

Dokumentowe bazy danych – MongoDB

ćwiczenie 2

Imiona i nazwiska autorów: Łukasz Kluza, Mateusz Sacha

Yelp Dataset

- www.yelp.com - serwis społecznościowy – informacje o miejscach/lokalach
- restauracje, kluby, hotele itd. **businesses**,
- użytkownicy odwiedzają te miejsca - "meldują się" **check-in**
- użytkownicy piszą recenzje **reviews** o miejscach/lokalach i wystawiają oceny
- przykładowy zbiór danych zawiera dane z 5 miast: Phoenix, Las Vegas, Madison, Waterloo i Edinburgh.

Zadanie 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/>

1. Zwróć dane wszystkich restauracji (kolekcja **business**, 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.
2. Ile każda firma otrzymała ocen/wskazówek (kolekcja **tip**) w 2012. Wynik powinien zawierać nazwę firmy oraz liczbę ocen/wskazówek. Wynik posortuj według liczby ocen (**tip**).
3. 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)
4. 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.
5. Wyznacz, jaką średnią ocenę uzyskała każda firma na podstawie wszystkich recenzji (kolekcja **review**, pole **stars**). Ogranicz do firm, które uzyskały średnią powyżej 3 gwiazdek.
 - a) Wynik powinien zawierać id firmy oraz średnią ocenę. Posortuj wynik wg id firmy.
 - b) Wynik powinien zawierać nazwę firmy oraz średnią ocenę. Posortuj wynik wg nazwy firmy.

Zadanie 1 - rozwiązanie

Wyniki:

przykłady, kod, zrzuty ekranów, komentarz ...

```
db.business.find({"categories": "Restaurants", "stars" : {$gte : 4},
"hours.Monday":{$exists : true}},
  {"_id": 0,"name" : 1, "full_address" : 1, "categories": 1, "hours": 1, "stars"
:1}).sort({"name": 1})
```

	categories	full_address	hours	name	stars
1	["Food", "Desserts", "Coffee & Tea", "Indian", "Restaurant"]	67 Nicolson Street#Me...	{"Monday": {"close": "22:00", "open": "10:00"}, "Tuesday": {"close": "22:00", "open": "11:00"}, "Friday": {"close": "22:00", "open": "11:00"}, "Saturday": {"close": "22:00", "open": "11:00"}, "Sunday": {"close": "22:00", "open": "11:00"}}	10-to-10 In Delhi	4.5
2	["Vietnamese", "Asian Fusion", "French", "Restaurants"]	4180 S Jones Blvd#Las...	{"Monday": {"close": "22:00", "open": "11:00"}, "Tuesday": {"close": "22:00", "open": "11:00"}, "Wednesday": {"close": "22:00", "open": "11:00"}, "Thursday": {"close": "22:00", "open": "11:00"}, "Friday": {"close": "22:00", "open": "11:00"}, "Saturday": {"close": "22:00", "open": "11:00"}, "Sunday": {"close": "22:00", "open": "11:00"}}	188 Restaurant	4
3	["Bars", "Asian Fusion", "Nightlife", "Lounges", "Karaoke"]	6145 W Sahara Ave#Uni...	{"Monday": {"close": "05:00", "open": "18:00"}, "Tuesday": {"close": "05:00", "open": "18:00"}, "Wednesday": {"close": "05:00", "open": "18:00"}, "Thursday": {"close": "05:00", "open": "18:00"}, "Friday": {"close": "05:00", "open": "18:00"}, "Saturday": {"close": "05:00", "open": "18:00"}, "Sunday": {"close": "05:00", "open": "18:00"}}	21 Restaurant & Lounge	4
4	["Food", "Live/Raw Food", "Juice Bars & Smoothies", "Vega"]	6140 W Chandler Blvd#...	{"Monday": {"close": "14:00", "open": "08:00"}, "Tuesday": {"close": "14:00", "open": "08:00"}, "Wednesday": {"close": "14:00", "open": "08:00"}, "Thursday": {"close": "14:00", "open": "08:00"}, "Friday": {"close": "14:00", "open": "08:00"}, "Saturday": {"close": "14:00", "open": "08:00"}, "Sunday": {"close": "14:00", "open": "08:00"}}	24 Carrots Juice Bar & Cafe	4
5	["Breakfast & Brunch", "Gluten-Free", "Vegetarian", "Rest"]	1701 E Guadalupe Rd#T...	{"Monday": {"close": "14:00", "open": "08:00"}, "Tuesday": {"close": "14:00", "open": "08:00"}, "Wednesday": {"close": "14:00", "open": "08:00"}, "Thursday": {"close": "14:00", "open": "08:00"}, "Friday": {"close": "14:00", "open": "08:00"}, "Saturday": {"close": "14:00", "open": "08:00"}, "Sunday": {"close": "14:00", "open": "08:00"}}	24 Carrots Juice Bar & Cafe	4
6	["Asian Fusion", "Restaurants"]	4632 S Maryland Pkwy#...	{"Monday": {"close": "22:00", "open": "11:00"}, "Tuesday": {"close": "22:00", "open": "11:00"}, "Wednesday": {"close": "22:00", "open": "11:00"}, "Thursday": {"close": "22:00", "open": "11:00"}, "Friday": {"close": "22:00", "open": "11:00"}, "Saturday": {"close": "22:00", "open": "11:00"}, "Sunday": {"close": "22:00", "open": "11:00"}}	2860 The Restaurant	4
7	["Coffee & Tea", "Food", "Breakfast & Brunch", "Bars", "W"]	10626 N 32nd St#Phoen...	{"Monday": {"close": "22:00", "open": "06:00"}, "Tuesday": {"close": "22:00", "open": "06:00"}, "Wednesday": {"close": "22:00", "open": "06:00"}, "Thursday": {"close": "22:00", "open": "06:00"}, "Friday": {"close": "22:00", "open": "06:00"}, "Saturday": {"close": "22:00", "open": "06:00"}, "Sunday": {"close": "22:00", "open": "06:00"}}	32 Shea	4.5
8	["Bakeries", "Food", "Breakfast & Brunch", "Coffee & Tea"]	305 N 4th St#Emerson ...	{"Monday": {"close": "14:30", "open": "07:00"}, "Tuesday": {"close": "14:30", "open": "07:00"}, "Wednesday": {"close": "14:30", "open": "07:00"}, "Thursday": {"close": "14:30", "open": "07:00"}, "Friday": {"close": "14:30", "open": "07:00"}, "Saturday": {"close": "14:30", "open": "07:00"}, "Sunday": {"close": "14:30", "open": "07:00"}}	4620 Bakery & Cafe	4.5
9	["Cafes", "American (Traditional)", "Diners", "Restaurant"]	4022 E Greenway Rd#St...	{"Monday": {"close": "14:00", "open": "05:00"}, "Tuesday": {"close": "14:00", "open": "05:00"}, "Wednesday": {"close": "14:00", "open": "05:00"}, "Thursday": {"close": "14:00", "open": "05:00"}, "Friday": {"close": "14:00", "open": "05:00"}, "Saturday": {"close": "14:00", "open": "05:00"}, "Sunday": {"close": "14:00", "open": "05:00"}}	40th Street Cafe	4
10	["American (New)", "Restaurants"]	108 King St#Capitol#M...	{"Monday": {"close": "14:00", "open": "11:30"}, "Tuesday": {"close": "14:00", "open": "11:30"}, "Wednesday": {"close": "14:00", "open": "11:30"}, "Thursday": {"close": "14:00", "open": "11:30"}, "Friday": {"close": "14:00", "open": "11:30"}, "Saturday": {"close": "14:00", "open": "11:30"}, "Sunday": {"close": "14:00", "open": "11:30"}}	43 North	4
11	["Vietnamese", "Restaurants"]	2844 N 43rd Ave#Phoen...	{"Monday": {"close": "20:00", "open": "09:00"}, "Tuesday": {"close": "20:00", "open": "09:00"}, "Wednesday": {"close": "20:00", "open": "09:00"}, "Thursday": {"close": "20:00", "open": "09:00"}, "Friday": {"close": "20:00", "open": "09:00"}, "Saturday": {"close": "20:00", "open": "09:00"}, "Sunday": {"close": "20:00", "open": "09:00"}}	43rd Express	4.5
12	["Bars", "Restaurants", "American (Traditional)", "Sports"]	5001 N Scottsdale Rd#...	{"Monday": {"close": "00:00", "open": "11:00"}, "Tuesday": {"close": "00:00", "open": "11:00"}, "Wednesday": {"close": "00:00", "open": "11:00"}, "Thursday": {"close": "00:00", "open": "11:00"}, "Friday": {"close": "00:00", "open": "11:00"}, "Saturday": {"close": "00:00", "open": "11:00"}, "Sunday": {"close": "00:00", "open": "11:00"}}	4th Floor Grille & Sports Bar	4
13	["Food", "Ethnic Food", "Thai", "Specialty Food", "Restau"]	2011 N Recker Rd#Mesa...	{"Monday": {"close": "21:00", "open": "11:00"}, "Tuesday": {"close": "21:00", "open": "11:00"}, "Wednesday": {"close": "21:00", "open": "11:00"}, "Thursday": {"close": "21:00", "open": "11:00"}, "Friday": {"close": "21:00", "open": "11:00"}, "Saturday": {"close": "21:00", "open": "11:00"}, "Sunday": {"close": "21:00", "open": "11:00"}}	5 R Cha Thai Bistro	4
14	["Food", "Ethnic Food", "Thai", "Specialty Food", "Restau"]	203 E 7th St#Tempe, AZ...	{"Monday": {"close": "21:00", "open": "11:00"}, "Tuesday": {"close": "21:00", "open": "11:00"}, "Wednesday": {"close": "21:00", "open": "11:00"}, "Thursday": {"close": "21:00", "open": "11:00"}, "Friday": {"close": "21:00", "open": "11:00"}, "Saturday": {"close": "21:00", "open": "11:00"}, "Sunday": {"close": "21:00", "open": "11:00"}}	5 R Cha Thai Diner	4
15	["Restaurants", "Fast Food", "Fast Food", "Burgers", "Burgers"]	155 N Country Club Dr...	{"Monday": {"close": "00:00", "open": "11:00"}, "Tuesday": {"close": "00:00", "open": "11:00"}, "Wednesday": {"close": "00:00", "open": "11:00"}, "Thursday": {"close": "00:00", "open": "11:00"}, "Friday": {"close": "00:00", "open": "11:00"}, "Saturday": {"close": "00:00", "open": "11:00"}, "Sunday": {"close": "00:00", "open": "11:00"}}	5 Star BBQ & Grill	4

```
db.tip.aggregate([
  {
    $match: {
      date: { $gte: "2012-01-01", $lt: "2013-01-01" }
    },
  },
  {
    $group: {
      _id: "$business_id",
      tip_count: { $sum: 1 }
    },
  },
  {
    $lookup: {
      from: "business",
      localField: "_id",
      foreignField: "business_id",
      as: "business_info"
    },
  },
  {
    $unwind: "$business_info"
  },
  {
    $project: {
      _id: 0,
      name: "$business_info.name",
      tip_count: 1
    },
  },
  {
    $sort: { 'tip_count': -1 }
  }
])
```

```
    }
  })
}
```

	{ name	{ tip_count
1	McCarran International Airport	1084
2	Phoenix Sky Harbor International Airport	622
3	Earl of Sandwich	430
4	Las Vegas Athletic Club Southwest	374
5	The Cosmopolitan of Las Vegas	351
6	Wicked Spoon	347
7	Sushi House Goyemon	258
8	Pho Kim Long	252
9	Secret Pizza	239

```
db.review.aggregate([
  {
    $unwind: "$votes"
  },
  {
    $group: {
      _id: null,
      funny_sum: { $sum: "$votes.funny" },
      funny_useful: { $sum: "$votes.useful" },
      funny_cool: { $sum: "$votes.cool" }
    }
  },
  {
    $project: {
      _id: 0
    }
  }
])
```

	{ funny_cool	{ funny_sum	{ funny_useful
1	735341	590956	1274331

```
db.user.aggregate([
  {
    $group: {
      _id: "$_id",
      funny_sum: { $sum: "$votes.funny" },
      useful_sum: { $sum: "$votes.useful" },
    }
  }
])
```

```

        cool_sum: { $sum: "$votes.cool" },
        name: { $first: "$name" }
    },
    {
        $match: { $or: [{ "funny_sum": 0 }, { "useful_sum": 0 }] }
    },
    {
        $sort: { "name": 1 }
    },
    {
        $project: {
            _id: 0,
            name: 1
        }
    }
]
])

```

	name
1	Bernard
2	'Anastacia
3	'Brandon
4	'David
5	'Jenelle
6	'Persian
7	'Virydiana
8	,Maria
9	.rick
10	006969123
11	A
12	A
13	A
14	A
15	A
16	A
17	A
18	A
19	A
20	A

```

db.business.aggregate([
  {
    $lookup: {
      from: "review",
      localField: "business_id",
      foreignField: "business_id",

```

```
        as: "reviews"
      }
    },
    {
      $group: {
        _id: 0,
        name: { $first: "$name" },
        average_stars: { $avg: "$reviews.stars" }
      }
    },
    {
      $sort: { "name": 1 }
    }
  ]
})
```

tutaj najprawdopodobnie mieliśmy jakiś mały problem ponieważ zapytanie z sortowaniem nie dawało rezultatu

Zadanie 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 wariant

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`, `mongodump` ...) oraz plik z kodem operacji zapytań (załącznik powinien mieć format zip).

Zadanie 2 - rozwiązanie

Wyniki:

a)

Wariant 1 (embedding):

Struktura danych:

Jedna kolekcja zawierająca wszystkie dane o firmach, wycieczkach, osobach i rezerwacjach osadzonych w jednym dokumencie.

```
Embedded documents:
{
  "_id": ObjectId,
  "company": {
    "name": String,
    "location": String
  },
  "tour": {
    "name": String,
    "date": Date,
    "duration": Number
  },
  "person": {
    "name": String,
    "email": String
  },
  "reservation": {
    "seats": Number,
    "price": Number
  },
  "rating": Number
}
```

Zalety:

- Prostota struktury danych - wszystkie powiązane dane są przechowywane w jednym dokumencie, co ułatwia ich pobieranie i aktualizację.
- Szybki dostęp do danych - nie ma potrzeby wykonywania złączeń między kolekcjami, co może poprawić wydajność zapytań.

Wady:

- Duplikacja danych - dane mogą być powielane w wielu dokumentach, co może prowadzić do redundancji i zwiększać rozmiar bazy danych.
 - Aktualizacja danych - jeśli dane są powielane w wielu miejscach, aktualizacja ich wymaga modyfikacji wielu dokumentów, co może być czasochłonne i skomplikowane.
-

Wariant 2 (references):

Struktura danych:

Oddzielne kolekcje dla firm, wycieczek, osób i rezerwacji, z referencjami między nimi.

```
Companies collection:
{
  "_id": ObjectId,
  "name": String,
  "location": String
}
```

```
Trips collection:
{
  "_id": ObjectId,
  "companyId": ObjectId,
  "name": String,
  "date": Date,
  "duration": Number
}
```

```
Persons collection:
{
  "_id": ObjectId,
  "name": String,
  "email": String
}
```

```
Reservations collection:
{
  "_id": ObjectId,
  "tourId": ObjectId,
  "personId": ObjectId,
  "seats": Number,
  "price": Number,
  "rating": Number
}
```

Zalety:

- Brak duplikacji danych - każda encja jest przechowywana w osobnym dokumencie, co eliminuje redundancję danych.
- Łatwa aktualizacja - jeśli dane są przechowywane w jednym dokumencie, aktualizacja wymaga zmiany tylko jednego miejsca.

Wady:

- Potrzeba wykonywania złączeń - w celu pobrania powiązanych danych konieczne jest wykonanie złączeń między kolekcjami, co może wpłynąć na wydajność zapytań.
- Większa złożoność struktury danych - konieczność zarządzania referencjami między kolekcjami może sprawić, że struktura danych będzie bardziej skomplikowana.

b)

Wariant 1 (przykładowe dane):

```
{
  "_id":{"$oid":"66322ba7fc33997b00b115ac"},
  "company":{"
    "name":"Adventure Tours Inc.",
    "location":"New York"
  },
  "tour":{"
    "name":"City Bike Tour",
    "date":{"$date":"2024-05-10T09:00:00Z"},
    "duration":3
  },
  "person":{"
    "name":"John Doe",
    "email":"john@example.com"
  },
  "reservation":{"
    "seats":2,
    "price":100
  },
  "rating":4
}
```


	company	person	rating	reservation
1	5ac {"name": "Adventure Tours Inc.", "location": "New York"}	{"name": "John Doe", "email": "john@example.com"}	4	{"seats": new Num
2	5ae {"name": "Travel Adventures Ltd.", "location": "Los Ang	{"name": "Emily Smith", "email": "emily@example.com"}	5	{"seats": new Num
3	5b0 {"name": "Nature Explorers LLC", "location": "Seattle"}	{"name": "Michael Johnson", "email": "michael@example.c	4	{"seats": new Num
4	5b2 {"name": "City Tours Inc.", "location": "Chicago"}	{"name": "Jessica Brown", "email": "jessica@example.com	3	{"seats": new Num
5	5b4 {"name": "Outdoor Adventures Ltd.", "location": "Denver	{"name": "Sophia Taylor", "email": "sophia@example.com"	4	{"seats": new Num
6	5b6 {"name": "Urban Explorers Inc.", "location": "San Franc	{"name": "Daniel Wilson", "email": "daniel@example.com"	5	{"seats": new Num
7	5b8 {"name": "Wilderness Discoveries LLC", "location": "Por	{"name": "Olivia Martinez", "email": "olivia@example.co	5	{"seats": new Num
8	5ba {"name": "Cultural Journeys Ltd.", "location": "Miami"}	{"name": "Ethan Johnson", "email": "ethan@example.com"}	4	{"seats": new Num
9	5bc {"name": "Adventure Seekers LLC", "location": "Austin"}	{"name": "Ava Brown", "email": "ava@example.com"}	4	{"seats": new Num
10	5be {"name": "Mountain Treks Inc.", "location": "Salt Lake	{"name": "Liam Wilson", "email": "liam@example.com"}	5	{"seats": new Num
11	5c0 {"name": "Nature Tours Ltd.", "location": "Vancouver"}	{"name": "Emma Thompson", "email": "emma@example.com"}	4	{"seats": new Num
12	5c1 {"name": "City Explorations Inc.", "location": "Toronto	{"name": "William Davis", "email": "william@example.com	5	{"seats": new Num
13	5c2 {"name": "Wildlife Adventures Ltd.", "location": "Calga	{"name": "Charlotte White", "email": "charlotte@example	5	{"seats": new Num
14	5c3 {"name": "Historical Tours Inc.", "location": "Boston"}	{"name": "James Brown", "email": "james@example.com"}	4	{"seats": new Num
15	5c4 {"name": "Adventures Unlimited LLC", "location": "Denve	{"name": "Sophie Johnson", "email": "sophie@example.com	4	{"seats": new Num
16	5c6 {"name": "Coastal Adventures Ltd.", "location": "Sydney	{"name": "Isabella Wilson", "email": "isabella@example.	5	{"seats": new Num
17	5c7 {"name": "Mountain Explorations Inc.", "location": "Den	{"name": "Ethan Thompson", "email": "ethan@example.com"	4	{"seats": new Num
18	5c8 {"name": "Scenic Tours Ltd.", "location": "Auckland"}	{"name": "Oliver Davis", "email": "oliver@example.com"}	5	{"seats": new Num
19	5c9 {"name": "Urban Adventures Ltd.", "location": "Chicago"}	{"name": "Sophia Brown", "email": "sophia@example.com"	4	{"seats": new Num

Wariant 2 (przykładowe dane):

```
Companies collection:
{
  "_id": ObjectId("60994c3d65c84c4bf432fc1a"),
  "name": "Adventure Tours Inc.",
  "location": "New York"
}
```

	_id	location	name
1	60994c3d65c84c4bf432fc1a	New York	Adventure Tours Inc.
2	663236e62b9d0a4a55d315eb	New York	Company 2
3	663236e62b9d0a4a55d315ef	New York	Company 3
4	663236e62b9d0a4a55d315f3	New York	Company 4
5	663236e62b9d0a4a55d315f7	New York	Company 5
6	663236e62b9d0a4a55d315fb	New York	Company 6
7	663236e62b9d0a4a55d315ff	New York	Company 7
8	663236e62b9d0a4a55d31603	New York	Company 8
9	663236e62b9d0a4a55d31607	New York	Company 9
10	663236e62b9d0a4a55d3160b	New York	Company 10
11	663236e62b9d0a4a55d3160f	New York	Company 11
12	663236e72b9d0a4a55d31613	New York	Company 12
13	663236e72b9d0a4a55d31617	New York	Company 13
14	663236e72b9d0a4a55d3161b	New York	Company 14
15	663236e72b9d0a4a55d3161f	New York	Company 15
16	663236e72b9d0a4a55d31623	New York	Company 16
17	663236e72b9d0a4a55d31627	New York	Company 17
18	663236e72b9d0a4a55d3162b	New York	Company 18
19	663236e72b9d0a4a55d3162f	New York	Company 19

```
Tours collection:
{
  "_id": ObjectId("60994c4f65c84c4bf432fc1b"),
  "companyId": ObjectId("60994c3d65c84c4bf432fc1a"),
  "name": "City Bike Tour",
  "date": ISODate("2024-05-10T09:00:00Z"),
  "duration": 3
}
```

	{_id	{ companyId	{ date	{ duration	{ name
1	60994c4f65c84c4bf432fc1b	60994c3d65c84c4bf432fc1a	2024-05-10T09:00:00.000Z	3	City Bike Tour
2	663236e62b9d0a4a55d315ec	663236e62b9d0a4a55d315eb	2024-05-08T12:34:46.812Z	3	Tour 2
3	663236e62b9d0a4a55d315f0	663236e62b9d0a4a55d315ef	2024-05-08T12:34:46.836Z	3	Tour 3
4	663236e62b9d0a4a55d315f4	663236e62b9d0a4a55d315f3	2024-05-08T12:34:46.857Z	3	Tour 4
5	663236e62b9d0a4a55d315f8	663236e62b9d0a4a55d315f7	2024-05-08T12:34:46.875Z	3	Tour 5
6	663236e62b9d0a4a55d315fc	663236e62b9d0a4a55d315fb	2024-05-08T12:34:46.895Z	3	Tour 6
7	663236e62b9d0a4a55d31600	663236e62b9d0a4a55d315ff	2024-05-08T12:34:46.914Z	3	Tour 7
8	663236e62b9d0a4a55d31604	663236e62b9d0a4a55d31603	2024-05-08T12:34:46.931Z	3	Tour 8
9	663236e62b9d0a4a55d31608	663236e62b9d0a4a55d31607	2024-05-08T12:34:46.948Z	3	Tour 9
10	663236e62b9d0a4a55d3160c	663236e62b9d0a4a55d3160b	2024-05-08T12:34:46.965Z	3	Tour 10
11	663236e62b9d0a4a55d31610	663236e62b9d0a4a55d3160f	2024-05-08T12:34:46.983Z	3	Tour 11
12	663236e72b9d0a4a55d31614	663236e72b9d0a4a55d31613	2024-05-08T12:34:47.003Z	3	Tour 12
13	663236e72b9d0a4a55d31618	663236e72b9d0a4a55d31617	2024-05-08T12:34:47.022Z	3	Tour 13
14	663236e72b9d0a4a55d3161c	663236e72b9d0a4a55d3161b	2024-05-08T12:34:47.039Z	3	Tour 14
15	663236e72b9d0a4a55d31620	663236e72b9d0a4a55d3161f	2024-05-08T12:34:47.056Z	3	Tour 15
16	663236e72b9d0a4a55d31624	663236e72b9d0a4a55d31623	2024-05-08T12:34:47.073Z	3	Tour 16
17	663236e72b9d0a4a55d31628	663236e72b9d0a4a55d31627	2024-05-08T12:34:47.094Z	3	Tour 17
18	663236e72b9d0a4a55d3162c	663236e72b9d0a4a55d3162b	2024-05-08T12:34:47.109Z	3	Tour 18
19	663236e72b9d0a4a55d31630	663236e72b9d0a4a55d3162f	2024-05-08T12:34:47.126Z	3	Tour 19

```
Persons collection:
{
  "_id": ObjectId("60994c7e65c84c4bf432fc1c"),
  "name": "John Doe",
  "email": "john@example.com"
}
```

1	60994c7e65c84c4bf432fc1c	john@example.com	John Doe
2	663236e62b9d0a4a55d315ed	person2@example.com	Person 2
3	663236e62b9d0a4a55d315f1	person3@example.com	Person 3
4	663236e62b9d0a4a55d315f5	person4@example.com	Person 4
5	663236e62b9d0a4a55d315f9	person5@example.com	Person 5
6	663236e62b9d0a4a55d315fd	person6@example.com	Person 6
7	663236e62b9d0a4a55d31601	person7@example.com	Person 7
8	663236e62b9d0a4a55d31605	person8@example.com	Person 8
9	663236e62b9d0a4a55d31609	person9@example.com	Person 9
10	663236e62b9d0a4a55d3160d	person10@example.com	Person 10
11	663236e62b9d0a4a55d31611	person11@example.com	Person 11
12	663236e72b9d0a4a55d31615	person12@example.com	Person 12
13	663236e72b9d0a4a55d31619	person13@example.com	Person 13
14	663236e72b9d0a4a55d3161d	person14@example.com	Person 14
15	663236e72b9d0a4a55d31621	person15@example.com	Person 15
16	663236e72b9d0a4a55d31625	person16@example.com	Person 16
17	663236e72b9d0a4a55d31629	person17@example.com	Person 17
18	663236e72b9d0a4a55d3162d	person18@example.com	Person 18
19	663236e72b9d0a4a55d31631	person19@example.com	Person 19

Reservations collection:

```
{
  "_id": ObjectId("60994c4f65c84c4bf432fc1b"),
  "tourId": ObjectId("60994c4f65c84c4bf432fc1b"),
  "personId": ObjectId("60994c7e65c84c4bf432fc1c"),
  "seats": 2,
  "price": 100,
  "rating": 4
}
```

	_id	personId	price	rating	seats	tourId
1	60994c4f65c84c4bf432fc1b	60994c7e65c84c4bf432fc1c	100	4	2	60994c4f65c84c4bf432fc1b
2	663236e62b9d0a4a55d315ee	663236e62b9d0a4a55d315ed	100	3	2	663236e62b9d0a4a55d315ec
3	663236e62b9d0a4a55d315f2	663236e62b9d0a4a55d315f1	100	4	2	663236e62b9d0a4a55d315f0
4	663236e62b9d0a4a55d315f6	663236e62b9d0a4a55d315f5	100	5	2	663236e62b9d0a4a55d315f4
5	663236e62b9d0a4a55d315fa	663236e62b9d0a4a55d315f9	100	4	2	663236e62b9d0a4a55d315f8
6	663236e62b9d0a4a55d315fe	663236e62b9d0a4a55d315fd	100	3	2	663236e62b9d0a4a55d315fc
7	663236e62b9d0a4a55d31602	663236e62b9d0a4a55d31601	100	4	2	663236e62b9d0a4a55d31600
8	663236e62b9d0a4a55d31606	663236e62b9d0a4a55d31605	100	3	2	663236e62b9d0a4a55d31604
9	663236e62b9d0a4a55d3160a	663236e62b9d0a4a55d31609	100	3	2	663236e62b9d0a4a55d31608
10	663236e62b9d0a4a55d3160e	663236e62b9d0a4a55d3160d	100	3	2	663236e62b9d0a4a55d3160c
11	663236e62b9d0a4a55d31612	663236e62b9d0a4a55d31611	100	5	2	663236e62b9d0a4a55d31610
12	663236e72b9d0a4a55d31616	663236e72b9d0a4a55d31615	100	4	2	663236e72b9d0a4a55d31614
13	663236e72b9d0a4a55d3161a	663236e72b9d0a4a55d31619	100	1	2	663236e72b9d0a4a55d31618
14	663236e72b9d0a4a55d3161e	663236e72b9d0a4a55d3161d	100	3	2	663236e72b9d0a4a55d3161c
15	663236e72b9d0a4a55d31622	663236e72b9d0a4a55d31621	100	2	2	663236e72b9d0a4a55d31620
16	663236e72b9d0a4a55d31626	663236e72b9d0a4a55d31625	100	4	2	663236e72b9d0a4a55d31624
17	663236e72b9d0a4a55d3162a	663236e72b9d0a4a55d31629	100	2	2	663236e72b9d0a4a55d31628
18	663236e72b9d0a4a55d3162e	663236e72b9d0a4a55d3162d	100	2	2	663236e72b9d0a4a55d3162c
19	663236e72b9d0a4a55d31632	663236e72b9d0a4a55d31631	100	1	2	663236e72b9d0a4a55d31630

- **c.1** Rozważmy zapytanie, które mam nam zwrócić wszystkie rezerwacje danej osoby które spełnią danę warunki:
 - muszą być zrealizowane przez daną firmę
 - ich cena powinna być większa od zadanej
 - czas ich trwania musi być równy zadanemu

Zrealizujemy teraz to zapytanie na dwóch bazach zaprojektowanych w różny sposób. W pierwszym przypadku szukamy wszystkich rezerwacji osoby o imieniu: *William Davis*, zrealizowanych w firmie: *City Explorations Inc.* z ceną powyżej 130 i trwających 4 dni.

Takie zapytanie możemy zrealizować tym poleceniem

```
db.tours.find(
{
  "company.name": "City Explorations Inc.",
  "person.name": "William Davis",
  "reservation.price": {$gt: 130},
  "tour.duration": {$eq: 4},
},
{
  "_id":0,
  "person.name": 1,
  "tour.name": 1,
  "tour.duration": 1,
  "company.name": 1,
  "reservation.seats" :1,
  "reservation.price" :1,
}
)
```

Jako rezultat dostaniemy:

	company	person	reservation	tour
1	{ "name": "City Explorations Inc." }	{ "name": "William Davis" }	{ "seats": new NumberInt("3"), "price": new NumberInt("180") }	{ "name": "Niagara Falls Tour", "duration": new NumberInt("4") }
2	{ "name": "City Explorations Inc." }	{ "name": "William Davis" }	{ "seats": new NumberInt("3"), "price": new NumberInt("150") }	{ "name": "Toronto Zoo Safari", "duration": new NumberInt("4") }
3	{ "name": "City Explorations Inc." }	{ "name": "William Davis" }	{ "seats": new NumberInt("3"), "price": new NumberInt("180") }	{ "name": "Niagara Falls Tour", "duration": new NumberInt("4") }
4	{ "name": "City Explorations Inc." }	{ "name": "William Davis" }	{ "seats": new NumberInt("3"), "price": new NumberInt("150") }	{ "name": "Toronto Zoo Safari", "duration": new NumberInt("4") }

W drugim przypadku tym razem szukamy wszystkich rezerwacji osoby o imieniu: *Sophia Taylor*, zrealizowanych w firmie: *Adventures Unlimited LLC* z ceną powyżej 75 i _trwających 5 dni.

Teraz polecenie realizujące to zapytanie wygląda następująco:

```
db.persons.aggregate([
  {
    $lookup: {
      from: "reservations",
      localField: "_id",
      foreignField: "personId",
      as: "reservation"
    }
  },
  {$unwind: "$reservation"},
  {
    $lookup: {
      from: "tours",
      localField: "reservation.tourId",
      foreignField: "_id",
      as: "tour"
    }
  },
  {$unwind: "$tour"},
  {
    $lookup: {
      from: "companies",
      localField: "tour.companyId",
      foreignField: "_id",
      as: "company"
    }
  },
  { $unwind: "$company" },
  { $match: { "name": "Sophia Taylor" } },
  { $match: { "company.name": "Adventures Unlimited LLC" } },
  { $match: { "reservation.price": { $gt: 75 } } },
  { $match: { "tour.duration": 5 } },
  {
    $project: { "_id": 0, "name": 1, "tour.name": 1, "tour.duration": 1,
      "company.name": 1, "reservation.seats": 1, "reservation.price": 1 }
  }
])
```

	company	name	reservation	tour
1	{"name": "Adventures Unlimited LLC"}	Sophia Taylor	{"seats": new NumberInt("2"), "price": new NumberInt("80")}	"r {"name": "Rock Climbing Adventure", "duration": new NumberInt("5")
2	{"name": "Adventures Unlimited LLC"}	Sophia Taylor	{"seats": new NumberInt("2"), "price": new NumberInt("90")}	"r {"name": "Rock Climbing Adventure", "duration": new NumberInt("5")

Wnioski

Mimo, że oba polecenie realizują dokładnie to samo zapytanie tylko z innymi parametrami ich kod jest zupełnie inny. W bazie danych typu *embedding* jest on dużo krótszy oraz łatwiejszy do napisania i zrozumienia. Dodatkowo sam czas wykonywania polecenia też jest szybszy ponieważ wykonujemy znacząco mniej operacji.

Natomiast w bazie danych typu *references* pomimo, że nie mamy redundancji danych to samo zapytanie jest dużo bardziej skomplikowane i bardziej czasochłonne ponieważ musimy używać polecenia *\$lookup* aby połączyć dwie kolekcje w jedną. W związku z tym baza typu *embedding* lepiej sprawdzi się niż *references* w sytuacji gdzie w przeciwnym wypadku musimy sięgać do wielu kolekcji jednocześnie.

- c.2 Tym razem porównujemy zapytania które zwracają nam unikalne imiona klientów.

embedding

```
db.tours.aggregate([
  {
    $group: {
      _id: "$person.name",
    }
  }
])
```

	{ _id
1	Sophia Taylor
2	Liam Wilson
3	Michael Johnson
4	Daniel Wilson
5	Charlotte White
6	Emily Smith
7	Emma Thompson
8	William Davis
9	Jessica Brown
10	Ava Brown
11	John Doe

references

```
db.persons.distinct("name")
```

	{ result
1	Ava Brown
2	Charlotte White
3	Daniel Wilson
4	Emily Smith
5	Emma Thompson
6	Jessica Brown
7	John Doe
8	Liam Wilson
9	Michael Johnson
10	Person 10
11	Person 11
12	Person 12
13	Person 13
14	Person 14
15	Person 15
16	Person 16
17	Person 17
18	Person 18
19	Person 19
20	Person 2

Wnioski

Do tego rodzaju zapytań lepiej sprawdzą się bazy typu *references* ponieważ w nich z definicji dla każdego rodzaju danych mamy osobne kolekcje. W bazie *embedding* to zapytanie też nie jest skomplikowane ale operacja *grupowania* może być bardziej czasochłonna niż *distinct*

- **c.3** Dodawanie nowych danych: fimry, wycieczki, klienta, rezerwacji:

embedding

```
db.tours.insertOne(  
  {  
    "company": {  
      "name": "Adventure Seekers Ltd.",  
      "location": "Denver"  
    },  
    "tour": {  
      "name": "Mountain Hiking Expedition",  

```

```

        "date": new Date("2025-07-15T08:00:00Z"),
        "duration": 5
    },
    "person": {
        "name": "David Johnson",
        "email": "david@example.com"
    },
    "reservation": {
        "seats": 2,
        "price": 120,
    },
    "rating": 4
},
)

```

```

▼ {
  ▼ "_id": {
    "$oid": "66375bf1bbce22392efc686a"
  },
  ▼ "company": {
    "name": "Adventure Seekers Ltd.",
    "location": "Denver"
  },
  ▼ "tour": {
    "name": "Mountain Hiking Expedition",
  ▼ "date": {
    "$date": "2025-07-15T08:00:00.000Z"
  },
    "duration": 5
  },
  ▼ "person": {
    "name": "David Johnson",
    "email": "david@example.com"
  },
  ▼ "reservation": {
    "seats": 2,
    "price": 120
  },
    "rating": 4
  }
}

```

references

```

db.companies.insertOne({
  "name": "Adventure Seekers Ltd.",
  "location": "Denver"
})

```



```
_id: ObjectId('66375ff0bbce22392efc686c')
name: "Adventure Seekers Ltd."
location: "Denver"
```

```
db.persons.insertOne({
  "name": "David Johnson",
  "email": "david@example.com"
})
```

```
_id: ObjectId('663760f1bbce22392efc686e')
name: "David Johnson"
email: "david@example.com"
```

```
db.tours.insertOne({
  "companyId" : ObjectId("66375ff0bbce22392efc686c"),
  "name": "Mountain Hiking Expedition",
  "date": new Date("2025-07-15T08:00:00Z"),
  "duration": 5
})
```

```
{
  "_id": {
    "$oid": "6637639fbbce22392efc6872"
  },
  "companyId": {
    "$oid": "66375ff0bbce22392efc686c"
  },
  "name": "Mountain Hiking Expedition",
  "date": {
    "$date": "2025-07-15T08:00:00.000Z"
  },
  "duration": 5
}
```

```
db.reservations.insertOne({
  "tourId": ObjectId("6637639fbbce22392efc6872"),
  "personId": ObjectId("663760f1bbce22392efc686e"),
  "seats": 2,
  "price": 120,
  "rating": 4
})
```

```
_id: ObjectId('6637659ebbce22392efc6874')
tourId: ObjectId('6637639fbbce22392efc6872')
personId: ObjectId('663760f1bbce22392efc686e')
seats: 2
price: 120
rating: 4
```

Wnioski

Dodawanie nowej rezerwacji, gdzie w bazie danych nie ma jeszcze danych klienta, firmy, wycieczki jest dużo łatwiejsze dla bazy danych *embedding*, natomiast minusem tego podejścia jest to, że dodając kolejne osoby do tej samej wycieczki będziemy musieli powiecać dane firmy oraz tej wycieczki. W bazie danych *references* natomiast musimy osobno dodać dane do każdej z kolekcji oraz zadbać o to aby klucze obce w relacjach były prawidłowe. Jednak dzięki temu unikamy duplikacji danych.

- **c.4** Usuwanie wcześniej dodanych danych

embedding

```
db.tours.deleteOne({
  _id: ObjectId("66375bf1bbce22392efc686a")
})
```

references

```
db.companies.deleteOne({
  _id: ObjectId("66375ff0bbce22392efc686c")
})
db.persons.deleteOne({
  _id: ObjectId("663760f1bbce22392efc686e")
})
db.tours.deleteOne({
  _id: ObjectId("6637639fbbce22392efc6872")
})
db.reservations.deleteOne({
  _id: ObjectId("6637659ebbce22392efc6874")
})
```

Wnioski

Usuwanie danych w oby tych bazach jest równie proste jednak w bazie typu *references* wymaga więcej operacji, które jednak nie są przesadnie trudne.

- **c.5** Zamiana lokalizacji siedzimy firmy *City Tours Inc.* na *New York*

embedding

```
db.tours.updateMany(
  {
    "company.name" : "City Tours Inc."
  },
  {
    $set: {"company.location" : "New York"}
  }
)
```

{_id	{ company	{ person	{ rating	{ reservation	{ tour
663626ecbdeb9822f761d4f6	{"name": "City Tours Inc.", "location": "New York"}	{"name": "John Doe", "email": "john.doe@example.com"}	4	{"seats": new NumberInt("1")}	{"name": "Lakefront Segway To
663626ecbdeb9822f761d4f7	{"name": "City Tours Inc.", "location": "New York"}	{"name": "John Doe", "email": "john.doe@example.com"}	5	{"seats": new NumberInt("1")}	{"name": "Historical Walking
663626ecbdeb9822f761d4f8	{"name": "City Tours Inc.", "location": "New York"}	{"name": "John Doe", "email": "john.doe@example.com"}	4	{"seats": new NumberInt("1")}	{"name": "Art Institute Tour"

references

```
db.companies.updateOne(
  {
    "name" : "City Tours Inc."
  },
  {
    $set: {"location" : "New York"}
  }
)
```

	{_id	{ location	{ name
1	6636170cbdeb9822f761d465	New York	City Tours Inc.

Wnioski

Mimo tego że oba polecenia są do siebie bardzo podobne to jednak tego typu aktualizację są wydaniejsze w bazie typu *references* ponieważ wystarczy zaktualizować lokalizację jedynie w jednym rekordzie w przeciwieństwie do bazy *embedding* gdzie jest potrzeba przeszukania każdej rezerwacji i zmian w wielu miejscach.

Zakładamy, że nazwy firm są unikalne

Podsumowanie

Przedstawiliśmy tutaj dwa zupełnie różne sposoby projektowania nierelacyjnych baz danych(*embedding* , *references*). Jednak ciężko jednoznacznie wskazać który sposób jest lepszy a który gorszy. Oba te podejście mają swoje wady i zalety a najważniejsze wydaje się być to aby dobierać model danych do konkretnego problemu. Dobrym pomysłem też mogłoby być połączenie ze sobą tych dwóch modeli tak aby połączyć ich największe zalety.

Punktacja:

zadanie	pkt
1	0,6
2	1,4
razem	2