb-g
Średnia ocena:
3.75
---
ID firmy:
--5jkZ3-nUPZxUvtcbr8Uw
Średnia ocena:
4.615384615384615
---
ID firmy:
--BlvDO_RG2yElKu9XA1_g
Średnia ocena:
3.9696969696969697
---
ID firmy:
--Dl2rW_xO8GuYBomlg9zw
Średnia ocena:
4.166666666666667
---
ID firmy:
--Ol5mVSMaW8ExtmWRUmKA
Średnia ocena:
5
---
ID firmy:
--XBxRlD92RaV6TyUnP8Ow
Średnia ocena:
3.6666666666666665
---
ID firmy:
--jFTZmywe7StuZ2hEjxyA
Średnia ocena:
4.333333333333333
---
ID firmy:
--m1g9P1wxNblrLANfVqlA
Średnia ocena:
4.25
---
ID firmy:
--qeSYxyn62mMjWvznNTdg
Średnia ocena:
4
---
```

Przypadek 2:

Kod:

```
const zadanie1fb = () => {
  const result = db.review.aggregate([
    {
      $match: {
        business_id: { $exists: true }
      }
    },
    {
      $group: {
        _id: '$business_id',
        average_stars: { $avg: '$stars' }
      }
    },
    {
      $match: {
        average_stars: { $gt: 3 }
      }
    },
    {
      $limit: 500
    },
    {
      $lookup: {
        from: 'business',
        localField: '_id',
        foreignField: 'business_id',
        as: 'business'
      }
    },
    {
      $unwind: '$business'
    },
    {
      $project: {
        _id: 0,
        business_name: '$business.name',
        average_stars: 1
      }
    },
    {
      $sort: {
        business_name: 1
      }
    }
  ]);
```

```
  console.log('Firmy z oceną powyżej 3 gwiazdek posortowane po nazwie:');
  while (result.hasNext()) {
    const document = result.next();
    console.log('Nazwa firmy:', document.business_name);
    console.log('Średnia ocena:', document.average_stars);
    console.log('---');
  }
};
```

Bardzo podobnie jak w wariancie 1, lecz tym razem musimy jeszcze połączyć kolekcje, aby uzyskać nazwy. Do tego wykorzystujemy $lookup.

Wynik:

```
Firmy z oceną powyżej 3 gwiazdek posortowane po nazwie:
Nazwa firmy:
2J Tech
Średnia ocena:
5
---
Nazwa firmy:
4 Real Intimate Apparel
Średnia ocena:
4.833333333333333
---
Nazwa firmy:
5th and Wine
Średnia ocena:
3.99438202247191
---
Nazwa firmy:
7-Eleven
Średnia ocena:
3.75
---
Nazwa firmy:
7-Eleven
Średnia ocena:
5
---
Nazwa firmy:
7-Eleven
Średnia ocena:
3.5
---
Nazwa firmy:
99 Cent Only Stores
Średnia ocena:
```

```
4.333333333333333
---
Nazwa firmy:
A Tailor
Średnia ocena:
5
---
Nazwa firmy:
A Taste of Heaven Cafe and Bakery
Średnia ocena:
4.333333333333333
---
Nazwa firmy:
Adair Fern Conservatory of the Arts
Średnia ocena:
3.3333333333333335
---
Nazwa firmy:
Adaven Children's Dentistry
Średnia ocena:
5
---
Nazwa firmy:
Ajo Al's Mexican Cafe
Średnia ocena:
3.328767123287671
---
```

W sprawozdaniu należy umieścić zrzuty ekranów (z kodem poleceń oraz z uzyskanymi wynikami). Dodatkowo należy dołączyć plik tekstowy (najlepiej z rozszerzeniem .js) zawierający kod poleceń

2. **Modelowanie danych**
    - Zaproponuj strukturę bazy danych dla wybranego/przykładowego zagadnienia/problemu
    - Należy wybrać jedno zagadnienie/problem (A lub B)

    Przykład A

    - Wykładowcy, przedmioty, studenci, oceny
    - Wykładowcy prowadzą zajęcia z poszczególnych przedmiotów
    - Studenci uczęszczają na zajęcia
    - Wykładowcy wystawiają oceny studentom
    - Studenci oceniają zajęcia

    Przykład B

    - Firmy, wycieczki, osoby
    - Firmy organizują wycieczki
    - Osoby rezerwują miejsca/wykupują bilety
    - Osoby oceniają wycieczki

    a) Warto zaproponować/rozważyć różne warianty struktury bazy danych i dokumentów w poszczególnych kolekcjach oraz przeprowadzić dyskusję każdego wariantu (wskazać wady i zalety każdego z wariantów)
    b) Kolekcje należy wypełnić przykładowymi danymi
    c) W kontekście zaprezentowania wad/zalet należy zaprezentować kilka przykładów/zapytań/zadań/operacji oraz dla których dedykowany jest dany wariantów

    …. Wyniki, zrzuty ekranów, kod, komentarz

Podejście tabelaryczne

Zalety

Takie podejście pozwala nam na uprządkowanie dokumentów w przejrzysty sposób poprzez pogrupowanie ich w kolekcje odpowiadającym różnym kategoriom. Dodatkowo takie podejście zapewniam nam większą wydajność podczas zapytań do osobno do każdego z dokumentów. Przydatne jest również to, iż dzięki temu możemy zachować mały rozmiar dokumentów, co pozwoli nam nie przekroczyć limitów gdy mamy bardzo dużo danych. Co więcej możemy szybko aktualizować dane zawarte w dokumentach. Pomaga to szczególnie w sytuacji, gdy dużo więcej zapisujemy danych w bazie niż ich odczytujemy.

Wady

Podejście tabelaryczne ma jednak wady. Aby uzyskać dane znajdujące się w różnych kolekcjach, musimy łączyć dane z dokumentów. Wiąże się to z koniecznością tworzenia skomplikowanych zapytań, jeśli na wynikowy dokument ma składać wiele dokumentów z różnych kolekcji. Wpływa to również na czas odczytu, gdyż wymagane jest złącznie dokumentów z kilku kolekcji.

Podejście dokumentowe

Zalety

Podejście dokumentowe zapewnia nam większą wydajność podczas gdy potrzebujemy całego dokumentu wraz z zagnieżdżonymi w nim dokumentami. Wtedy realnie potrzebujemy tylko jednego zapytania. W takim podejściu także łatwiej zdefiniować relację między dokumentami. Wystarczy tylko zagnieździć dokumenty będące w relacji. Szybciej możemy również odczytywać dane z bazy co przydatne jest, gdy nie musimy w niej dużo zapisywać ale dużo odczytywać.

Wady

Podejście to jednak również ma kilka wad. Nie mamy w nim możliwości tworzenia zapytań, które dotyczą zagnieżdżonych dokumentów. Przez to musimy dostać się do nich poprzez dokument, w którym są zagnieżdżone co powoduje, że zapytania stają się bardzo skomplikowane. Również o wiele trudniej przez to aktualizować naszą bazę danych, gdyż wymagane jest przejście przez kolejno zagnieżdżone dokumenty.

Baza danych

Jako zadanie do implementacji wybrałem Przykład B.

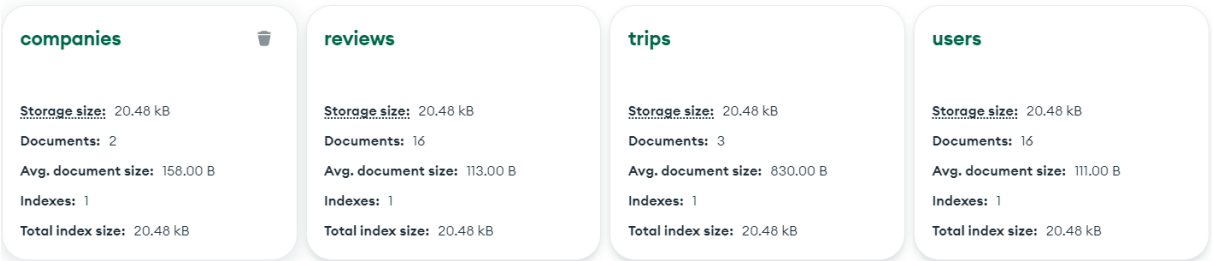
Analogicznie jak dla zadania 1, początkowo stworzyłem lokalną bazę danych ‘Trips-AZ’ z pomocą MongoDB Compass. Dodałem także 4 kolekcje:

- trips – zawiera informacje na temat wycieczek,

- users – zawiera informacje o uczestnikach wycieczek, przewodnikach itp.,

- reviews – zawiera opinie użytkowników na temat wycieczek,

- companies – zawiera informacje o firmach organizujących wycieczki.



Dodając dane ustawiałem również samodzielnie indeksy, aby zapewnić relacje w bazie danych. Indeksy stworzone zostały jako ciąg 24 znaków zapisanych w systemie szesnastkowym. Do pomocy w ich generowaniu wykorzystałem krótki skrypt w języku Python, który wypisuje klucz.

Skrypt:

```python
import secrets
import string

def generate_key():
    characters = string.hexdigits[:-6]
    key = ''.join(secrets.choice(characters) for _ in range(24))
    return key

# Wygenerowanie klucza
key = generate_key()
print(key)
```

Kolekcja Trips

W kolekcji znajdują się informacje na temat wycieczki takie jak:

- tripName - nazwa wycieczki,

- company – firma organizująca wycieczkę,

- duration – czas trwania wycieczki w dniach,

- groupSize – wielkość grupy osób,

-rating – ocena na podstawie opinii,

- numberOfOpinions – liczba opinii na temat wycieczki,

- price – cena wycieczki,

- description – opis wycieczki,

- startingDates – daty rozpoczęcia,

- details – miejsce, czas pobytu w poszczególnych lokacjach oraz odwiedzane restauracje,

- guides – lista przewodników.

Poniżej zamieszczam przykładowy dokument:


```json
{
  "_id": {
    "$oid": "3c88db8cf2afda39709c295a"
  },
  "company": {
    "$oid": "24718c38a4c1e7bbd92e3f21"
  },
  "tripName": "Paris in the summer",
  "duration": 7,
  "groupSize": 20,
  "rating": 4,
  "numberOfOpinions": 6,
  "price": 600,
```

```
  "description": "Paris is best city in the world. You can see Eiffel Tower,
Louvre and other famous places.",
  "startingDates": [
    "2023-06-11",
    "2023-07-21"
  ],
  "details": [
    {
      "place": "Paris",
      "accommodation": "Hotel Paris",
      "day": 5,
      "restaurants": [
        {
          "name": "Le Jules Verne",
          "cuisine": "French"
        },
        {
          "name": "Chez L'Ami Jean",
          "cuisine": "Italian"
        }
      ]
    },
    {
      "place": "Versailles",
      "accommodation": "Hotel France",
      "day": 1,
      "restaurants": [
        {
          "name": "L'Orangerie",
          "cuisine": "French"
        }
      ]
    },
    {
      "place": "Disneyland Paris",
      "accommodation": "Hotel Mickey",
      "day": 1,
      "restaurants": [
        {
          "name": "Auberge de Cendrillon",
          "cuisine": "French"
        }
      ]
    }
  ],
  "guides": [
    {
      "$oid": "6abd2ba0cefae0c6043562cf"
    }]}
```

Kolekcja users

Kolekcja zawiera informacje na temat użytkowników takie jak:

- name – imię i nazwisko osoby,

- email – email danej osoby,

- phone – numer telefonu danej osoby,

- role – rola danej osoby.

Poniżej przykładowy dokument:


```
{
  "_id": {
    "$oid": "30651955b1df2b47b4f5fa5c"
  },
  "name": "Shawn Michaels",
  "email": "hbk@gmail.com",
  "phone": "369258147",
  "role": "user"
}
```


Kolekcja reviews

Kolekcja zawiera:

- trip – wycieczka które dotyczy opinia,

- opinion – opinia odnośnie wycieczki,

- rating – ocena wycieczki,

- user – użytkownik, który dodał opinie.

Poniżej przykład:


```
{
  "_id": {
    "$oid": "8bcda4c7866ee28f06133b8c"
  },
  "trip": {
    "$oid": "3c88db8cf2afda39709c295a"
  },
  "opinion": "Not bad at all",
  "rating": 4,
  "user": {
    "$oid": "5c8a1dfa2f8fb814b56fa181"
  }
}
```

Kolekcja companies

Kolekcja zawiera informacje na temat firm takie jak:

- companyName – nazwa firmy,

- address – adres firmy,

- owner – właściciel firmy,

- trips – obsługiwane wycieczki.

Przykład:

```
{
  "_id": {
    "$oid": "0a60ccbc93d4291fedae1b4f"
  },
  "companyName": "Tripex",
  "address": "Lonly Street 1, 00-021, Praga, Czech Republic",
  "owner": {
    "$oid": "41a74e9caa34de98feb0f486"
  },
  "trips": [
    {
      "$oid": "6c78fb8cf4afda39709c2955"
    },
    {
      "$oid": "7a18fadcf4afda39709c295d"
    }
  ]
}
```

Operacje na bazie danych - zapytania:

Użytkownicy posortowani malejąco według liczby recenzji

```
const getUserReviewCounts = () => {
  const pipeline = [
    {
      $group: {
        _id: "$user",
        count: { $sum: 1 }
      }
    },
    {
      $sort: { count: -1 }
    }
  ];
```

```
  const result = db.reviews.aggregate(pipeline);

  print("User Review Counts:");

  while (result.hasNext()) {
    const document = result.next();
    console.log("User:", JSON.stringify(document._id));
    console.log("Count:", document.count);
    console.log("---");
  }
};
```

Wynik:

```
User Review Counts:
User:
"37a270d6088f2283a3f7df0e"
Count:
2
---
User:
"9e136739c439c0048dea81be"
Count:
2
---
User:
"b96f420ff60165da52ad9310"
Count:
2
---
User:
"11df1a504e6f3b241c117adc"
Count:
2
---
User:
"5c8a1dfa2f8fb814b56fa181"
Count:
2
---
User:
"532c022a8c71eae5e8d084c8"
Count:
1
---
User:
"a190bf57d67909284561adeb"
```

```
Count:
1
---
User:
"3f752fb23a76d57721249a04"
Count:
1
---
User:
"30651955b1df2b47b4f5fa5c"
Count:
1
---
User:
"57381942300d7dca5ecaa69e"
Count:
1
---
User:
"5c8a24402f8fb814b56fa190"
Count:
1
---
```

Informacje na temat wycieczek, których średnia ocena jest większa od 3.6

```
const getTripStats = () => {
  const pipeline = [
    { $match: { rating: { $gte: 3.6 } } },
    {
      $group: {
        _id: null,
        numberOFTrips: { $sum: 1 },
        numberOfOpinions: { $sum: '$numberOfOpinions' },
        rating: { $avg: '$rating' },
        avgPrice: { $avg: '$price' },
        minPrice: { $min: '$price' },
        maxPrice: { $max: '$price' }
      }
    }
  ];

  const result = db.trips.aggregate(pipeline);

  print("Trip Statistics:");

  while (result.hasNext()) {
```

```
    const document = result.next();
    console.log("Number of Trips:", document.numberOFTrips);
    console.log("Number of Opinions:", document.numberOfOpinions);
    console.log("Average Rating:", document.rating);
    console.log("Average Price:", document.avgPrice);
    console.log("Minimum Price:", document.minPrice);
    console.log("Maximum Price:", document.maxPrice);
  }
};
```

Wynik:

```
Trip Statistics:
Number of Trips:
2
Number of Opinions:
11
Average Rating:
3.8
Average Price:
1050
Minimum Price:
600
Maximum Price:
1500
```

Wycieczki odbywające się w tegoroczne wakacje

```
const getTripsByDateRange = () => {
  const pipeline = [
    {
      $match: {
        startingDates: {
          $gte: '2023-07-01',
          $lte: '2023-08-31'
        }
      }
    },
    {
      $project: {
        tripName: 1,
        startingDates: 1
      }
    }
  ];

  const result = db.trips.aggregate(pipeline);
```

```
  print("Trips by Date Range:");

  while (result.hasNext()) {
    const document = result.next();
    console.log("Trip Name:", document.tripName);
    console.log("Starting Date: ", document.startingDates);
    console.log("---");
  }
};
```

Wynik:

```
Trips by Date Range:
Trip Name:
Paris in the summer
Starting Date:
2023-06-11,2023-07-21
---
Trip Name:
Lisbon and Porto
Starting Date:
2023-06-14,2023-08-18
---
```

Pliki JSON zapisano dzięki MongoDB Compass. Znajdują się one w katalogu „Zadanie2 – JSON”.

Dodatkowo analogicznie jak w zadaniu pierwszym zapisałem wyniki zapytań do plików .txt, które znajdują się w katalogu „Zadanie2 – wyniki”.

W sprawozdaniu należy zamieścić przykładowe dokumenty w formacie JSON (pkt a) i b)), oraz kod zapytań/operacji (pkt c)), wraz z odpowiednim komentarzem opisującym strukturę dokumentów oraz polecenia ilustrujące wykonanie przykładowych operacji na danych

Do sprawozdania należy kompletny zrzut wykonanych/przygotowanych baz danych (taki zrzut można wykonać np. za pomocą poleceń mongoexport, mongdump …) oraz plik z kodem operacji zapytań (załącznik powinien mieć format zip).

Punktacja za zadanie (razem 2pkt)