Dokumentowe bazy danych – MongoDB

ćwiczenie 2

Imiona i nazwiska autorów: Stas Kochevenko & Wiktor Dybalski

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 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 średnia 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

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.

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

Wynik zapytania:

	{} categories	{} full_address	{} hours	{} name	{} stars
1	["Food", "Desserts", "Coffee & Tea",	67 Nicolson Street⏎Ne…	{"Monday": {"close": "22:00", "open": "10:00"}, "Tuesday": {"c	10-to-10 In Delhi	4.5
2	["Vietnamese", "Asian Fusion", "Frenc	4180 S Jones Blvd⏎Las…	{"Monday": {"close": "22:00", "open": "11:00"}, "Friday": {"cl	188 Restaurant	4
3	["Bars", "Asian Fusion", "Nightlife",	6145 W Sahara Ave⏎Uni…	{"Monday": {"close": "05:00", "open": "18:00"}, "Tuesday": {"c	21 Restaurant & Lounge	4
4	["Food", "Live/Raw Food", "Juice Bars	6140 W Chandler Blvd⏎…	{"Monday": {"close": "14:00", "open": "08:00"}, "Tuesday": {"c	24 Carrots Juice Bar & Cafe	4
5	["Breakfast & Brunch", "Gluten-Free",	1701 E Guadalupe Rd⏎T…	{"Monday": {"close": "14:00", "open": "08:00"}, "Tuesday": {"c	24 Carrots Juice Bar & Cafe	4
6	["Asian Fusion", "Restaurants"]	4632 S Maryland Pkwy⏎…	{"Monday": {"close": "22:00", "open": "11:00"}, "Tuesday": {"c	2860 The Restaurant	4
7	["Coffee & Tea", "Food", "Breakfast &	10626 N 32nd St⏎Phoen…	{"Monday": {"close": "22:00", "open": "06:00"}, "Tuesday": {"c	32 Shea	4.5
8	["Bakeries", "Food", "Breakfast & Bru	305 N 4th St⏎Emerson …	{"Monday": {"close": "14:30", "open": "07:00"}, "Tuesday": {"c	4&20 Bakery & Cafe	4.5
9	["Cafes", "American (Traditional)", "	4022 E Greenway Rd⏎St…	{"Monday": {"close": "14:00", "open": "05:00"}, "Tuesday": {"c	40th Street Cafe	4
10	["American (New)", "Restaurants"]	108 King St⏎Capitol⏎M…	{"Monday": {"close": "14:00", "open": "11:30"}, "Tuesday": {"c	43 North	4
11	["Vietnamese", "Restaurants"]	2844 N 43rd Ave⏎Phoen…	{"Monday": {"close": "20:00", "open": "09:00"}, "Friday": {"cl	43rd Express	4.5

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`).

```
db.tip.aggregate([
    {$match: {date: {$gte: "2012-01-01", $lte: "2012-12-31"}}},
    {$group: {_id: "$business_id", totalTips: {$sum: 1}}},
    {$lookup: {
        from: "business",
        localField: "_id",
        foreignField: "business_id",
        as: "business-info"
        }
    },
    {$unwind: "$business-info"},
    {$sort: {totalTips: -1}},
    {$project: {_id: 0, name: "$business_info.name", totalTips: 1}}
                ])
```

Wynik zapytania:

Nie udało nam się zarejestrować wyniku zapytania z powodu zbyt długiego czasu wykonywania podzapytania. Zatem użyliśmy prostszego zapytania bez kosztownego łączenia kolekcji:

```
db.tip.aggregate([
    {$match: {date: {$gte: "2012-01-01", $lte: "2012-12-31"}}},
    {$group: {_id: "$business_id", totalTips: {$sum: 1}}},
    {$sort: {totalTips: -1}}])
```

Wynik zapytania:

Output | Result 26 | Result 26-2

1-20 of 21+

	{} _id	{} totalTips
1	jf67Z1pnwElRSXllpQHiJg	1084
2	hW0Ne_HTHEAgGF1rAdmR-g	622
3	2e2e7WgqU1BnpxmQL5jbfw	430
4	CsNOg-u_wCuXSt9Z-xU92Q	374
5	AtjsjFzalWqJ7S9DUFQ4bw	351
6	zt1TpTuJ6y9n551sw9TaEg	347
7	AeKTQBtPRDHLAFL9bzbUnA	258
8	FV16IeXJp2W6pnghTz2FAw	252
9	eq6lQI039SBLC6sHm3idGA	239
10	Ht8mXLuqJSTPrU9kvzosUA	227
11	DjOxXobyGDwWt89q4z1twg	221
12	UZbkczu-KJour2Hgt_5WWw	209
13	JpHE7yhMS5ehA9e8WG_ETg	208
14	8buIr1zBCO7OEcAQSZko7w	201
15	Xhg93cMdemu5pAMkDoEdtQ	197
16	vxxMqBaAHuWdx4impsLSSA	197
17	4bEjOyTaDG24SY5TxsaUNQ	196
18	H_SuH7uLiYahDMbNBB9kog	185
19	z3SyT8blMIhsZNvKJgKcRA	184
20	eWPFXL1Bmu1ImtIa2Rqliw	184

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).

```
db.review1.aggregate([
    {$group: {
    _id: "Number_of_categories",
    totalFunny: {$sum: {
        $cond: {if: {$gt: ["$votes.funny", 0]},
                        then: 1, else: 0}}},
    totalUseful: {$sum: {
        $cond: {if: {$gt: ["$votes.useful", 0]},
                        then: 1, else: 0}}},
    totalCool: {$sum: {
        $cond: {if: {$gt: ["$votes.cool", 0]},
                        then: 1, else: 0}}}
    }}
])
```

Wynik zapytania:

	{} _id	{} totalCool	{} totalFunny	{} totalUseful
1	Number_of_categories	90800	73048	145477

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.

```
db.user.aggregate([
    {
        $group: {
            _id: "$name",
            totalFunny: {$sum: {$cond: {if: {$gt: ["$votes.funny", 0]}, then: "$votes.funny", else: 0}}},
            totalUseful: {$sum: {$cond: {if: {$gt: ["$votes.useful", 0]}, then: "$votes.useful", else: 0}}}
        }
    },
    {
        $match: {
            totalFunny: 0,
            totalUseful: 0
        }
    },
    {
        $project: {
            _id: 0,
            name: "$_id",
            totalFunny: 1,
            totalUseful: 1
        }
    },
    {
        $sort: {name: 1}
    }
])
```

Wynik zapytania:

1-20 of 21+

	{} name	{} totalFunny	{} totalUseful
1	Bernard	0	0
2	,Maria	0	0
3	006969123	0	0
4	A C	0	0
5	A Castro	0	0
6	A Renee	0	0
7	A Thomas	0	0
8	A. D.	0	0
9	A.P.	0	0
10	AA Printing	0	0
11	AAA HOUSECLEANERS	0	0
12	AZFoodie	0	0
13	Aartje	0	0
14	Abbay	0	0
15	Abdallah	0	0
16	Abdallaziz	0	0
17	Abdel	0	0
18	AbdulRahman	0	0
19	Abee	0	0
20	Abegael	0	0

5. Wyznacz, jaką średnia 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.

```
db.review1.aggregate([
    {$group: {
        _id: "$business_id",
        avg_stars: {$avg: "$stars"}
    }},
    {$match: {
        "avg_stars": {$gt: 3}
    }},
    {$sort: {
        _id: 1
    }}
])
```

Wynika zapytania A:

1-20 of 21+

	{} _id	{} avg_stars
1	--XBxRlD92RaV6TyUnP8Ow	3.6666666666666665
2	-0HGqwlfw3I8nkJyMHxAsQ	4
3	-0QBrNvhrPQCaeo7mTo0zQ	4.333333333333333
4	-1bOb2izeJBZjHC7NWxiPA	3.824324324324324
5	-34jE_5dujSWMIOBudQsiQ	4.75
6	-3JXOT-i2gDbASLgDeOSwQ	4.75
7	-3WVw1TNQbPBzaKCaQQ1AQ	3.7892156862745097
8	-3qWPkQAxsggQJYqgi5myA	3.25
9	-3xbryp44xhpN4BohxXDdQ	3.8151260504201683
10	-4BtfPW3_v092G8pqHrv4w	3.625
11	-4zMr3jk0ykmsk5sxsQMlA	4
12	-5IgAihccTd8_iENVgpd9A	3.5
13	-5bLhMLNGbIVyoPOVl0yLg	3.3333333333333335
14	-63VfA2tnYyzwRt81B1AKw	4.0588235294117645
15	-65rmLRQ5JDwI-l8UDEV7Q	3.8
16	-6Ft3hulif7O2sIdKiG0Og	4.5
17	-6O53B-ksqSKzWM6Y9moEQ	3.25
18	-6Roo-EHgSdUa4rP3tWyRw	3.4571428571428573
19	-6c7Tfec-X0aDmzjKO6R-Q	3.1666666666666665
20	-7DxAxnXkAlTXUrtByoXAg	4.666666666666667

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

b) Wynik powinien zawierać nazwę firmy oraz średnią ocenę. Posortuj wynik wg nazwy firmy.

```
db.review1.aggregate([
    {$group: {
        _id: "$business_id",
        avg_stars: {$avg: "$stars"}
        }
    },
    {$lookup: {
```

```
        from: "business",
        localField: "_id",
        foreignField: "business_id",
        as: "business-info"
        }
    },
    {$unwind: "$business-info"},
    {$project: {
            _id: 0,
            business_name: "$business-info.name",
            avg_stars: 1
        }
    },
    {$sort:
        {business_name: 1}
        }
])
```

Wynik zapytania B:

Output | Result 18 | 1-100 of 101+

	business_name	avg_stars
1	#1 Brothers Pizza	4.125
2	1 Hr Photo Shack	4.846153846153846
3	19th Donut Hole	4.055555555555555
4	1st Emergency Pet Care	4.375
5	1st Nails II	4.928571428571429
6	1st Pet Veterinary Centers	3.9444444444444446
7	2 Men and A Pizza	3.0714285714285716
8	2007 Nails	3.5
9	24 Hour Fitness	3.3026315789473686
10	24 Hour Fitness	3.6451612903225805
11	24 Hour Fitness	3.4210526315789473
12	24 Hour Fitness	3.5454545454545454
13	24 Hour Fitness	4.1
14	24 Hour Fitness	4.5
15	24 Hour Fitness	3.8095238095238093
16	24 Hour Fitness	3.5714285714285716
17	24 Hour Fitness	3.2666666666666666
18	24 Hour Fitness	3.3157894736842106
19	24 Hour Fitness	3.32
20	24 Hours Laundromat	4.0625
21	25th Street Automotive	4.71875
22	360 Physical Therapy	3.6666666666666665
23	3Nuts Inc	4.375
24	4 Kings North	4
25	4 Wheelers Supply & Off Road Centers	4.75

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

Zadanie 2 - rozwiązanie

Podejście tabelaryczne

a) Analiza danego wariantu

Podejście tabelaryczne jest znane z modelu relacyjnego: uporządkujemy dane w taki sposób, że każda tabela/kolekcja reprezentuje pewien rodzaj encji, np. Klienci, Nauczyciele, Przedmioty. Encje są powiązane między sobą przez referencje.

- Zalety
  - Bardziej przejrzysty model danych: dane są pogrupowane w taki sposób, że nie ma potrzeby nadużywania dokumentów zagnieżdżonych
  - Brak redundancji danych (oprócz sytuacji, gdzie "rejestrujemy" referencje w obu encjach relacji (w MongoDB), np. Product ma informacje o swoim Dostawce, a Dostawca ma informacje o swoich produktach)
  - Szybka modyfikacji danych: dzięki temu, że nie ma redundancji, potrzebujemy modyfikować dane tylko w jednym miejscu
  - Zapewniona spójność danych: modyfikacja danych tylko w jednym miejscu zapewnia, że dane będą spójne
  - Większa wydajność w sytuacji, gdy potrzebujemy danych dotyczących konkretnego małego dokumentu, a nie całego dokumentu wraz z zagnieżdżonymi dokumentami
- Wady
  - Konieczność używania operacji łączenia dokumentów różnych kolekcji w sytuacji, gdy potrzebujemy dokument wraz z dokumentami, które znajdują się z nim w relacji
  - Konieczność użycia złożonych zapytań (w MongoDB) do łączenia wielu dokumentów z różnych kolekcji
  - W przypadku wielokrotnego odczytywania danych z powiązanych między sobą dokumentów różnych kolekcji zmniejsza się wydajność, ponieważ za każdym razem potrzebujemy ponownie łączyć dane

b) Utworzenie kolekcji i wypełnienie kolekcji przykładowymi danymi

Przykładowa struktura bazy danych wygląda następująco:

```
Companies
{
    "_id": Number,
    "organized_trips_id": [Number],
    "company_name": String
}

Customers
{
    "_id": Number,
    "owned_tickets_id": [Number],
    "customer_review_id": [Number],
    "first_name": String,
    "last_name": String,
}

Trips
{
    "_id": Number,
    "company_id": [Number],
    "tickets_id": [Number],
    "date": Date,
    "location": String,
    "trip_review_id": [Number],
    "took_place": Boolean
}

Tickets
{
    "_id": Number,
    "trip_id": [Number],
    "reserved_seat_no": Number,
    "ticket_status": String,
    "first_name": String,
    "last_name": String
}

Reviews
{
    "_id": Number,
    "customer_id": Number,
    "trip_id": Number,
    "company_id": Number,
    "review_date": Date,
    "stars": Number,
    "review_description": String
}
```

Stworzenie bazy danych według wyżej przedstawionego pomysłu:

```
use tab_trip_database

db.createCollection("Companies")
db.createCollection("Customers")
db.createCollection("Trips")
db.createCollection("Tickets")
db.createCollection("Reviews")
```

Stworzenie rożnego rodzaju danych dla przykładowej bazy danych:

Kilka przydatnych informacji:

- took_place ustawione na True w kolekcji Trips oznacza, że wycieczka się już odbyła,
- review_id oraz ticket_id to liczby 3-cyfrowe zaczynające się od trip_id. 3 bilety dla np. trip_id=1 to: 101,102,103,
- seats_no to liczby odpowiadające numerom ticketów: np dla ticketa 102, seats_no to 2 itd
- ticket_status to jeden symbol spośród 'N', 'P' i 'C', oznaczających kolejno "New", "Paid" i "Canceled"

```
db.Companies.insertMany([
  { "_id": 1, "organized_trips_id": [1, 2], "company_name": "Adventure Works" },
  { "_id": 2, "organized_trips_id": [3, 4], "company_name": "Travel Corp" },
  { "_id": 3, "organized_trips_id": [5], "company_name": "Holiday Makers" },
  { "_id": 4, "organized_trips_id": [6, 7], "company_name": "Globe Trotters" },
  { "_id": 5, "organized_trips_id": [8], "company_name": "Pathfinders" }
])

db.Customers.insertMany([
  { "_id": 1, "owned_tickets_id": [101, 504], "customer_review_id": [101], "first_name": "John", "last_name":
"Doe" },
  { "_id": 2, "owned_tickets_id": [201], "customer_review_id": [201], "first_name": "Anna", "last_name": "Smith"
},
  { "_id": 3, "owned_tickets_id": [102, 301], "customer_review_id": [102, 301], "first_name": "David",
"last_name": "Brown" },
  { "_id": 4, "owned_tickets_id": [501, 502], "customer_review_id": [], "first_name": "Lisa", "last_name":
"White" },
  { "_id": 5, "owned_tickets_id": [503], "customer_review_id": [], "first_name": "Mark", "last_name": "Taylor" }
])

db.Trips.insertMany([
  { "_id": 1, "company_id": 1, "tickets_id": [101, 102], "date": {"$date": "2024-04-15T00:00:00Z"}, "location":
"Warsaw", "trip_review_id": [101, 102], "took_place": true },
  { "_id": 2, "company_id": 1, "tickets_id": [201], "date": {"$date": "2024-05-20T00:00:00Z"}, "location":
"Krakow", "trip_review_id": [201], "took_place": true },
  { "_id": 3, "company_id": 2, "tickets_id": [301], "date": {"$date": "2024-06-10T00:00:00Z"}, "location":
"Gdansk", "trip_review_id": [301], "took_place": true },
  { "_id": 4, "company_id": 2, "tickets_id": [], "date": {"$date": "2024-07-15T00:00:00Z"}, "location": "Poznan",
"trip_review_id": [], "took_place": false },
  { "_id": 5, "company_id": 3, "tickets_id": [501, 502, 503, 504], "date": {"$date": "2024-08-25T00:00:00Z"},
"location": "Wroclaw", "trip_review_id": [], "took_place": false },
  { "_id": 6, "company_id": 4, "tickets_id": [], "date": {"$date": "2024-09-14T00:00:00Z"}, "location": "Sopot",
"trip_review_id": [], "took_place": false },
  { "_id": 7, "company_id": 4, "tickets_id": [], "date": {"$date": "2024-09-15T00:00:00Z"}, "location": "Gdynia",
"trip_review_id": [], "took_place": false },
  { "_id": 8, "company_id": 5, "tickets_id": [], "date": {"$date": "2024-09-29T00:00:00Z"}, "location":
"Katowice", "trip_review_id": [], "took_place": false }
])


db.Tickets.insertMany([
  { "_id": 501, "trip_id": 5, "reserved_seat_no": 1, "ticket_status": 'P', "first_name": "Lisa", "last_name":
"White" },
  { "_id": 101, "trip_id": 1, "reserved_seat_no": 1, "ticket_status": 'P', "first_name": "John", "last_name":
"Doe" },
  { "_id": 102, "trip_id": 1, "reserved_seat_no": 2, "ticket_status": 'P', "first_name": "David", "last_name":
"Brown" },
  { "_id": 201, "trip_id": 2, "reserved_seat_no": 1, "ticket_status": 'P', "first_name": "Anna", "last_name":
"Smith" },
  { "_id": 502, "trip_id": 5, "reserved_seat_no": 2, "ticket_status": 'N', "first_name": "Lisa", "last_name":
"White" },
  { "_id": 503, "trip_id": 5, "reserved_seat_no": 1, "ticket_status": 'C', "first_name": "Mark", "last_name":
"Taylor" },
  { "_id": 301, "trip_id": 3, "reserved_seat_no": 1, "ticket_status": 'N', "first_name": "David", "last_name":
"Brown" },
  { "_id": 504, "trip_id": 5, "reserved_seat_no": 4, "ticket_status": 'C', "first_name": "John", "last_name":
"Doe" }
])

db.Reviews.insertMany([
  { "_id": 101, "customer_id": 1, "trip_id": 1, "company_id": 1, "review_date": {"$date": "2023-03-
```

```
01T00:00:00Z"}, "stars": 5, "review_description": "Excellent experience!" },
  { "_id": 102, "customer_id": 3, "trip_id": 1, "company_id": 1, "review_date": {"$date": "2023-03-
02T00:00:00Z"}, "stars": 4, "review_description": "Very good trip, well organized." },
  { "_id": 201, "customer_id": 2, "trip_id": 2, "company_id": 1, "review_date": {"$date": "2023-04-
01T00:00:00Z"}, "stars": 3, "review_description": "Decent, but could be better." },
  { "_id": 301, "customer_id": 3, "trip_id": 3, "company_id": 2, "review_date": {"$date": "2023-04-
26T00:00:00Z"}, "stars": 5, "review_description": "It was super!" },
])
```

Kolekcje po wypełnieniu danymi:

- Companies

	_id	company_name	organized_trips_id
1	1	Adventure Works	[new NumberInt("1"), new NumberInt("2")]
2	2	Travel Corp	[new NumberInt("3"), new NumberInt("4")]
3	3	Holiday Makers	[new NumberInt("5")]
4	4	Globe Trotters	[new NumberInt("6"), new NumberInt("7")]
5	5	Pathfinders	[new NumberInt("8")]

- Customers

	_id	customer_review_id	first_name	last_name	owned_tickets_id
1	1	[new NumberInt("101")]	John	Doe	[new NumberInt("101"), new NumberInt("504")]
2	2	[new NumberInt("201")]	Anna	Smith	[new NumberInt("201")]
3	3	[new NumberInt("102"), new NumberInt("301")]	David	Brown	[new NumberInt("102"), new NumberInt("301")]
4	4	[]	Lisa	White	[new NumberInt("501"), new NumberInt("502")]
5	5	[]	Mark	Taylor	[new NumberInt("503")]

- Reviews

	_id	company_id	customer_id	review_date	review_description	stars	trip_id
1	101	1	1	{"$date": "2023-03-01T00:00:00Z"}	Excellent experience!	5	1
2	102	1	3	{"$date": "2023-03-02T00:00:00Z"}	Very good trip, well organized.	4	1
3	201	1	2	{"$date": "2023-04-01T00:00:00Z"}	Decent, but could be better.	3	2
4	301	2	3	{"$date": "2023-04-26T00:00:00Z"}	It was super!	5	3

- Tickets

	_id	first_name	last_name	reserved_seat_no	ticket_status	trip_id
1	501	Lisa	White	1	P	5
2	101	John	Doe	1	P	1
3	102	David	Brown	2	P	1
4	201	Anna	Smith	1	P	2
5	502	Lisa	White	2	N	5
6	503	Mark	Taylor	1	C	5
7	301	David	Brown	1	N	3
8	504	John	Doe	4	C	5

- Trips

	_id	company_id	date	location	tickets_id	took_place	trip_review_id
1	1	1	{"$date": "2024-04-	Warsaw	[new NumberInt("101"), new NumberInt("102	true	[new NumberInt("100"), new Numbe
2	2	1	{"$date": "2024-05-	Krakow	[new NumberInt("201")]	true	[new NumberInt("201")]
3	3	2	{"$date": "2024-06-	Gdansk	[new NumberInt("301")]	true	[new NumberInt("301")]
4	4	2	{"$date": "2024-07-	Poznan	[]	false	[]
5	5	3	{"$date": "2024-08-	Wroclaw	[new NumberInt("501"), new NumberInt("502	false	[]
6	6	4	{"$date": "2024-09-	Sopot	[]	false	[]
7	7	4	{"$date": "2024-09-	Gdynia	[]	false	[]
8	8	5	{"$date": "2024-09-	Katowice	[]	false	[]

c) Analiza wad/zalet danego podejścia na konkretnych przykładach

- Wyświetlenie wszystkich ocen z `reviews` wraz z `company_name` dla każdej wycieczki `trip`, która już się odbyła

Przez to, że mamy umieszczone te dane w kilku różnych kolekcjach, zapytanie będzie składać się z kilku operatorów, a czas wykonania będzie nie najlepszy, ponieważ potrzebujemy łączyć kilka tabel. W drugim podejściu będzie pokazane alternatywne wykonanie tego zypatania.

```
db.Trips.aggregate(
{$match: {took_place: true}},

{$lookup: {from: "Companies",
           localField: "company_id",
           foreignField: "_id",
           as: "Company"}},

{$unwind: "$Company"},

{$lookup: {from: "Reviews",
           localField: "trip_review_id",
           foreignField: "_id",
           as: "Review"}},

{$project: {"company_name" : "$Company.company_name", location: 1, "marks" : "$Review.stars"}}
)
```

Wynik danego zypytania:

	{} _id	{} company_name	{} location	{} marks
1	1	Adventure Works	Warsaw	[new NumberInt("5"), new NumberInt("4")]
2	2	Adventure Works	Krakow	[new NumberInt("3")]
3	3	Travel Corp	Gdansk	[new NumberInt("5")]

- Wyświetlenie wszystkich biletów `tickets`, które mają status "Paid" albo "New"

W tym przypadku w łatwy sposób można skorzystać z kolekcji Tickets, która zawiera interesujące nas informacje (wystarczy dodać prosty warunek). Natomiast w strukturze z zagnieżdżonymi dokumentami mielibyśmy odwoływaś się do nich wewnątrz innych kolekcji (będzie pokazane niżej).

```
db.Tickets.find({$or : [{ticket_status: 'P'}, {ticket_status: 'N'}]})
```

Wynik danego zypytania:

	{} _id	{} first_name	{} last_name	{} reserved_seat_no	{} ticket_status	{} trip_id
1	101	John	Doe	1	P	1
2	102	David	Brown	2	P	1
3	201	Anna	Smith	1	P	2
4	301	David	Brown	1	N	3
5	501	Lisa	White	1	P	5
6	502	Lisa	White	2	N	5

- Dodanie nowego `review`

W podejściu tabelarycznym potrzebujemy tylko 1 raz wstawić dokument, zawierający informacje o danym review, dodając jego ID do list `reviews` w kolekcjach `trips` oraz `customers`. Dzięki braku redundacji takich złożonych struktur danych, wstawione dane będą zajmowały mniej miejsca, niż w przypadku podejścia dokumentowego, ponieważ w tym podejściu będziemy potrzebowali zrobić kilka razy więcej insertów i odpowiednio wykorzystać kilka razy więcej pamięci do przechowywania redundantych danych, natomiast będą one wygodniejsze w wykorzystaniu.

Przed wstawieniem ustawiliśmy status wydarzenia na odbyty.

```
db.Trips.updateOne(
    { _id: 5 }, { $set: { took_place: true } }
)
```

```
db.Reviews.insertOne({ "_id": 501, "customer_id": 1, "trip_id": 5, "company_id": 3, "review_date": {"$date":
"2024-05-03T00:00:00Z"}, "stars": 4.5, "review_description": "Great trip but it seems to be too expensive" },)
db.Customers.updateOne({"_id" : 1}, {$push: {customer_review_id: 501}})
db.Trips.updateOne({"_id" : 5}, {$push: {trip_review_id: 501}})
```

Wynik danego zypytania:

Reviews:

	_id	company_id	customer_id	review_date	review_description	stars	trip_id
1	101	1	1	{"$date": "2023-03-01T00:00:00Z"}	Excellent experience!	5	1
2	102	1	3	{"$date": "2023-03-02T00:00:00Z"}	Very good trip, well organized.	4	1
3	201	1	2	{"$date": "2023-04-01T00:00:00Z"}	Decent, but could be better.	3	2
4	301	2	3	{"$date": "2023-04-26T00:00:00Z"}	It was super!	5	3
5	501	3	1	{"$date": "2024-05-03T00:00:00Z"}	Great trip but it seems to be too expensive	4.5	5

Customers:

	_id	customer_review_id	first_name	last_name	owned_tickets_id
1	1	[new NumberInt("101"), new NumberInt("501")]	John	Doe	[new NumberInt("101"), new NumberInt("504")]
2	2	[new NumberInt("201")]	Anna	Smith	[new NumberInt("201")]
3	3	[new NumberInt("102"), new NumberInt("301")]	David	Brown	[new NumberInt("102"), new NumberInt("301")]
4	4	[]	Lisa	White	[new NumberInt("501"), new NumberInt("502")]
5	5	[]	Mark	Taylor	[new NumberInt("503")]

Trips:

	_id	company_id	date	location	ticke...	took_place	trip_review_id
1	1	1	{"$date": "2024-04-15T00:00:00Z"}	Warsaw	[new NumberIn	true	[new NumberInt("101"), new NumberInt("102")]
2	2	1	{"$date": "2024-05-20T00:00:00Z"}	Krakow	[new NumberIn	true	[new NumberInt("201")]
3	3	2	{"$date": "2024-06-10T00:00:00Z"}	Gdansk	[new NumberIn	true	[new NumberInt("301")]
4	4	2	{"$date": "2024-07-15T00:00:00Z"}	Poznan	[]	false	[]
5	5	3	{"$date": "2024-08-25T00:00:00Z"}	Wroclaw	[new NumberIn	true	[new NumberInt("501")]
6	6	4	{"$date": "2024-09-14T00:00:00Z"}	Sopot	[]	false	[]
7	7	4	{"$date": "2024-09-15T00:00:00Z"}	Gdynia	[]	false	[]
8	8	5	{"$date": "2024-09-29T00:00:00Z"}	Katowice	[]	false	[]

- Wyświetlenie średniej oceny dla każdej firmy `company_name` na podstawie wszystkich ocen z `reviews` dla każdej wycieczki `trip`, która już się odbyła

Skoro dane są podzielone na kilka różnych kolekcji, zapytanie będzie dość skomplikowane i nie wydajne. Nie możemy uniknąć łączenia kilku tabel, co ma istotny wpływ na czas wykonania polecenia i jego rozmiar. W drugim podejściu będzie pokazane alternatywne wykonanie tego zypatania.

```
db.Trips.aggregate([
  {$match: {took_place: true}},
  {$lookup: {
    from: "Companies",
    localField: "company_id",
    foreignField: "_id",
    as: "Company"
  }},
  {$unwind: "$Company"},
  {$lookup: {
    from: "Reviews",
    localField: "trip_review_id",
    foreignField: "_id",
    as: "Review"
  }},
  {$unwind: "$Review"},
  {$project: {
    "company_id": "$Company._id",
    "company_name": "$Company.company_name",
    "marks": "$Review.stars"
  }},
  {$group: {
    _id: "$company_name",
    average_rating: {$avg: "$marks"}
  }}
])
```

Wynik danego zypytania:

	{} _id	{} average_rating
1	Travel Corp	5
2	Holiday Makers	4.5
3	Adventure Works	4

Podejście dokumentowe

a) Analiza danego wariantu

W takim podejściu uporządkujemy dane w taki sposób, że każda kolekcja przedstawia pewien rodzaj encji, ale nie potrzebujemy przedstawiać wszystkich encji w osobnych kolekcjach. Encje mogą być powiązane między sobą przez referencje (w przypadku kilku dokumentów w różnych kolekcjach) albo za pomocą dokumentów zagnieżdżonych.

- Zalety
  - Brak konieczności używania kosztownych operacji do łączenia kilku dokumentów różnych kolekcji
  - Dobre podejście w przypadku rzadkiego modyfikowania danych, ponieważ mamy redundację danych
  - Lepsza wydajność w przypadku, gdy potrzebujemy informacji z całego dokumentu wraz z zagnieżdżonymi dokumentami
  - Utrzymywanie powiązynch elementów w jednym dokumencie poprzez użycie dokumentów zagnieżdżonych
- Wady
  - Redundacja danych
  - Utrudnienia modyfikacji: w konsekwencji redundacji danych potrzebujemy modyfikować te same dane wielokrotnie w różnych miejscah
  - Konieczność użycia bardziej złożonych zapytań w przypadku, gdy potrzebujemy dostać się do informacji zawartych w dokumentach zagnieżdżonych
  - W szczególnych przypadkach trudny do zrozumienia model danych (np. w przypadku wielokrotnego zagnieżdżenia dokumentów w zagnieżdżnych dokumentach)
  - Brak zagwarantowanej spójności danych

W takim podejściu przekształciliśmy poprzednią strukturę tabelaryczną, używając dokumentów zagnieżdżonych. Z 5 kolekcji (Customers, Trips, Reviews, Companies, Tickets) zostało tylko 2 (Customers i Trips). Informacje z kolekcji "Companies" zostały przeniesione do "Trips", informacje o ticketach i reviews każdego z kilentów zostały przeniesione do dwóch tablic dokumentów zagnieżdżonych w kolekcji "Customers". Wynika to z tego, że te dane prawie nigdy nie są zmieniane (chyba że status Ticketu, ale to nawet łatwiej zmieniać w dokumencie danego klienta). W taki sposób zmniejszyła się ilość danych, natomias trochę utrudnił się cały schemat.

b) Utworzenie kolekcji i wypełnienie kolekcji przykładowymi danymi

Przykładowa struktura bazy danych wygląda następująco:

```
Customers
{
    "_id": Number,
    "owned_tickets": [
            {
                "_id": Number,
                "reserved_seat_no": Number,
                "ticket_status": String
            }
    ],
    "customer_reviews": [
            {
                "_id": Number,
                "trip_id": Number,
                "review_date": Date,
                "stars": Number,
                "review_description": String
            }
    ],
    "first_name": String,
    "last_name": String,
}
```

```
Trips
{
    "_id": Number,
    "company_name": String,
    "tickets": [
            {
                "_id": Number,
                "reserved_seat_no": Number,
                "ticket_status": String
            }
    ],
    "date": Date,
    "location": String,
    "trip_reviews": [
            {
                "_id": Number,
                "customer_id": Number,
                "review_date": Date,
                "stars": Number,
                "review_description": String
            }
    ],
    "took_place": Boolean
}
```

Stworzenie bazy danych według wyżej przedstawionego pomysłu:

```
use doc_trip_database

db.createCollection("Customers")
db.createCollection("Trips")
```

Stworzenie rożnego rodzaju danych dla przykładowej bazy danych:

Kilka przydatnych informacji:

- took_place ustawione na True w kolekcji Trips oznacza, że wycieczka się już odbyła,
- review_id oraz ticket_id to liczby 3-cyfrowe zaczynające się od trip_id. 3 bilety dla np. trip_id=1 to: 101,102,103,
- seats_no to liczby odpowiadające numerom ticketów: np dla ticketa 102, seats_no to 2 itd
- ticket_status to jeden symbol spośród 'N', 'P' i 'C', oznaczających kolejno "New", "Paid" i "Canceled"

```
db.Customers.insertMany([
  { "_id": 1, "owned_tickets": [

  { "_id": 101, "trip_id": 1, "reserved_seat_no": 1, "ticket_status": 'P'},
  { "_id": 504, "trip_id": 5, "reserved_seat_no": 4, "ticket_status": 'C'}

  ], "customer_reviews": [
        { "review_id": 101, "trip_id": 1, "review_date": {"$date": "2023-03-01T00:00:00Z"}, "stars": 5,
"review_description": "Excellent experience!" }
  ], "first_name": "John", "last_name": "Doe" },

  { "_id": 2, "owned_tickets": [

  { "_id": 201, "trip_id": 2, "reserved_seat_no": 1, "ticket_status": 'P'}

  ], "customer_reviews": [

  { "review_id": 201, "customer_id": 2, "trip_id": 2, "review_date": {"$date": "2023-04-01T00:00:00Z"}, "stars":
3, "review_description": "Decent, but could be better." }

  ], "first_name": "Anna", "last_name": "Smith" },
  { "_id": 3, "owned_tickets": [

  { "_id": 102, "trip_id": 1, "reserved_seat_no": 2, "ticket_status": 'P'},
```

```
  { "_id": 301, "trip_id": 3, "reserved_seat_no": 1, "ticket_status": 'N'}

  ], "customer_reviews": [

  { "review_id": 102, "trip_id": 1, "review_date": {"$date": "2023-03-02T00:00:00Z"}, "stars": 4,
"review_description": "Very good trip, well organized." },
  { "review_id": 301, "trip_id": 3, "review_date": {"$date": "2023-04-26T00:00:00Z"}, "stars": 5,
"review_description": "It was super!" }


  ], "first_name": "David", "last_name": "Brown" },
  { "_id": 4, "owned_tickets": [

   { "_id": 501, "trip_id": 5, "reserved_seat_no": 1, "ticket_status": 'P'},
   { "_id": 502, "trip_id": 5, "reserved_seat_no": 2, "ticket_status": 'N'}

  ], "customer_reviews": [], "first_name": "Lisa", "last_name": "White" },
  { "_id": 5, "owned_tickets": [{ "_id": 503, "trip_id": 5, "reserved_seat_no": 1, "ticket_status": 'C'}],
"customer_reviews": [], "first_name": "Mark", "last_name": "Taylor" }
])

db.Trips.insertMany([
  { "_id": 1, "company_name": "Adventure Works", "tickets": [

  { "_id": 101, "customer_id": 1, "reserved_seat_no": 1, "ticket_status": 'P'},
  { "_id": 102, "customer_id": 3, "reserved_seat_no": 2, "ticket_status": 'P'}

  ], "date": {"$date": "2024-04-15T00:00:00Z"}, "location": "Warsaw", "trip_reviews": [

  { "review_id": 101, "customer_id": 1, "review_date": {"$date": "2023-03-01T00:00:00Z"}, "stars": 5,
"review_description": "Excellent experience!" },
  { "review_id": 102, "customer_id": 3, "review_date": {"$date": "2023-03-02T00:00:00Z"}, "stars": 4,
"review_description": "Very good trip, well organized." }


  ], "took_place": true },
  { "_id": 2, "company_name": "Adventure Works", "tickets": [

  { "_id": 201, "customer_id": 2, "trip_id": 2, "reserved_seat_no": 1, "ticket_status": 'P'}

  ], "date": {"$date": "2024-05-20T00:00:00Z"}, "location": "Krakow", "trip_reviews": [

  { "review_id": 201, "customer_id": 2, "review_date": {"$date": "2023-04-01T00:00:00Z"}, "stars": 3,
"review_description": "Decent, but could be better." }


  ], "took_place": true },
  { "_id": 3, "company_name": "Travel Corp", "tickets": [

  { "_id": 301, "customer_id": 3, "reserved_seat_no": 1, "ticket_status": 'N'}

  ], "date": {"$date": "2024-06-10T00:00:00Z"}, "location": "Gdansk", "trip_reviews": [

  { "review_id": 301, "customer_id": 3, "trip_id": 3, "review_date": {"$date": "2023-04-26T00:00:00Z"}, "stars":
5, "review_description": "It was super!" }

  ], "took_place": true },
  { "_id": 4, "company_name": "Travel Corp", "tickets": [], "date": {"$date": "2024-07-15T00:00:00Z"},
"location": "Poznan", "trip_reviews": [], "took_place": false },
  { "_id": 5, "company_name": "Holiday Makers", "tickets": [

  { "_id": 501, "customer_id": 4, "reserved_seat_no": 1, "ticket_status": 'P'},
  { "_id": 502, "customer_id": 4, "reserved_seat_no": 2, "ticket_status": 'N'},
  { "_id": 503, "customer_id": 5, "reserved_seat_no": 1, "ticket_status": 'C'},
  { "_id": 504, "customer_id": 1, "reserved_seat_no": 4, "ticket_status": 'C'}

  ], "date": {"$date": "2024-08-25T00:00:00Z"}, "location": "Wroclaw", "trip_reviews": [], "took_place": false },
  { "_id": 6, "company_name": "Globe Trotters", "tickets": [], "date": {"$date": "2024-09-14T00:00:00Z"},
"location": "Sopot", "trip_reviews": [], "took_place": false },
  { "_id": 7, "company_name": "Globe Trotters", "tickets": [], "date": {"$date": "2024-09-15T00:00:00Z"},
"location": "Gdynia", "trip_reviews": [], "took_place": false },
  { "_id": 8, "company_name": "Pathfinders", "tickets": [], "date": {"$date": "2024-09-29T00:00:00Z"},
```

```
"location": "Katowice", "trip_reviews": [], "took_place": false }
])
```

Kolekcje po wypełnieniu danymi:

- Customers

	_id	customer_reviews	first_name	last_name	owned_tickets
1	1	[{"review_id": new NumberInt("101"), "trip_id": new Numbe	John	Doe	[{"_id": new NumberInt("101"), "trip_id": new NumberInt("
2	2	[{"review_id": new NumberInt("201"), "customer_id": new N	Anna	Smith	[{"_id": new NumberInt("201"), "trip_id": new NumberInt("
3	3	[{"review_id": new NumberInt("102"), "trip_id": new Numbe	David	Brown	[{"_id": new NumberInt("102"), "trip_id": new NumberInt("
4	4	[]	Lisa	White	[{"_id": new NumberInt("501"), "trip_id": new NumberInt("
5	5	[]	Mark	Taylor	[{"_id": new NumberInt("503"), "trip_id": new NumberInt("

- Trips

	_id	compan...	date	location	tickets	took_place	trip_reviews
1	1	Adventure Wor…	{"$date": "2024-04-15	Warsaw	[{"_id": new NumberInt("101"), "customer_id	true	[{"review_id": new NumberInt("101"), "cu
2	2	Adventure Wor…	{"$date": "2024-05-20	Krakow	[{"_id": new NumberInt("201"), "customer_id	true	[{"review_id": new NumberInt("201"), "cu
3	3	Travel Corp	{"$date": "2024-06-10	Gdansk	[{"_id": new NumberInt("301"), "customer_id	true	[{"review_id": new NumberInt("301"), "cu
4	4	Travel Corp	{"$date": "2024-07-15	Poznan	[]	false	[]
5	5	Holiday Makers	{"$date": "2024-08-25	Wroclaw	[{"_id": new NumberInt("501"), "customer_id	false	[]
6	6	Globe Trotters	{"$date": "2024-09-14	Sopot	[]	false	[]
7	7	Globe Trotters	{"$date": "2024-09-15	Gdynia	[]	false	[]
8	8	Pathfinders	{"$date": "2024-09-29	Katowice	[]	false	[]

c) Analiza wad/zalet danego podejścia na konkretnych przykładach

- Wyświetlenie wszystkich ocen z `reviews` wraz z `company_name` dla każdej wycieczki `trip`, która już się odbyła

Dzięki temu, że mamy umieszczone te dane w jednym miejscu, zapytanie będzie dość proste, a czas wykonania polecenia krótki, ponieważ nie potrzebujemy łączyć kilku tabel, jak to było pokazane wyżej w przypadku innego podejścia

```
db.Trips.aggregate(
    {$match: {"took_place": true}},
    {$project: {company_name: 1, location: 1, "marks": "$trip_reviews.stars"}}
)
```

Wynik danego zypytania:

	_id	company_name	location	marks
1	1	Adventure Works	Warsaw	[new NumberInt("5"), new NumberInt("4")]
2	2	Adventure Works	Krakow	[new NumberInt("3")]
3	3	Travel Corp	Gdansk	[new NumberInt("5")]

- Wyświetlenie wszystkich biletów `tickets`, które mają status "Paid" albo "New"

W tym przypadku Customer agreguje Tickets, więc potrzebujemy bardziej skomplikowanych zapytań aby dostać się do informacji zamieszczonych w dokumentach zagnieżdżonych. Natomiast w strukturze tablicowej, jak było pokazane wyżej, takie zapytanie jest bardziej przejrzyste, ponieważ wszystkie dane są na jednym poziome w ramach jednej kolekcji.

```
db.Customers.aggregate(
    {$unwind: "$owned_tickets"},
    {$match: {$or: [{"owned_tickets.ticket_status": 'P'}, {"owned_tickets.ticket_status": 'N'}]}},
    {$project: {_id: 0, _id: "$owned_tickets._id", first_name: 1, last_name: 1, reserved_seat_no :
"$owned_tickets.reserved_seat_no",
    ticket_status: "$owned_tickets.ticket_status", trip_id: "$owned_tickets.trip_id"}}
)
```

Wynik danego zypytania:

	{} _id	{} first_name	{} last_name	{} reserved_seat_no	{} ticket_status	{} trip_id
1	101	John	Doe	1	P	1
2	102	David	Brown	2	P	1
3	201	Anna	Smith	1	P	2
4	301	David	Brown	1	N	3
5	501	Lisa	White	1	P	5
6	502	Lisa	White	2	N	5

- Dodanie nowego `review`

W podejściu dokumentowym potrzebujemy kilka razy (zależnie od zamodelowanej struktury bazy danych) wstawić dokument, zawierający informacje o danym review, do różnych kolekcji. W porównywaniu do sposoby tabelarycznego, tutaj będziemy potrzebowali zrobić kilka razy więcej insertów i odpowiednio wykorzystać kilka razy więcej pamięci do przechowywania redundantych danych, natomiast będą one wygodniejsze w wykorzystaniu.

Przed wstawieniem ustawiliśmy status wydarzenia na odbyty.

```
db.Trips.updateOne(
    { _id: 5 },
    { $set: { took_place: true } }
)
```

```
db.Customers.updateOne({"_id" : 1}, {$push: {customer_reviews: { "_id": 501, "trip_id": 5, "review_date": {"$date": "2024-05-03T00:00:00Z"}, "stars": 4.5, "review_description": "Great trip but it seems to be too expensive" }}})
db.Trips.updateOne({"_id" : 5}, {$push: {trip_reviews: { "_id": 501, "customer_id": 1, "review_date": {"$date": "2024-05-03T00:00:00Z"}, "stars": 4.5, "review_description": "Great trip but it seems to be too expensive" }}})
```

Wynik danego zypytania:

Customers:

		{} first_name	{} last_name	{} owned_tickets
1	, {"_id": new NumberInt("501"), "trip_id": new NumberInt("5"), "review_date": {"$date": "2024-05-03	John	Doe	[{"_id": new NumberInt(
2	cription": "Decent, but could be better."}]	Anna	Smith	[{"_id": new NumberInt(
3	ganized."}, {"review_id": new NumberInt("301"), "trip_id": new NumberInt("3"), "review_date": {"$da	David	Brown	[{"_id": new NumberInt(
4		Lisa	White	[{"_id": new NumberInt(
5		Mark	Taylor	[{"_id": new NumberInt(

Trips:

	{}₂ _id	{} company_name	{} date	{} loc...	{} tickets	{} took_place	{} trip_reviews
1	1	Adventure Works	{"$date": "2024-04-15	Warsaw	[{"_id": new	true	[{"review_id": new NumberInt("101"), "customer_id": new Nu
2	2	Adventure Works	{"$date": "2024-05-20	Krakow	[{"_id": new	true	[{"review_id": new NumberInt("201"), "customer_id": new Nu
3	3	Travel Corp	{"$date": "2024-06-10	Gdansk	[{"_id": new	true	[{"review_id": new NumberInt("301"), "customer_id": new Nu
4	4	Travel Corp	{"$date": "2024-07-15	Poznan	[]	false	[]
5	5	Holiday Makers	{"$date": "2024-08-25	Wroclaw	[{"_id": new	true	[{"_id": new NumberInt("501"), "customer_id": new NumberIn
6	6	Globe Trotters	{"$date": "2024-09-14	Sopot	[]	false	[]
7	7	Globe Trotters	{"$date": "2024-09-15	Gdynia	[]	false	[]
8	8	Pathfinders	{"$date": "2024-09-29	Katowice	[]	false	[]

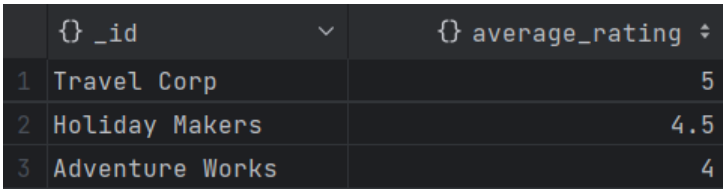
- Wyświetlenie średniej oceny dla każdej firmy `company_name` na podstawie wszystkich ocen z `reviews` dla każdej wycieczki `trip`, która już się odbyła

W porównywaniu do podejścia tabelarycznego, tutaj polecenie jest prostsze i wydajniejsze, skoro mamy informacje w jednej kolekcji i nie potrzebujemy używać kosztowny operator łączenia.

```
db.Trips.aggregate(
    {$match: {"took_place": true}},
    {$project: {company_name: 1, "marks": "$trip_reviews.stars"}},
    {$unwind: "$marks"},
    {$group: {
        _id: "$company_name",
        average_rating: {$avg: "$marks"}
```

```
        }}
)
```

Wynik danego zypytania:



	{} _id	{} average_rating
1	Travel Corp	5
2	Holiday Makers	4.5
3	Adventure Works	4

Wnioski

W ramach danego ćwiczenia przetestowaliśmy działanie różnych operatorów do wyszukiwania danych oraz rozważyliśmy różne podejścia do modelowania dokumentowej bazy danych w MongoDB. Po przeprowadzeniu różnych eksperymentów, możemy wyciągnąć wniosek, że dokumentowe bazy danych są "samoopisujące się" i bardzo przydatne. Na podstawie zrealizowanego zadania №2 możemy stwierdzić, że każde podejście ma swoje wady i zalety. W przypadku, gdy dane są często modyfikowane, a zapytania do kilku encji są rzadkie - najlepszym sposobem jest "tabelaryczna" baza danych (podobnie do Transact SQL, dane każdej encji są umieszczone w różnych kolekcjach), natomiast w przypadku, gdy dane są modyfikowane rzadko, ale często korzystamy z różnych relacji, lepszym wariantem będzie "dokumentowe" podejście (dane są umieszczone w dokumentach zagnieżdżonych).

Punktacja:

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