Dokumentowe bazy danych – MongoDB

Ćwiczenie/zadanie

Imiona i nazwiska autorów: Paweł Gadomski, Jakub Stachecki

Odtwórz z backupu bazę north0

```
mongorestore --nsInclude='north0.*' ./dump/
```

```
use north0
```

Zadanie 1 - operacje wyszukiwania danych, przetwarzanie dokumentów

a)

stwórz kolekcję `OrdersInfo` zawierającą następujące dane o zamówieniach

- pojedynczy dokument opisuje jedno zamówienie

```
[
    {
        "_id": ...

            OrderID": ... numer zamówienia
```

```
"Customer": {  ... podstawowe informacje o kliencie skladającym
    "CustomerID": ... identyfikator klienta
    "CompanyName": ... nazwa klienta
    "City": ... miasto
    "Country": ... kraj
},

"Employee": {  ... podstawowe informacje o pracowniku obsługującym zamówienie
    "EmployeeID": ... idntyfikator pracownika
    "FirstName": ... imie
    "LastName": ... nazwisko
    "Title": ... stanowisko

},

"Dates": {
    "OrderDate": ... data złożenia zamówienia
    "RequiredDate": data wymaganej realizacji
}

"Orderdetails": [  ... pozycje/szczegóły zamówienia - tablica takich pozycji
    {
        "UnitPrice": ... cena
        "Quantity": ... liczba sprzedanych jednostek towaru
        "Discount": ... zniżka
        "Value": ... wartośc pozycji zamówienia
        "product": { ... podstawowe informacje o produkcie
            "ProductID": ... identyfikator produktu
            "ProductName": ... nazwa produktu
            "QuantityPerUnit": ... opis/opakowannie
            "CategoryID": ... identyfikator kategorii do której należy produkt
            "CategoryName" ... nazwę tej kategorii
        },
    },
    ...
```

```
        ],

        "Freight": ... opłata za przesyłkę
        "OrderTotal"  ... sumaryczna wartosc sprzedanych produktów

        "Shipment" : {  ... informacja o wysyłce
            "Shipper": { ... podstawowe inf o przewoźniku
                "ShipperID":
                    "CompanyName":
}
... inf o odbiorcy przesyłki
"ShipName": ...
"ShipAddress": ...
"ShipCity": ...
"ShipCountry": ...
}
}
]
```

b)

stwórz kolekcję `CustomerInfo` zawierającą następujące dane kazdym klencie

- pojedynczy dokument opisuje jednego klienta

```
[
    {
        "_id": ...

            "CustomerID": ... identyfikator klienta
"CompanyName": ... nazwa klienta
"City": ... miasto
"Country": ... kraj
```

```
"Orders": [ ... tablica zamówień klienta o strukturze takiej jak w punkcie a) (oczywiście bez informacji o kliencie)

]


]
```

c)

Napisz polecenie/zapytanie: Dla każdego klienta pokaż wartość zakupionych przez niego produktów z kategorii 'Confections' w 1997r

- Spróbuj napisać to zapytanie wykorzystując
    - oryginalne kolekcje (`customers, orders, orderdertails, products, categories`)
    - kolekcję `OrderInfo`
    - kolekcję `CustomerInfo`
- porównaj zapytania/polecenia/wyniki

```
[
    {
        "_id":

            "CustomerID": ... identyfikator klienta
"CompanyName": ... nazwa klienta
"ConfectionsSale97": ... wartość zakupionych przez niego produktów z kategorii 'Confections'  w 1997r

}
]
```

d)

Napisz polecenie/zapytanie: Dla każdego klienta poaje wartość sprzedaży z podziałem na lata i miesiące Spróbuj napisać to zapytanie wykorzystując

- oryginalne kolekcje (`customers, orders, orderdertails, products, categories`)
- kolekcję `OrderInfo`
- kolekcję `CustomerInfo`
- porównaj zapytania/polecenia/wyniki

```
[
    {
        "_id":

            "CustomerID": ... identyfikator klienta
"CompanyName": ... nazwa klienta

"Sale": [ ... tablica zawierająca inf o sprzedazy
    {
        "Year":  ....
        "Month": ....
        "Total": ...
    }
    ...
]
}
]
```

e)

Załóżmy że pojawia się nowe zamówienie dla klienta 'ALFKI', zawierające dwa produkty 'Chai' oraz "Ikura"

- pozostałe pola w zamówieniu (ceny, liczby sztuk prod, inf o przewoźniku itp. możesz uzupełnić wg własnego uznania) Napisz polecenie które dodaje takie zamówienie do bazy
- aktualizując oryginalne kolekcje `orders`, `orderdetails`
- aktualizując kolekcję `OrderInfo`
- aktualizując kolekcję `CustomerInfo`

Napisz polecenie

- aktualizując oryginalną kolekcję orderdetails`
- aktualizując kolekcję `OrderInfo`
- aktualizując kolekcję `CustomerInfo`

f)

Napisz polecenie które modyfikuje zamówienie dodane w pkt e) zwiększając zniżkę o 5% (dla każdej pozycji tego zamówienia)

Napisz polecenie

- aktualizując oryginalną kolekcję `orderdetails`
- aktualizując kolekcję `OrderInfo`
- aktualizując kolekcję `CustomerInfo`

UWAGA: W raporcie należy zamieścić kod poleceń oraz uzyskany rezultat, np wynik polecenia `db.kolekcka.fimd().limit(2)` lub jego fragment

Zadanie 1 - rozwiązanie

> Wyniki:
>
> przykłady, kod, zrzuty ekranów, komentarz ...

a)

```
db.orders.aggregate([
    {
        $match: { }
    },
    {
        $lookup: {
            from: "customers",
            localField: "CustomerID",
            foreignField: "CustomerID",
            as: "Customer_tmp"
        }
    },
    {
        $unwind: "$Customer_tmp"
    },
    {
        $lookup: {
            from: "employees",
            localField:"EmployeeID",
            foreignField: "EmployeeID",
            as: "Employee_tmp"
        }
    },
    {
        $unwind: "$Employee_tmp"
    },
    {
        $lookup: {
            from: "orderdetails",
            localField: "OrderID",
            foreignField: "OrderID",
            as: "OrderDetails_tmp"
        }
    },
    {
```

```
        $unwind: "$OrderDetails_tmp"
    },
    {
        $lookup: {
            from: "products",
            localField: "OrderDetails_tmp.ProductID",
            foreignField: "ProductID",
            as: "Product_tmp"
        }
    },
    {
        $unwind: "$Product_tmp"
    },
    {
        $lookup: {
            from: "categories",
            localField: "Product_tmp.CategoryID",
            foreignField: "CategoryID",
            as: "Category_tmp"
        }
    },
    {
        $unwind: "$Category_tmp"
    },
    {
        $lookup: {
            from: "shippers",
            localField: "ShipVia",
            foreignField: "ShipperID",
            as: "Shipper_tmp"
        }
    },
    {
        $unwind: "$Shipper_tmp"
    },
    {
```

```
        $addFields: {
            Customer: {
                CustomerID: "$Customer_tmp.CustomerID",
                CompanyName: "$Customer_tmp.CompanyName",
                City: "$Customer_tmp.City",
                Country: "$Customer_tmp.Country",
            },
            Employee: {
                EmployeeID: "$Employee_tmp.EmployeeID",
                FirstName: "$Employee_tmp.FirstName",
                LastName: "$Employee_tmp.LastName",
                Title: "$Employee_tmp.Title"
            },
            Dates: {
                OrderDate: "$OrderDate",
                RequiredDate: "$RequiredDate"
            },
            OrderDetails: {
                UnitPrice: "$OrderDetails_tmp.UnitPrice",
                Quantity: "$OrderDetails_tmp.Quantity",
                Discount: "$OrderDetails_tmp.Discount",
                Value: {
                    $multiply: [ "$OrderDetails_tmp.UnitPrice", "$OrderDetails_tmp.Quantity", { $subtract: [1,
"$OrderDetails_tmp.Discount"] } ]
                },
                Product: {
                    ProductID: "$Product_tmp.ProductID",
                    ProductName: "$Product_tmp.ProductName",
                    QuantityPerUnit: "$Product_tmp.QuantityPerUnit",
                    CategoryID: "$Product_tmp.CategoryID",
                    CategoryName: "$Category_tmp.CategoryName"
                }
            },
            Shipment: {
                Shipper: {
                    ShipperID: "$Shipper_tmp.ShipperID",
```

```
                    CompanyName: "$Shipper_tmp.CompanyName"
                },
                ShipName: "$ShipName",
                ShipAddress: "$ShipAddress",
                ShipCity: "$ShipCity",
                ShipCountry: "$ShipCountry"
            }
        }
    },
    {
        $group: {
            _id: "$OrderID",
            OrderTotal: { $sum : "$OrderDetails.Value"  },
            OrderDetails: { $push : "$OrderDetails" },
            OrderID: { $first: "$OrderID" },
            Customer: { $first: "$Customer" },
            Employee: { $first: "$Employee" },
            Dates: { $first: "$Dates" },
            Freight: { $first: "$Freight" },
            Shipment: { $first: "$Shipment" }
        }
    },
    {
        $out: "OrdersInfo"
    }
])
```

b)

```
db.OrdersInfo.aggregate([
    {
        $group: {
            _id: "$Customer.CustomerID",
```

```
            CustomerID: { $first: "$Customer.CustomerID" },
            CompanyName: { $first: "$Customer.CompanyName" },
            City: { $first: "$Customer.City" },
            Country: { $first: "$Customer.Country" },
            Orders: {
                $push: {
                    OrderID: "$OrderID",
                    OrderTotal: "$OrderTotal",
                    OrderDetails: "$OrderDetails",
                    Employee: "$Employee",
                    Dates: "$Dates",
                    Freight: "$Freight",
                    Shipment: "$Shipment"
                }
            }
        }
    },
    {
        $out: "CustomerInfo"
    }
])
```

c)

```
// Oryginalne kolekcje:
db.orders.aggregate([
    {
        $lookup: {
            from: "orderdetails",
            localField: "OrderID",
            foreignField: "OrderID",
            as: "Orderdetails_tmp"
        }
    },
```

```
    {
        $unwind: "$Orderdetails_tmp"
    },
    {
        $lookup: {
            from: "products",
            localField: "Orderdetails_tmp.ProductID",
            foreignField: "ProductID",
            as: "Products_tmp"
        }
    },
    {
        $unwind: "$Products_tmp"
    },
    {
        $lookup: {
            from: "categories",
            localField: "Products_tmp.CategoryID",
            foreignField: "CategoryID",
            as: "Category_tmp"
        }
    },
    {
        $unwind: "$Category_tmp"
    },
    {
        $match: {
            "Category_tmp.CategoryName": "Confections",
            $expr: {
                $eq: [{ $year: "$OrderDate" }, 1997]
            }
        }
    },
    {
        $addFields: {
            Value: {
```

```
                $multiply: [ "$Orderdetails_tmp.UnitPrice", "$Orderdetails_tmp.Quantity", { $subtract: [1,
"$Orderdetails_tmp.Discount"] } ]
            },
        }
    },
    {
        $lookup: {
            from: "customers",
            localField: "CustomerID",
            foreignField: "CustomerID",
            as: "Customer_tmp"
        }
    },
    {
        $unwind: "$Customer_tmp"
    },
    {
        $group: {
            _id: "$CustomerID",
            CustomerID: { $first: "$CustomerID" },
            CompanyName: { $first: "$Customer_tmp.CompanyName" },
            ConfectionsSale97: { $sum: "$Value" }
        }
    },
    {
        $project: {
            _id: 0,
            CustomerID: 1,
            CompanyName: 1,
            ConfectionsSale97: { $round: ["$ConfectionsSale97", 2] }
        }
    }
])

// Kolekcja OrdersInfo
db.OrdersInfo.aggregate([
```

```
    {
        $match: {
            $expr: { $eq: [ {$year: "$Dates.OrderDate"}, 1997 ] },
        }
    },
    {
        $addFields: {
            filteredOrderDetails: {
                $filter: {
                    input: "$OrderDetails",
                    as: "od",
                    cond: { $eq: [ "$$od.Product.CategoryName", "Confections" ] }
                }
            }
        }
    },
    {
        $addFields: {
            ConfectionsSale97: {
                $sum: "$filteredOrderDetails.Value"
            },
            CustomerID: "$Customer.CustomerID",
            CompanyName: "$Customer.CompanyName"
        }
    },
    {
        $match: {
            ConfectionsSale97: { $gt: 0 }
        }
    },
    {
        $group: {
            _id: "$CustomerID",
            CustomerID: { $first: "$CustomerID" },
            CompanyName: { $first: "$Customer.CompanyName" },
            ConfectionsSale97: { $sum: "$ConfectionsSale97" }
```

```
            }
        },
        {
            $project: {
                _id: 0,
                CustomerID: 1,
                CompanyName: 1,
                ConfectionsSale97: { $round: ["$ConfectionsSale97", 2] }
            }
        }
    ])

    // Kolekcja CustomerInfo
    db.CustomerInfo.aggregate([
        {
            $addFields: {
                Orders97: {
                    $filter: {
                        input: "$Orders",
                        as: "od",
                        cond: { $eq: [ {$year: "$$od.Dates.OrderDate"}, 1997 ] }
                    }
                }
            }
        },
        {
            $unwind: "$Orders97"
        },
        {
            $addFields: {
                Confections97: {
                    $filter: {
                        input: "$Orders97.OrderDetails",
                        as: "od",
                        cond: { $eq: [ "$$od.Product.CategoryName", "Confections" ] }
                    }
```

```
            }
        }
    },
    {
        $match: {
            Confections97: { $ne: [], $exists: true }
        }
    },
    {
        $unwind: "$Confections97"
    },
    {
        $group: {
            _id: "$CustomerID",
            CustomerID: { $first: "$CustomerID" },
            CompanyName: { $first: "$CompanyName"},
            ConfectionsSale97: { $sum: "$Confections97.Value" }
        }
    },
    {
        $project: {
            _id: 0,
            CustomerID: 1,
            CompanyName: 1,
            ConfectionsSale97: { $round: ["$ConfectionsSale97", 2] }
        }
    }
])
```

Porównanie zapytań:

Oryginalne kolekcje:

Zalety: źródłowe dane, bez redundacji. Wady: Ciężka agregacja, dużo lookup i unwind oraz wolniejsze dla dużych danych

Kolekcja OrdersInfo:

Zalety: Brak lookupów, dużo prostsze zapytanie niż w wersji z oryginalnymi kolekcjami Wady: Musimy wcześneij przygotować tą kolekcje na podstawie danych źródłowych, trzeba pamiętać o jej aktualizacji w przypadku zmiany danych oryginalnych kolekcji

Kolekcja CustomerInfo

Zalety: Brak lookupów, dużo prostsze zapytanie niż w wersji z oryginalnymi kolekcjami oraz subiektywnie nieco prostsze niz OrdersInfo Wady: Redundacja jak w przypadku powyżej, i problemy z aktualizacją w wypadku aktualizacji oryginalnych danych

Porównanie wyników:

W każdym z 3 przypadków wyniki są identyczne i oto przykładowe 3 z nich:


```
[
  {
    "CompanyName": "Antonio Moreno Taquería",
    "ConfectionsSale97": 958.93,
    "CustomerID": "ANTON"
  },
  {
    "CompanyName": "Around the Horn",
    "ConfectionsSale97": 375.2,
    "CustomerID": "AROUT"
  },
  {
    "CompanyName": "Berglunds snabbköp",
    "ConfectionsSale97": 561.96,
    "CustomerID": "BERGS"
  }
]
```


d)

```
// Oryginalne kolekcje
db.orders.aggregate([
    {
        $lookup: {
            from: "orderdetails",
            localField: "OrderID",
            foreignField: "OrderID",
            as: "Details"
        }
    },
    {
        $unwind: "$Details"
    },
    {
        $lookup: {
            from: "products",
            localField: "Details.ProductID",
            foreignField: "ProductID",
            as: "Product"
        }
    },
    {
        $unwind: "$Product"
    },
    {
        $lookup: {
            from: "customers",
            localField: "CustomerID",
            foreignField: "CustomerID",
            as: "Customer"
        }
    },
    {
        $unwind: "$Customer"
    },
```

```
    {
        $addFields: {
            Year: { $year: "$OrderDate" },
            Month: { $month: "$OrderDate" },

            Value: {
                $multiply: [
                    "$Details.UnitPrice",
                    "$Details.Quantity",
                    {$subtract: [1, "$Details.Discount"] }
                ]
            }
        }
    },
    {
        $group: {
            _id: {
                CustomerID: "$Customer.CustomerID",
                Year: "$Year",
                Month: "$Month"
            },
            CompanyName: { $first: "$Customer.CompanyName" },
            Total: { $sum: "$Value" }
        }
    },
    {
        $group: {
            _id: "$_id.CustomerID",
            CustomerID: { $first: "$_id.CustomerID" },
            CompanyName: { $first: "$CompanyName" },
            Sale: {
                $push: {
                    Year: "$_id.Year",
                    Month: "$_id.Month",
                    Total: { $round: ["$Total", 2] }
                }
```

```
            }
        }
    }
])

// OrdersInfo

db.OrdersInfo.aggregate([
    {
        $project: {
            customerID: "$Customer.CustomerID",
            companyName: "$Customer.CompanyName",
            year: { $year: "$Dates.OrderDate" },
            month: { $month: "$Dates.OrderDate" },
            orderTotal: "$OrderTotal"
        }
    },
    {
        $group: {
            _id: {
                customerID: "$customerID",
                year: "$year",
                month: "$month"
            },
            companyName: { $first: "$companyName" },
            Total: { $sum: "$orderTotal" }
        }
    },
    {
        $group: {
            _id: "$_id.customerID",
            CustomerID: { $first: "$_id.customerID"},
            CompanyName: { $first: "$companyName" },
            Sale: {
                $push: {
                    Year: "$_id.year",
```

```
                    Month: "$_id.month",
                    Total: {$round: ["$Total", 2]}
                }
            }
        }
    }
])

//CustomerInfo

db.CustomerInfo.aggregate([
    { $unwind: "$Orders" },
    {
        $project: {
            CustomerID: 1,
            CompanyName: 1,
            Year: { $year: "$Orders.Dates.OrderDate" },
            Month: { $month: "$Orders.Dates.OrderDate" },
            OrderTotal: "$Orders.OrderTotal"
        }
    },
    {
        $group: {
            _id: {
                CustomerID: "$CustomerID",
                Year: "$Year",
                Month: "$Month"
            },
            CompanyName: { $first: "$CompanyName" },
            TotalSales: { $sum: "$OrderTotal" }
        }
    },
    {
        $group: {
            _id: "$_id.CustomerID",
            CustomerID: {$first: "$_id.CustomerID" },
```

```
            CompanyName: { $first: "$CompanyName" },
            Sale: {
                $push: {
                    Year: "$_id.Year",
                    Month: "$_id.Month",
                    Total: {$round: ["$TotalSales", 2]}
                }
            }
        }
    }
])
```

Porównanie zapytań:

Wady i zalety poszczególnych zapytań są identyczne do tych z zadania **c)**

Oryginalne kolekcje:

Zalety: źródłowe dane, bez redundacji. Wady: Ciężka agregacja, dużo lookup i unwind oraz wolniejsze dla dużych danych

Kolekcja OrdersInfo:

Zalety: Brak lookupów, dużo prostsze zapytanie niż w wersji z oryginalnymi kolekcjami Wady: Musimy wcześneij przygotować tą kolekcje na podstawie danych źródłowych, trzeba pamiętać o jej aktualizacji w przypadku zmiany danych oryginalnych kolekcji

Kolekcja CustomerInfo

Zalety: Brak lookupów, dużo prostsze zapytanie niż w wersji z oryginalnymi kolekcjami oraz subiektywnie nieco prostsze niz OrdersInfo Wady: Redundacja jak w przypadku powyżej, i problemy z aktualizacją w wypadku aktualizacji oryginalnych danych

Porównanie wyników:

W każdym z 3 przypadków wyniki są praktycznie identyczne, jedyne różnice mogą być widoczne w kolejności dokumentów w tablicy Sale i oto przykładowe 3 z nich:


```
[
  {
    "_id": "ALFKI",
    "CompanyName": "Alfreds Futterkiste",
    "CustomerID": "ALFKI",
    "Sale": [
      {
        "Year": 1998,
        "Month": 3,
        "Total": 471.2
      },
      {
        "Year": 1998,
        "Month": 4,
        "Total": 933.5
      },
      {
        "Year": 1997,
        "Month": 8,
        "Total": 814.5
      },
      {
        "Year": 1998,
        "Month": 1,
        "Total": 845.8
      },
      {
        "Year": 1997,
        "Month": 10,
        "Total": 1208
      }
    ]
```

```
    },
    {
      "_id": "ANATR",
      "CompanyName": "Ana Trujillo Emparedados y helados",
      "CustomerID": "ANATR",
      "Sale": [
        {
          "Year": 1998,
          "Month": 3,
          "Total": 514.4
        },
        {
          "Year": 1997,
          "Month": 11,
          "Total": 320
        },
        {
          "Year": 1996,
          "Month": 9,
          "Total": 88.8
        },
        {
          "Year": 1997,
          "Month": 8,
          "Total": 479.75
        }
      ]
    },
    {
      "_id": "ANTON",
      "CompanyName": "Antonio Moreno Taquería",
      "CustomerID": "ANTON",
      "Sale": [
        {
          "Year": 1998,
          "Month": 1,
```

```
        "Total": 660
      },
      {
        "Year": 1997,
        "Month": 6,
        "Total": 2082
      },
      {
        "Year": 1997,
        "Month": 9,
        "Total": 1188.86
      },
      {
        "Year": 1996,
        "Month": 11,
        "Total": 403.2
      },
      {
        "Year": 1997,
        "Month": 5,
        "Total": 1940.85
      },
      {
        "Year": 1997,
        "Month": 4,
        "Total": 749.06
      }
    ]
  }
]
```


e)

```
// aktualizując oryginalne kolekcje `orders`, `orderdetails`
const alfki = db.customers.findOne({ CustomerID: "ALFKI" });
const chaiProductInfo = db.products.findOne({ ProductName: "Chai" });
const ikuraProductInfo = db.products.findOne({ ProductName: "Ikura" });

const newOrderID = db.OrdersInfo.find().sort({_id: -1}).limit(1).toArray()[0].OrderID + 1;


const orderResult = db.orders.insertOne({
    OrderID: newOrderID,
    CustomerID: "ALFKI",
    EmployeeID: 3,
    OrderDate: new Date(2024, 4, 11),
    RequiredDate: new Date(2025, 5, 11),
    ShippedDate: new Date(2025, 4, 8),
    ShipVia: 2,
    Freight: 15.80,
    ShipName: "Myslovitz",
    ShipAddress: "Szkolna 17",
    ShipCity: "Białystok",
    ShipRegion: null,
    ShipPostalCode: "23-535",
    ShipCountry: "Poland"
})

db.orderdetails.insertMany([{
    Discount: 0,
    OrderID: db.orders.find().sort({ OrderID: -1 }).limit(1).toArray()[0].OrderID,
    ProductID: chaiProductInfo.ProductID,
    Quantity: 5,
    UnitPrice: chaiProductInfo.UnitPrice
},
    {
        Discount: 0,
        OrderID: db.orders.find().sort({ OrderID: -1 }).limit(1).toArray()[0].OrderID,
```

```
        ProductID: ikuraProductInfo.ProductID,
        Quantity: 5,
        UnitPrice: ikuraProductInfo.UnitPrice
    }])

// aktualizując kolekcję `OrderInfo`
const employee = db.employees.findOne({ EmployeeID: 4 });
const newOrderID = db.OrdersInfo.find().sort({_id: -1}).limit(1).toArray()[0].OrderID + 1;

const chaiValue = chaiProductInfo.UnitPrice * 5;
const ikuraValue = ikuraProductInfo.UnitPrice * 10;
const orderTotal = chaiValue + ikuraValue;

db.OrdersInfo.insertOne({
    Customer: {
        CustomerID: alfki.CustomerID,
        CompanyName: alfki.CompanyName,
        City: alfki.City,
        Country: alfki.Country
    },
    Dates: {
        OrderDate: new Date(2024, 4, 11),
        RequiredDate: new Date(2025, 5, 11)
    },
    Employee: {
        EmployeeID: employee.EmployeeID,
        FirstName: employee.FirstName,
        LastName: employee.LastName,
        Title: employee.Title
    },
    Freight: 312.3,
    OrderDetails: [
        {
            UnitPrice: chaiProductInfo.UnitPrice,
            Quantity: 5,
            Discount: 0,
```

```
            Value: chaiValue,
            Product: {
                ProductID: chaiProductInfo.ProductID,
                ProductName: chaiProductInfo.ProductName,
                QuantityPerUnit: chaiProductInfo.QuantityPerUnit,
                CategoryID: chaiProductInfo.CategoryID,
                CategoryName: db.categories.find( { CategoryID: chaiProductInfo.CategoryID } ).toArray()[0].CategoryName
            }
        },
        {
            UnitPrice: ikuraProductInfo.UnitPrice,
            Quantity: 10,
            Discount: 0,
            Value: ikuraValue,
            Product: {
                ProductID: ikuraProductInfo.ProductID,
                ProductName: ikuraProductInfo.ProductName,
                QuantityPerUnit: ikuraProductInfo.QuantityPerUnit,
                CategoryID: ikuraProductInfo.CategoryID,
                CategoryName: db.categories.find( { CategoryID: ikuraProductInfo.CategoryID } ).toArray()[0].CategoryName
            }
        }
    ],
    OrderID: newOrderID,
    OrderTotal: orderTotal,
    Shipment: {
        Shipper: {
            ShipperID: 30,
            CompanyName: "Kult"
        },
        ShipName: "Brooklyn",
        ShipAddress: "Bruhhh",
        ShipCity: "Kraków",
        ShipCountry: "Poland",
    }
})
```

```
const newOrderID = db.OrdersInfo.find().sort({_id: -1}).limit(1).toArray()[0].OrderID + 1;
// aktualizując kolekcje CustomerInfo
db.CustomerInfo.updateOne({ CustomerID: "ALFKI" }, {
    $push: {
        Orders: {
            OrderID: newOrderID,
            Dates: {
                OrderDate: new Date(2024, 4, 11),
                ShippedDate: new Date(2025, 4, 11)
            },
            Freight: 323,
            OrderTotal: orderTotal,
            Shipment: {
                Shipper: {
                    ShipperID: 1,
                    CompanyName: db.shippers.find({ ShipperID: 1 }).toArray()[0].CompanyName
                },
                ShipName: "Taco",
                ShipCity: "Warsaw",
                ShipCountry: "Poland",
            },
            Employee: {
                EmployeeID: employee.EmployeeID,
                FirstName: employee.FirstName,
                LastName: employee.LastName,
                Title: employee.Title
            },
            OrderDetails: [
            {
                Product: {
                    ProductID: chaiProductInfo.ProductID,
                    ProductName: chaiProductInfo.ProductName,
                    QuantityPerUnit: chaiProductInfo.QuantityPerUnit,
                    CategoryID: chaiProductInfo.CategoryID,
```

```
                    CategoryName: db.categories.find( { CategoryID :  chaiProductInfo.CategoryID } ).toArray()
[0].CategoryName
                },
                UnitPrice: chaiProductInfo.UnitPrice,
                Quantity: 5,
                Discount: 0,
                Value: chaiValue
            },
            {
                Product: {
                    ProductID: ikuraProductInfo.ProductID,
                    ProductName: ikuraProductInfo.ProductName,
                    QuantityPerUnit: ikuraProductInfo.QuantityPerUnit,
                    CategoryID: ikuraProductInfo.CategoryID,
                    CategoryName: db.categories.find( { CategoryID :  ikuraProductInfo.CategoryID } ).toArray()
[0].CategoryName
                },
                UnitPrice: ikuraProductInfo.UnitPrice,
                Quantity: 10,
                Discount: 0,
                Value: ikuraValue
            },
            ]
        }
    }
})
```

Komentarz:

Dodawanie produktów ma podobny poziom skomplikowaności we wszystkich przypadkach natomiast dla aktulizowania kolekcji OrdersInfo i CustomerInfo potrzebujemy znać wartość zamówienia która musi być wczesniej obliczona

f)

```
// Oryginalne kolekcje
db.orderdetails.updateMany(
  { OrderID: newOrderID },
  { $inc: { Discount: 0.05 } }
)

// Kolekcja OrdersInfo
db.OrdersInfo.updateOne(
  { OrderID: newOrderID },
  [{
    $set: {
      OrderDetails: {
        $map: {
          input: "$OrderDetails",
          as: "od",
          in: {
            $mergeObjects: [
              "$$od",
              {
                Discount: { $add: ["$$od.Discount", 0.05] },
                Value: {
                  $multiply: [
                    "$$od.UnitPrice",
                    "$$od.Quantity",
                    { $subtract: [1, { $add: ["$$od.Discount", 0.05] }] }
                  ]
                }
              }
            ]
          }
        }
      },
      OrderTotal: {
        $sum: {
          $map: {
```

```
            input: "$OrderDetails",
            as: "od",
            in: {
              $multiply: [
                "$$od.UnitPrice",
                "$$od.Quantity",
                { $subtract: [1, { $add: ["$$od.Discount", 0.05] }] }
              ]
            }
          }
        }
      }
    }
  }]
)

// Kolekcja CustomerInfo
db.CustomerInfo.updateOne(
  {
    "CustomerID": "ALFKI",
    "Orders.OrderID": newOrderID
  },
  [{
    $set: {
      Orders: {
        $map: {
          input: "$Orders",
          as: "order",
          in: {
            $cond: [
              { $eq: ["$$order.OrderID", newOrderID] },
              {
                $mergeObjects: [
                  "$$order",
                  {
                    "OrderDetails": {
```

```
                $map: {
                  input: "$$order.OrderDetails",
                  as: "od",
                  in: {
                    $mergeObjects: [
                      "$$od",
                      {
                        Discount: { $add: ["$$od.Discount", 0.05] },
                        Value: {
                          $multiply: [
                            "$$od.UnitPrice",
                            "$$od.Quantity",
                            { $subtract: [1, { $add: ["$$od.Discount", 0.05] }] }
                          ]
                        }
                      }
                    ]
                  }
                }
              },
              OrderTotal: {
                $sum: {
                  $map: {
                    input: "$$order.OrderDetails",
                    as: "od",
                    in: {
                      $multiply: [
                        "$$od.UnitPrice",
                        "$$od.Quantity",
                        { $subtract: [1, { $add: ["$$od.Discount", 0.05] }] }
                      ]
                    }
                  }
                }
              }
            }
```

```
                ]
              },
              "$$order"
            ]
          }
        }
      }
    }
  }]
)
```

Komentarz:

Widzimy, że aktualizowanie zniżki dla kolekcji CustomerInfo jest znacznie bardziej skomplikowane niż dla OrdersInfo natomiast najprościej jest zaaktualizować zniżke dla oryginalnych danych.

Zadanie 2 - modelowanie danych

Zaproponuj strukturę bazy danych dla wybranego/przykładowego zagadnienia/problemu

Należy wybrać jedno zagadnienie/problem (A lub B lub C)

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

Przykład C

- Własny przykład o podobnym stopniu złożoności

a) Zaproponuj różne warianty struktury bazy danych i dokumentów w poszczególnych kolekcjach oraz przeprowadzić dyskusję każdego wariantu (wskazać wady i zalety każdego z wariantów)

- zdefiniuj schemat/reguły walidacji danych
- wykorzystaj referencje
- dokumenty zagnieżdżone
- tablice

b) Kolekcje należy wypełnić przykładowymi danymi

c) W kontekście zaprezentowania wad/zalet należy zaprezentować kilka przykładów/zapytań/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 w wersji źródłowej (np. plik .js, np. plik .md), załącznik powinien mieć format zip

Zadanie 2 - rozwiązanie

> Wyniki:
>
> przykłady, kod, zrzuty ekranów, komentarz ...

Wariant 1: Struktura z oddzielnymi kolekcjami i referencjami

W tym podejściu dane są podzielone na odrębne kolekcje, które są ze sobą powiązane za pomocą referencji.

Kolekcja Users

```
{
  "_id": ObjectId("5f8a7b2d9d3e7a1c3c7b4a1c"),
         "username": "jan_kowalski",
         "email": "jan.kowalski@example.com",
         "password": "$2a$10$X7JhbS1Ehp8YK4Xo9jM1e.Lqz4DZM6rXV8wY9X9Y8X7X8X7X8X7X8",
         "displayName": "Jan Kowalski",
         "bio": "Pasjonat technologii i kawy",
         "joinDate": ISODate("2023-01-15T12:00:00Z"),
         "followersCount": 120,
         "followingCount": 85,
         "postsCount": 45
}
```

Kolekcja Posts

```
{
  "_id": ObjectId("6a1b2c3d4e5f6a7b8c9d0e1f"),
         "userId": ObjectId("5f8a7b2d9d3e7a1c3c7b4a1c"),
         "content": "Dzisiaj jest piękny dzień na programowanie!",
         "createdAt": ISODate("2023-05-20T14:30:00Z"),
         "updatedAt": ISODate("2023-05-20T14:30:00Z"),
         "likesCount": 25,
         "commentsCount": 8,
         "maxComments": 20,  // Limit komentarzy ustawiony przez użytkownika
         "tags": ["programowanie", "technologia"]
}
```

Kolekcja Comments

```
{
  "_id": ObjectId("7b8c9d0e1f2a3b4c5d6e7f8a"),
         "postId": ObjectId("6a1b2c3d4e5f6a7b8c9d0e1f"),
         "userId": ObjectId("1a2b3c4d5e6f7a8b9c0d1e2f"),
         "content": "Całkowicie się zgadzam!",
         "createdAt": ISODate("2023-05-20T15:45:00Z"),
         "likesCount": 3
}
```

Kolekcja Follows

```
{
  "_id": ObjectId("8c9d0e1f2a3b4c5d6e7f8a9b"),
         "followerId": ObjectId("1a2b3c4d5e6f7a8b9c0d1e2f"),
         "followingId": ObjectId("5f8a7b2d9d3e7a1c3c7b4a1c"),
         "createdAt": ISODate("2023-03-10T09:15:00Z")
}
```

Kolekcja Likes

```
{
  "_id": ObjectId("9d0e1f2a3b4c5d6e7f8a9b0c"),
         "userId": ObjectId("1a2b3c4d5e6f7a8b9c0d1e2f"),
         "postId": ObjectId("6a1b2c3d4e5f6a7b8c9d0e1f"),
         "createdAt": ISODate("2023-05-20T16:20:00Z")
}
```

Zalety:

- Czysta struktura danych z jasno zdefiniowanymi relacjami
- Łatwość aktualizacji pojedynczych elementów (np. liczników)
- Mniejsze dokumenty, co ułatwia ich przetwarzanie
- Dobra skalowalność dla dużej liczby użytkowników i postów

Wady:

- Konieczność wykonywania wielu zapytań do bazy danych w celu pobrania powiązanych danych
- Większa złożoność zapytań przy pobieraniu kompletnych informacji
- Trudniejsze zarządzanie spójnością danych

Przykładowe zapytanie - pobieranie posta z informacjami o autorze i komentarzami:

```
//Pobranie posta
db.Posts.findOne({ _id: ObjectId("6a1b2c3d4e5f6a7b8c9d0e1f") })

// Pobranie autora posta
db.Users.findOne({ _id: ObjectId("5f8a7b2d9d3e7a1c3c7b4a1c") })

//Pobranie komentarzy do posta
db.Comments.find({ postId: ObjectId("6a1b2c3d4e5f6a7b8c9d0e1f") }).toArray()

//Pobranie autorów komentarzy
db.Users.find({ _id: { $in: [ObjectId("1a2b3c4d5e6f7a8b9c0d1e2f"), ObjectId("2b3c4d5e6f7a8b9c0d1e2f3a")] } }).toArray()
```

Wariant 2: Struktura z dokumentami zagnieżdżonymi

W tym podejściu wykorzystujemy zagnieżdżone dokumenty, aby zmniejszyć liczbę zapytań do bazy danych.

Kolekcja Users

```
{
  "_id": ObjectId("5f8a7b2d9d3e7a1c3c7b4a1c"),
         "username": "jan_kowalski",
         "email": "jan.kowalski@example.com",
         "password": "$2a$10$X7JhbS1Ehp8YK4Xo9jM1e.Lqz4DZM6rXV8wY9X9Y8X7X8X7X8X7X8",
         "displayName": "Jan Kowalski",
         "bio": "Pasjonat technologii i kawy",
         "joinDate": ISODate("2023-01-15T12:00:00Z"),
         "stats": {
           "followers": 120,
           "following": 85,
           "posts": 45
         }
}
```

Kolekcja Posts z zagnieżdżonymi komentarzami

```
{
  "_id": ObjectId("6a1b2c3d4e5f6a7b8c9d0e1f"),
         "author": {
    "_id": ObjectId("5f8a7b2d9d3e7a1c3c7b4a1c"),
           "username": "jan_kowalski",
           "displayName": "Jan Kowalski"
  },
  "content": "Dzisiaj jest piękny dzień na programowanie!",
         "createdAt": ISODate("2023-05-20T14:30:00Z"),
         "updatedAt": ISODate("2023-05-20T14:30:00Z"),
         "likes": 25,
         "maxComments": 20,
         "tags": ["programowanie", "technologia"],
         "comments": [
```

```
    {
      "_id": ObjectId("7b8c9d0e1f2a3b4c5d6e7f8a"),
      "author": {
        "_id": ObjectId("1a2b3c4d5e6f7a8b9c0d1e2f"),
        "username": "anna_nowak",
        "displayName": "Anna Nowak"
      },
      "content": "Całkowicie się zgadzam!",
      "createdAt": ISODate("2023-05-20T15:45:00Z"),
      "likes": 3
    }
  ]
}
```

Zalety:

- Szybsze pobieranie kompletnych danych o poście wraz z komentarzami
- Lepsza wydajność przy odczycie danych

Wady:

- Trudniejsza aktualizacja zagnieżdżonych danych
- Duplikacja danych (np. informacje o autorze są powielane)
- Problemy ze skalowalnością przy dużej liczbie komentarzy

Przykładowe zapytanie - pobieranie posta z komentarzami:

```
db.Posts.findOne({ _id: ObjectId("6a1b2c3d4e5f6a7b8c9d0e1f") })
```

Wariant 3: Struktura hybrydowa

W tym podejściu łączymy zalety obu poprzednich wariantów, wykorzystując zarówno referencje, jak i zagnieżdżone dokumenty.

Kolekcja Users

```
{
  "_id": ObjectId("5f8a7b2d9d3e7a1c3c7b4a1c"),
         "username": "jan_kowalski",
         "email": "jan.kowalski@example.com",
         "password": "$2a$10$X7JhbS1Ehp8YK4Xo9jM1e.Lqz4DZM6rXV8wY9X9Y8X7X8X7X8X7X8",
         "profile": {
           "displayName": "Jan Kowalski",
           "bio": "Pasjonat technologii i kawy",
           "joinDate": ISODate("2023-01-15T12:00:00Z")
           },
  "stats": {
    "followers": 120,
           "following": 85,
           "posts": 45
    }
}
```

Kolekcja Posts

```
{
  "_id": ObjectId("6a1b2c3d4e5f6a7b8c9d0e1f"),
         "userId": ObjectId("5f8a7b2d9d3e7a1c3c7b4a1c"),
         "authorSummary": {
           "username": "jan_kowalski",
           "displayName": "Jan Kowalski"
  },
         "content": "Dzisiaj jest piękny dzień na programowanie!",
```

```
          "createdAt": ISODate("2023-05-20T14:30:00Z"),
          "updatedAt": ISODate("2023-05-20T14:30:00Z"),
          "stats": {
            "likes": 25,
            "comments": 8
          },
          "maxComments": 20,
          "tags": ["programowanie", "technologia"],
          "recentComments": [
    {
      "_id": ObjectId("7b8c9d0e1f2a3b4c5d6e7f8a"),
      "userId": ObjectId("1a2b3c4d5e6f7a8b9c0d1e2f"),
      "authorSummary": {
        "username": "anna_nowak",
        "displayName": "Anna Nowak"
      },
      "content": "Całkowicie się zgadzam!",
      "createdAt": ISODate("2023-05-20T15:45:00Z"),
      "likes": 3
    }
  ]
}
```

Kolekcja Comments

```
{
  "_id": ObjectId("7b8c9d0e1f2a3b4c5d6e7f8a"),
          "postId": ObjectId("6a1b2c3d4e5f6a7b8c9d0e1f"),
          "userId": ObjectId("1a2b3c4d5e6f7a8b9c0d1e2f"),
          "authorSummary": {
            "username": "anna_nowak",
            "displayName": "Anna Nowak"
  },
  "content": "Całkowicie się zgadzam!",
```

```
            "createdAt": ISODate("2023-05-20T15:45:00Z"),
            "likes": 3
}
```

Zalety:

- Zrównoważone podejście łączące zalety obu poprzednich wariantów
- Szybki dostęp do najnowszych komentarzy bez dodatkowych zapytań
- Możliwość pobierania większej liczby komentarzy w razie potrzeby
- Lepsza skalowalność niż w przypadku pełnego zagnieżdżenia
- Mniejsza duplikacja danych niż w wariancie 2

Wady:

- Większa złożoność implementacji
- Nadal występuje pewna duplikacja danych

Przykładowe zapytanie - pobieranie posta z najnowszymi komentarzami:

```
db.Posts.findOne({ _id: ObjectId("6a1b2c3d4e5f6a7b8c9d0e1f") });
```

Przykładowe zapytanie - pobieranie wszystkich komentarzy do posta:

```
db.Comments
  .find({ postId: ObjectId("6a1b2c3d4e5f6a7b8c9d0e1f") })
  .sort({ createdAt: -1 })
  .toArray();
```

Wybraliśmy wariant hybrydowy. Baza składa się z trzech głównych kolekcji: Users, Posts i Comments, które są ze sobą powiązane za pomocą referencji.

Kolekcja Users

Przechowuje informacje o użytkownikach systemu:

- _id: unikalny identyfikator użytkownika
- username: nazwa użytkownika (3-30 znaków, tylko litery, cyfry i podkreślenia)
- email: adres email użytkownika
- password: zahaszowane hasło
- profile: zagnieżdżony obiekt zawierający:
    - displayName: wyświetlana nazwa użytkownika
    - bio: krótki opis użytkownika (do 160 znaków)
    - joinDate: data dołączenia do serwisu
- stats: zagnieżdżony obiekt ze statystykami:
    - followers: liczba obserwujących
    - following: liczba obserwowanych
    - posts: liczba opublikowanych postów

Kolekcja Posts

Zawiera posty użytkowników:

- _id: unikalny identyfikator posta

- userId: referencja do autora posta (powiązanie z kolekcją Users)
- authorSummary: podstawowe informacje o autorze:
- username: nazwa użytkownika
- displayName: wyświetlana nazwa
- content: treść posta (1-280 znaków)
- createdAt: data utworzenia posta
- updatedAt: data ostatniej aktualizacji
- stats: statystyki posta:
    - likes: liczba polubień
    - comments: liczba komentarzy
    - maxComments: maksymalna dozwolona liczba komentarzy (0-1000)
- tags: tablica tagów posta
- recentComments: tablica najnowszych komentarzy z podstawowymi informacjami

Kolekcja Comments

Przechowuje komentarze do postów:

- _id: unikalny identyfikator komentarza
- postId: referencja do posta (powiązanie z kolekcją Posts)

-userId: referencja do autora komentarza (powiązanie z kolekcją Users)

- authorSummary: podstawowe informacje o autorze:

    - username: nazwa użytkownika
    - displayName: wyświetlana nazwa
- content: treść komentarza (1-280 znaków)
- createdAt: data utworzenia komentarza
- likes: liczba polubień komentarza

Ponizej znajduje się walidacja oraz insert przykładowych danych, a także przykładowe polecenia.

```
db.createCollection("Users", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["username", "email", "password", "profile", "stats"],
      properties: {
        username: {
          bsonType: "string",
          minLength: 3,
          maxLength: 30,
          pattern: "^[a-zA-Z0-9_]+$",
          description:
            "Musi być ciągiem znaków o długości 3-30 zawierającym tylko litery, cyfry i podkreślenia"
        },
        email: {
          bsonType: "string",
          pattern: "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+.[a-zA-Z]{2,}$",
          description: "Musi być poprawnym adresem email"
        },
        password: {
          bsonType: "string",
          minLength: 60,
          maxLength: 60,
          description: "Musi być zahaszowanym hasłem"
```

```
        },
        profile: {
          bsonType: "object",
          required: ["displayName", "joinDate"],
          properties: {
            displayName: {
              bsonType: "string",
              minLength: 1,
              maxLength: 50
            },
            bio: {
              bsonType: "string",
              maxLength: 160
            },
            joinDate: {
              bsonType: "date"
            }
          }
        },
        stats: {
          bsonType: "object",
          required: ["followers", "following", "posts"],
          properties: {
            followers: {
              bsonType: "int",
              minimum: 0
            },
            following: {
              bsonType: "int",
              minimum: 0
            },
            posts: {
              bsonType: "int",
              minimum: 0
            }
          }
```

```
        }
      }
    }
  }
})

db.createCollection("Posts", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: [
        "userId",
        "authorSummary",
        "content",
        "createdAt",
        "stats",
        "maxComments"
      ],
      properties: {
        userId: {
          bsonType: "objectId"
        },
        authorSummary: {
          bsonType: "object",
          required: ["username", "displayName"],
          properties: {
            username: {
              bsonType: "string"
            },
            displayName: {
              bsonType: "string"
            }
          }
        },
        content: {
          bsonType: "string",
```

```
      minLength: 1,
      maxLength: 280
    },
    createdAt: {
      bsonType: "date"
    },
    updatedAt: {
      bsonType: "date"
    },
    stats: {
      bsonType: "object",
      required: ["likes", "comments"],
      properties: {
        likes: {
          bsonType: "int",
          minimum: 0
        },
        comments: {
          bsonType: "int",
          minimum: 0
        }
      }
    },
    maxComments: {
      bsonType: "int",
      minimum: 0,
      maximum: 1000
    },
    tags: {
      bsonType: "array",
      items: {
        bsonType: "string"
      }
    },
    recentComments: {
      bsonType: "array",
```

```
            items: {
              bsonType: "object",
              required: [
                "_id",
                "userId",
                "authorSummary",
                "content",
                "createdAt"
              ],
              properties: {
                _id: {
                  bsonType: "objectId"
                },
                userId: {
                  bsonType: "objectId"
                },
                authorSummary: {
                  bsonType: "object",
                  required: ["username", "displayName"],
                  properties: {
                    username: {
                      bsonType: "string"
                    },
                    displayName: {
                      bsonType: "string"
                    }
                  }
                },
                content: {
                  bsonType: "string",
                  minLength: 1,
                  maxLength: 280
                },
                createdAt: {
                  bsonType: "date"
                },
```

```
              likes: {
                bsonType: "int",
                minimum: 0
              }
            }
          }
        }
      }
    }
  }
})

db.createCollection("Comments", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["postId", "userId", "authorSummary", "content", "createdAt"],
      properties: {
        postId: {
          bsonType: "objectId"
        },
        userId: {
          bsonType: "objectId"
        },
        authorSummary: {
          bsonType: "object",
          required: ["username", "displayName"],
          properties: {
            username: {
              bsonType: "string"
            },
            displayName: {
              bsonType: "string"
            }
          }
        },
```

```
        content: {
          bsonType: "string",
          minLength: 1,
          maxLength: 280
        },
        createdAt: {
          bsonType: "date"
        },
        likes: {
          bsonType: "int",
          minimum: 0
        }
      }
    }
  }
})

db.Users.insertMany([
  {
    _id: ObjectId("5f8a7b2d9d3e7a1c3c7b4a1c"),
    username: "jan_kowalski",
    email: "jan.kowalski@example.com",
    password: "$2a$10$X7JhbS1Ehp8YK4Xo9jM1e.Lqz4DZM6rXV8wY9X9Y8X7X8X7X8X7X8",
    profile: {
      displayName: "Jan Kowalski",
      bio: "Pasjonat technologii i kawy",
      joinDate: ISODate("2023-01-15T12:00:00Z")
    },
    stats: {
      followers: 120,
      following: 85,
      posts: 45
    }
  },
  {
    _id: ObjectId("1a2b3c4d5e6f7a8b9c0d1e2f"),
```

```
    username: "anna_nowak",
    email: "anna.nowak@example.com",
    password: "$2a$10$Y8X7X8X7X8X7X8X7X8X7X.Lqz4DZM6rXV8wY9X9Y8X7X8X7X8X7X8",
    profile: {
      displayName: "Anna Nowak",
      bio: "Miłośniczka książek i podróży",
      joinDate: ISODate("2023-02-10T09:30:00Z")
    },
    stats: {
      followers: 85,
      following: 120,
      posts: 32
    }
  },
  {
    _id: ObjectId("2b3c4d5e6f7a8b9c0d1e2f3a"),
    username: "marek_wisniewski",
    email: "marek.wisniewski@example.com",
    password: "$2a$10$Z9Y8X7X8X7X8X7X8X7X8X.Lqz4DZM6rXV8wY9X9Y8X7X8X7X8X7X8",
    profile: {
      displayName: "Marek Wiśniewski",
      bio: "Programista, fan piłki nożnej",
      joinDate: ISODate("2023-03-05T15:45:00Z")
    },
    stats: {
      followers: 65,
      following: 42,
      posts: 28
    }
  }
])

db.Posts.insertMany([
  {
    _id: ObjectId("6a1b2c3d4e5f6a7b8c9d0e1f"),
    userId: ObjectId("5f8a7b2d9d3e7a1c3c7b4a1c"),
```

```
    authorSummary: {
      username: "jan_kowalski",
      displayName: "Jan Kowalski"
    },
    content: "Dzisiaj jest piękny dzień na programowanie!",
    createdAt: ISODate("2023-05-20T14:30:00Z"),
    updatedAt: ISODate("2023-05-20T14:30:00Z"),
    stats: {
      likes: 25,
      comments: 2
    },
    maxComments: 20,
    tags: ["programowanie", "technologia"],
    recentComments: [
      {
        _id: ObjectId("7b8c9d0e1f2a3b4c5d6e7f8a"),
        userId: ObjectId("1a2b3c4d5e6f7a8b9c0d1e2f"),
        authorSummary: {
          username: "anna_nowak",
          displayName: "Anna Nowak"
        },
        content: "Całkowicie się zgadzam!",
        createdAt: ISODate("2023-05-20T15:45:00Z"),
        likes: 3
      },
      {
        _id: ObjectId("8c9d0e1f2a3b4c5d6e7f8a9b"),
        userId: ObjectId("2b3c4d5e6f7a8b9c0d1e2f3a"),
        authorSummary: {
          username: "marek_wisniewski",
          displayName: "Marek Wiśniewski"
        },
        content: "Jaki projekt dziś realizujesz?",
        createdAt: ISODate("2023-05-20T16:20:00Z"),
        likes: 1
      }
```

```
    ]
  },
  {
    _id: ObjectId("7b2c3d4e5f6a7b8c9d0e1f2a"),
    userId: ObjectId("1a2b3c4d5e6f7a8b9c0d1e2f"),
    authorSummary: {
      username: "anna_nowak",
      displayName: "Anna Nowak"
    },
    content:
      "Właśnie skończyłam czytać świetną książkę! Polecam 'Duma i uprzedzenie'.",
    createdAt: ISODate("2023-05-21T10:15:00Z"),
    updatedAt: ISODate("2023-05-21T10:15:00Z"),
    stats: {
      likes: 18,
      comments: 1
    },
    maxComments: 10,
    tags: ["książki", "czytanie"],
    recentComments: [
      {
        _id: ObjectId("9d0e1f2a3b4c5d6e7f8a9b0c"),
        userId: ObjectId("5f8a7b2d9d3e7a1c3c7b4a1c"),
        authorSummary: {
          username: "jan_kowalski",
          displayName: "Jan Kowalski"
        },
        content: "Też ją czytałem, świetna pozycja!",
        createdAt: ISODate("2023-05-21T11:30:00Z"),
        likes: 2
      }
    ]
  }
])

db.Comments.insertMany([
```

```
  {
    _id: ObjectId("7b8c9d0e1f2a3b4c5d6e7f8a"),
    postId: ObjectId("6a1b2c3d4e5f6a7b8c9d0e1f"),
    userId: ObjectId("1a2b3c4d5e6f7a8b9c0d1e2f"),
    authorSummary: {
      username: "anna_nowak",
      displayName: "Anna Nowak"
    },
    content: "Całkowicie się zgadzam!",
    createdAt: ISODate("2023-05-20T15:45:00Z"),
    likes: 3
  },
  {
    _id: ObjectId("8c9d0e1f2a3b4c5d6e7f8a9b"),
    postId: ObjectId("6a1b2c3d4e5f6a7b8c9d0e1f"),
    userId: ObjectId("2b3c4d5e6f7a8b9c0d1e2f3a"),
    authorSummary: {
      username: "marek_wisniewski",
      displayName: "Marek Wiśniewski"
    },
    content: "Jaki projekt dziś realizujesz?",
    createdAt: ISODate("2023-05-20T16:20:00Z"),
    likes: 1
  },
  {
    _id: ObjectId("9d0e1f2a3b4c5d6e7f8a9b0c"),
    postId: ObjectId("7b2c3d4e5f6a7b8c9d0e1f2a"),
    userId: ObjectId("5f8a7b2d9d3e7a1c3c7b4a1c"),
    authorSummary: {
      username: "jan_kowalski",
      displayName: "Jan Kowalski"
    },
    content: "Też ją czytałem, świetna pozycja!",
    createdAt: ISODate("2023-05-21T11:30:00Z"),
    likes: 2
```

```
  }
])
```

Przykładowe polecenia:

```
// Wyszukiwanie postów po tagach
db.Posts.find({ tags: "programowanie" })
  .sort({ createdAt: -1 })
  .limit(20)
  .toArray()

//wynik
        [
        {
          "_id": {"$oid": "6a1b2c3d4e5f6a7b8c9d0e1f"},
          "authorSummary": {
            "username": "jan_kowalski",
            "displayName": "Jan Kowalski"
          },
          "content": "Dzisiaj jest piękny dzień na programowanie!",
          "createdAt": {"$date": "2023-05-20T14:30:00.000Z"},
          "maxComments": 20,
          "recentComments": [
            {
              "_id": {"$oid": "7b8c9d0e1f2a3b4c5d6e7f8a"},
              "userId": {"$oid": "1a2b3c4d5e6f7a8b9c0d1e2f"},
              "authorSummary": {
                "username": "anna_nowak",
                "displayName": "Anna Nowak"
              },
              "content": "Całkowicie się zgadzam!",
              "createdAt": {"$date": "2023-05-20T15:45:00.000Z"},
              "likes": 3
```

```
            },
            {
              "_id": {"$oid": "8c9d0e1f2a3b4c5d6e7f8a9b"},
              "userId": {"$oid": "2b3c4d5e6f7a8b9c0d1e2f3a"},
              "authorSummary": {
                "username": "marek_wisniewski",
                "displayName": "Marek Wiśniewski"
              },
              "content": "Jaki projekt dziś realizujesz?",
              "createdAt": {"$date": "2023-05-20T16:20:00.000Z"},
              "likes": 1
            }
          ],
          "stats": {
            "likes": 25,
            "comments": 2
          },
          "tags": ["programowanie", "technologia"],
          "updatedAt": {"$date": "2023-05-20T14:30:00.000Z"},
          "userId": {"$oid": "5f8a7b2d9d3e7a1c3c7b4a1c"}
        }
        ]

// Pobieranie wszystkich komentarzy do posta
db.Comments
        .find({ postId: ObjectId("6a1b2c3d4e5f6a7b8c9d0e1f") })
        .sort({ createdAt: -1 })
        .toArray();

//wynik

[
  {
    "_id": {"$oid": "8c9d0e1f2a3b4c5d6e7f8a9b"},
    "authorSummary": {
      "username": "marek_wisniewski",
```

```
      "displayName": "Marek Wiśniewski"
    },
    "content": "Jaki projekt dziś realizujesz?",
    "createdAt": {"$date": "2023-05-20T16:20:00.000Z"},
    "likes": 1,
    "postId": {"$oid": "6a1b2c3d4e5f6a7b8c9d0e1f"},
    "userId": {"$oid": "2b3c4d5e6f7a8b9c0d1e2f3a"}
  },
  {
    "_id": {"$oid": "7b8c9d0e1f2a3b4c5d6e7f8a"},
    "authorSummary": {
      "username": "anna_nowak",
      "displayName": "Anna Nowak"
    },
    "content": "Całkowicie się zgadzam!",
    "createdAt": {"$date": "2023-05-20T15:45:00.000Z"},
    "likes": 3,
    "postId": {"$oid": "6a1b2c3d4e5f6a7b8c9d0e1f"},
    "userId": {"$oid": "1a2b3c4d5e6f7a8b9c0d1e2f"}
  }
]

// Aktualizacja licznika komentarzy w poście

db.Posts.updateOne(
        { _id: ObjectId("6a1b2c3d4e5f6a7b8c9d0e1f") },
        {
          $inc: { "stats.comments": 1 },
          $push: {
            recentComments: {
              $each: [
                {
                  _id: new ObjectId(),
                  userId: ObjectId("1a2b3c4d5e6f7a8b9c0d1e2f"),
                  authorSummary: {
                    username: "anna_nowak",
```

```
                    displayName: "Anna Nowak"
                  },
                  content: "Nowy komentarz",
                  createdAt: new Date(),
                  likes: 0
                }
              ],
              $sort: { createdAt: -1 },
              $slice: 5
            }
          }
        }
)

// Aktualizacja profilu użytkownika

db.Users.updateOne(
        { _id: ObjectId("5f8a7b2d9d3e7a1c3c7b4a1c") },
        {
          $set: {
            "profile.displayName": "Jan K. Kowalski"
          }
        }
)
```

Punktacja:

zadanie	pkt
1	1
2	1

razem	2