Dokumentowe bazy danych – MongoDB

Ćwiczenie/zadanie

Imiona i nazwiska autorów: Damian Torbus, Adam Woźny

Odtwórz z backupu bazę `north0`

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

```
use north0
```

Baza `north0` jest kopią relacyjnej bazy danych `Northwind`

- poszczególne kolekcje odpowiadają tabelom w oryginalnej bazie `Northwind`

Wprowadzenie

zapoznaj się ze strukturą dokumentów w bazie `North0`

```
db.customers.find()
db.orders.find();
db.orderdetails.find();
```

Operacje wyszukiwania danych, przetwarzanie dokumentów

Zadanie 1

stwórz kolekcję `OrdersInfo` zawierającą następujące dane o zamówieniach

- kolekcję `OrdersInfo` należy stworzyć przekształcając dokumenty w oryginalnych kolekcjach `customers, orders, orderdetails, employees, shippers, products, categories, suppliers` do kolekcji w której 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": ...
    }
  }
]
```

Polecenie::

```
use('b')
db.orders.aggregate([
  { // Customer lookup + unwind, żeby był pojedyńczym dokumentem, a nie listą
    $lookup: {
      from: "customers",
      localField: "CustomerID",
      foreignField: "CustomerID",
      as: "Customer"
    }
  },
  {
    $unwind: "$Customer"
  },
  { // Employees lookup + unwind, żeby był pojedyńczym dokumentem, a nie listą
    $lookup: {
      from: "employees",
      localField: "EmployeeID",
      foreignField: "EmployeeID",
      as: "Employee"
    }
  },
  {
    $unwind: "$Employee"
  },
  { // Shippers lookup + unwind, żeby był pojedyńczym dokumentem, a nie listą
    $lookup: {
      from: "shippers",
      localField: "ShipVia",
      foreignField: "ShipperID",
      as: "Shipper"
    }
  },
  {
    $unwind: "$Shipper"
  },
  { // Orderdetails lookup + unwind, żeby był pojedyńczym dokumentem, a nie listą,
ponieważ chcemy uniknąć mapowania po liście
    $lookup: {
      from: "orderdetails",
      localField: "OrderID",
      foreignField: "OrderID",
      as: "Orderdetails"
```

```
    }
  },
  {
    $unwind: "$Orderdetails"
  },
  { // pobieramy wartości product i category, ale nie dajemy ich bezpośrednio
tylko do zagnieżdżonych dokumentów
    $lookup: {
      from: "products",
      localField: "Orderdetails.ProductID",
      foreignField: "ProductID",
      as: "Product"
    }
  },
  {
    $unwind: "$Product"
  },
  { // tu tak samo
    $lookup: {
      from: "categories",
      localField: "Product.CategoryID",
      foreignField: "CategoryID",
      as: "Category"
    }
  },
  {
    $unwind: "$Category"
  },
  { // proces dokładania wartości do istniejących dokumentów zagnieżdzonych
    $addFields: {
      "Orderdetails.Value": {
        $multiply: [
          "$Orderdetails.UnitPrice",
          "$Orderdetails.Quantity",
          { $subtract: [1, "$Orderdetails.Discount"] }
        ]
      },
      "Orderdetails.product": {
        ProductID: "$Product.ProductID",
        ProductName: "$Product.ProductName",
        QuantityPerUnit: "$Product.QuantityPerUnit",
        CategoryID: "$Category.CategoryID",
        CategoryName: "$Category.CategoryName"
      }
    }
  },
  { // grupujemy, OrderID, Employee, Dates, Freight, Shipment zachowujemy bo są
unikalne, a Orderdetails kumulujemy, tak jak przed operacją unwind
    $group: {
      _id: "$OrderID",
      OrderID: { $first: "$OrderID" },
      Customer: { $first: {
        CustomerID: "$Customer.CustomerID",
        CompanyName: "$Customer.CompanyName",
```

```
        City: "$Customer.City",
        Country: "$Customer.Country"
      }},
      Employee: { $first: {
        EmployeeID: "$Employee.EmployeeID",
        FirstName: "$Employee.FirstName",
        LastName: "$Employee.LastName",
        Title: "$Employee.Title"
      }},
      Dates: { $first: {
        OrderDate: "$OrderDate",
        RequiredDate: "$RequiredDate"
      }},
      Freight: { $first: "$Freight" },
      Shipment: { $first: {
        Shipper: {
          ShipperID: "$Shipper.ShipperID",
          CompanyName: "$Shipper.CompanyName"
        },
        ShipName: "$ShipName",
        ShipAddress: "$ShipAddress",
        ShipCity: "$ShipCity",
        ShipCountry: "$ShipCountry"
      }},
      Orderdetails: { $push: {
        UnitPrice: "$Orderdetails.UnitPrice",
        Quantity: "$Orderdetails.Quantity",
        Discount: "$Orderdetails.Discount",
        Value: "$Orderdetails.Value",
        product: "$Orderdetails.product"
      }}
    }
  },
  {
    $addFields: {
      OrderTotal: {
        $sum: "$Orderdetails.Value"
      }
    }
  },
  {
    $out: "OrdersInfo"
  }
])
```

Zapytanie zostało wykonane bardzo schematycznie i można je podzielić na etapy:

1. Za pomocą operacji `$lookup` oraz `$unwind` są dołączane informacje z kolekcji innych niż orders. w przypadkach innych niż ordeDetails mamy pewność, że relacja jest 1-1, więc operacja `$unwind` polega tylko na przekształceniu jednodokumentowej listy na po prostu jeden dokument, w przypadku orderDetails, to relacja jest 1-n, więc dla każdego takie połączenia powstaje nowy dokument.

2. Następnie, kiedy każdy z dokumentów ma tylko jeden zagnieżdżony dokument orderDetails jest wykonywana operacja `$addFields` i dodawane są pola o produkcie oraz pole values.
3. Grupowanie dokumentów, tak żeby dla każdego zamówienie był tylko jeden unikalny dokument, została zastosowana operacja `$first`, tam gdzie wartość dla każdego zamówienia była taka sama oraz operacja `$push`, żeby zagregować pola związane z orderdetails (prodkt, wartość etc) w jedną listę dla każdego dokumentu.
4. Zsumowanie pola `Value`, żeby dostać wartość całego zamówienia.
5. Zapisanie do nowej kolekcji `OrdersInfo`

Napisanie zapytania przy dobrej znajomości SQLa było dosyć intuicyjne (pomijając konkretną składnie mongodb, ona jest całkowicie różna), operacje `$lookup`, działaja identycznie jak `join`, a logika używania `$addFields` również jest prosta. Największą różnica było zastosowanie tak dużej ilości operacji `$first` w `$group` zamiast grupowania po tych wszystkich atrybutach, ale w tym wypadku zawsze jest założenie, że wszystko po czym sie grupuje trafia do `id`, co może być uciążliwe i z tego co udało nam się wyczytać nie jest rekomendowane, a jako, że w obrębie grupy wartość tych atrybótów jest taka sama to jakakolwiek operacja wyboru jednej z nich da dobry rezultat (równie dobrze mogłoby to być `$last`).

Zadanie 2

stwórz kolekcję `CustomerInfo` zawierającą następujące dane o każdym kliencie

- 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)

    ]


]
```

Polecenie:

```
db.OrdersInfo.aggregate([
  { // korzystamy z poprzednio stworzonej kolekcji, wszystkie infromacje poza tą o
kliencie są agregowane do listy dokumentów
    $group: {
```

```
        _id: "$Customer.CustomerID",
        CustomerID: { $first: "$Customer.CustomerID" },
        CompanyName: { $first: "$Customer.CompanyName" },
        City: { $first: "$Customer.City" },
        Country: { $first: "$Customer.Country" },
        Orders: {
          $push: {
            _id: "$_id",
            OrderID: "$OrderID",
            Employee: "$Employee",
            Dates: "$Dates",
            Orderdetails: "$Orderdetails",
            Freight: "$Freight",
            OrderTotal: "$OrderTotal",
            Shipment: "$Shipment"
          }
        }
      }
    },
    { // wybieramy tylko interesujące nas atrybuty
      $project: {
        _id: 1,
        CustomerID: 1,
        CompanyName: 1,
        City: 1,
        Country: 1,
        Orders: 1
      }
    },
    {
      $out: "CustomerInfo"
    }
  ])
```

W tym przypadku dzięki wykorzystaniu uprzednio stworzonej struktury zapytanie zarówno składniowo, jak i logicznie jest bardzo proste (jeśli w zadaniu trzeba użyć tylko oryginalnych kolekcji to jako zapytanie proszę potrktować poprzednie zapytanie z dołożoną tą cześcia, dalej powinna to być najproszta wersja). Zadaniem polecenia była dołożenie kolejnego poziomu zagnieżdzenia/agregacji dla klienta, w poprzednim zapytaniu zostały zagnieżdżone infromacje z orderDetails dla kolekcji `orders`, teraz tworzony jest kolejny poziom zagnieżdżenia uprzednion stworzonych informacji dla kolekcji `customer`. Zapytanie ma tylko trzy etapy:

1. Grupowania dokumentów z `OrdersInfo` dla klienta, z agregacją zamówienień (a w zasadzie tylko części informacji)
2. Wybieranie interesujących atrybótów
3. Zapis do `CustomerInfo`

Zadanie 3

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
    - zamieść odpowiedni komentarz
        - które wersje zapytań były "prostsze"

```
[
  {
    "_id":

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

  }
]
```

Zaptytania

Na bazie istniejących kolekcji

```
db.orders.aggregate([
  // filtr na dacie
  {
    $match: {
      OrderDate: {
        $gte: ISODate("1997-01-01T00:00:00Z"),
        $lt:  ISODate("1998-01-01T00:00:00Z")
      }
    }
  },

  // dołączenia informacji z innych kolekcji
  {
    $lookup: {
      from: "orderdetails",
      localField: "OrderID",
      foreignField: "OrderID",
      as: "Details"
    }
  },
  { $unwind: "$Details" },

  {
```

```
    $lookup: {
      from: "products",
      localField: "Details.ProductID",
      foreignField: "ProductID",
      as: "Prod"
    }
  },
  { $unwind: "$Prod" },

  {
    $lookup: {
      from: "categories",
      localField: "Prod.CategoryID",
      foreignField: "CategoryID",
      as: "Cat"
    }
  },
  { $unwind: "$Cat" },

  // filtr na kategorii
  {
    $match: {
      "Cat.CategoryName": "Confections"
    }
  },

  // dodawanie pola value
  {
    $addFields: {
      lineValue: {
        $round: [
          {
            $multiply: [
              "$Details.UnitPrice",
              "$Details.Quantity",
              { $subtract: [1, "$Details.Discount"] }
            ]
          },
          2
        ]
      }
    }
  },

  // grupowanie po kliencie i suma wartoścu
  {
    $group: {
      _id: "$CustomerID",
      ConfectionsSale97: { $sum: "$lineValue" }
    }
  },

  // dołaczenie informacji o kliencie
  {
```

```
    $lookup: {
      from: "customers",
      localField: "_id",
      foreignField: "CustomerID",
      as: "Cust"
    }
  },
  { $unwind: "$Cust" },


  {
    $project: {
      _id: 1,
      CustomerID: "$_id",
      CompanyName: "$Cust.CompanyName",
      ConfectionsSale97: 1
    }
  }
]);
```

Na bazie kolekcji OrderInfo

```
db.OrdersInfo.aggregate([
  // filtr na dacie
  {
    $match: {
      "Dates.OrderDate": {
        $gte: ISODate("1997-01-01T00:00:00Z"),
        $lt:  ISODate("1998-01-01T00:00:00Z")
      }
    }
  },

  // rozwijanie zamówienia, żeby móc wyfiltrować
  {
    $unwind: "$Orderdetails"
  },

  // filtr na kategorii
  {
    $match: {
      "Orderdetails.product.CategoryName": "Confections"
    }
  },

  // grupowanie po kliencie
  {
    $group: {
      _id: "$Customer.CustomerID",
      CompanyName: { $first: "$Customer.CompanyName" },
      ConfectionsSale97: { $sum: "$Orderdetails.Value" }
```

```
    }
  },
  {
    $project: {
      _id: 1,
      CustomerID: "$_id",
      CompanyName: 1,
      ConfectionsSale97: 1
    }
  }
]);
```

Na bazie kolekcji CustomerInfo

```
db.CustomerInfo.aggregate([
  // rozwijanie zamówienia, żeby móc wyfiltrować
  { $unwind: "$Orders" },

  // filtr na dacie
  {
    $match: {
      "Orders.Dates.OrderDate": {
        $gte: ISODate("1997-01-01T00:00:00Z"),
        $lt:  ISODate("1998-01-01T00:00:00Z")
      }
    }
  },

  // rozwijanie zamówienia, żeby móc wyfiltrować
  { $unwind: "$Orders.Orderdetails" },

  // filtr na kategorii
  {
    $match: {
      "Orders.Orderdetails.product.CategoryName": "Confections"
    }
  },

  // grupowanie po kliencie
  {
    $group: {
      _id: "$CustomerID",
      CompanyName: { $first: "$CompanyName" },
      ConfectionsSale97: { $sum: "$Orders.Orderdetails.Value" }
    }
  },

  {
    $project: {
      _id: 1,
      CustomerID: "$_id",
```

```
      CompanyName: 1,
      ConfectionsSale97: 1
    }
  }
]);
```

Komentarz

Przykładowy wynik

```
{
    "_id": "DUMON",
    "CompanyName": "Du monde entier",
    "ConfectionsSale97": 60,
    "CustomerID": "DUMON"
  },
  {
    "_id": "EASTC",
    "CompanyName": "Eastern Connection",
    "ConfectionsSale97": 480,
    "CustomerID": "EASTC"
  },
```

Budowa zapytań

Za pomocą oryginalnych kolekcji:

bazując na kolekcji `orders` zapytanie schematycznie posiadało kilka kroków

1. przefiltrowanie po dacie
2. dołączenie kolekcji z innych tabel, tak żeby można było dostać informacje o kliencie oraz o kategorii zapytania
3. przefiltrowanie po kategorii
4. wyliczenie wartości dodatkowych pól
5. agregacji wraz z sumą po kliencie

Same zapytanie nie było zbyt długie ani trudne do napisanie, ale jednak z powodu dość sporej ilości operacji `$lookup` nie jest zbyt czytelne na pierwszy rzut oka, nie widać bezpośredniego połączenie miedzy dokumentami. Również z tego samego powodu zapytanie nie jest efektywne

Za pomocą kolekcji `OrdersInfo`

kroki zapytania:

1. przefiltrowanie po dacie
2. przefiltrowanie po kategorii
3. wyliczenie wartości dodatkowych pól
4. agregacji wraz z sumą po kliencie

W tym wypadku korzystamy z faktu, że w kolekcji `OrdersInfo` infromacje z oryginalnych kolekcji zostały już z sobą połączone, przez co możliwe jest filtrowanie po dacie oraz kategorii bez łączenia z innymi kolekcjami. Również informacje o kliencie się tam już znajdują. Ze względu na kompletny brak operacji `$lookup` bezpośrednia zależność pomiędzy konkretnymi dokumentami jest bardzo wyraźna (np. kategoria jest pod kolekcją zagnieżdżoną `Orderdetails.product.CategoryName`, w poprzednim wypadku trzeba było wykonać operacje `$lookup` z orders -> ordersDetails -> product). Z powodu mniejszej liczby operacji to zapytanie jest bardziej efektywne niż poprzednie

Za pomocą kolekcji `CustomerInfo`

kroki zapytania:

1. przefiltrowanie po dacie
2. rozłożenie listy zamówień na poszcególne dokumenty
3. przefiltrowanie po kategorii
4. wyliczenie wartości dodatkowych pól
5. agregacji wraz z sumą po kliencie (krok odwrotny do 2)

Zapytanie bardzo podobne do porzedniego, tylko, że w tym wypadku, żeby zastosować jakikolwiek filtr w odniesieniu do zamówienie (w tym przypadku po jego kategorii) trzeba było rozłożyć zagregowaną liste zamówień na pojedyńcze dokumenty (po tej operacji struktura była bardzo podobna do tej z `OrdersInfo`) oraz wykonać takie same kroki jak w poprzednim zapytaniu.

Porównianie

Wszystkie zapytania zwróciły dokładnie taki sam wynik, taki był cel tego zadania. W zapytaniu korzystających z oryginalnych kolekcji trzeba było urzyć sporej liczby operacji `$lookup`, ponieważ w żadnej z nich nie ma wszystkich potrzebnych informacji. Następnie operacje filtrowania oraz grupowania z sumą są nie do uniknięcia w przypadku każdego z tych zapytań. Co było zaskoczeniem zapytanie korzystające z `CustomerInfo` było bardziej skomplikowane, niż to korzystające z `OrdersInfo`. Myśleliśmy, że tak będzie, poniważ poziom agregacji w wyniku oraz w kolekcji jest taki sam (klient), ale ze względu na filtrowania po zagnieżdżonej liście zamówień niezbędny było rozwinięcie tej listy do osobnych dokumentów, co sprowadziło tą kolekcje praktycznie do `OrdersInfo`, dalej kroki obydwu zapytań było takie same i proste - bezpośrenie filtrowanie i nieuniknione grupowanie z sumą, tak więc różniły się tylko krokiem rozwinięcie listy.

Zadanie 4

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
    - zamieść odpowiedni komentarz
        - które wersje zapytań były "prostsze"

```
[
  {
```

```
    "_id":

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

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

Polecenia

Na bazie istniejących kolekcji

```
db.orders.aggregate([
  // dołączenie infromacji o zamówieniach
  {
    $lookup: {
      from: "orderdetails",
      localField: "OrderID",
      foreignField: "OrderID",
      as: "Details"
    }
  },
  { $unwind: "$Details" },

  // obliczanie wartości zamówienia
  {
    $addFields: {
      lineValue: {
        $round: [
          {
            $multiply: [
              "$Details.UnitPrice",
              "$Details.Quantity",
              { $subtract: [1, "$Details.Discount"] }
            ]
          },
          2
        ]
      }
    }
  },

  // wybór potrzebnych trybótów
```

```
    {
      $project: {
        CustomerID: 1,
        OrderDate: 1,
        lineValue: 1,
        Year: { $year: "$OrderDate" },
        Month: { $month: "$OrderDate" }
      }
    },

    // Suma wartości dla klienta-roku-miesiąc - tu niestety nie dało się użyć $first
    {
      $group: {
        _id: {
          CustomerID: "$CustomerID",
          Year: "$Year",
          Month: "$Month"
        },
        TotalSales: { $sum: "$lineValue" }
      }
    },

    // dołączenie informacji o kliencie, potrzebne do company name
    {
      $lookup: {
        from: "customers",
        localField: "_id.CustomerID",
        foreignField: "CustomerID",
        as: "Cust"
      }
    },
    { $unwind: "$Cust" },

    // wybór i przerzutowanie pól z $_id do pól z wartościami
    {
      $project: {
        CustomerID: "$_id.CustomerID",
        CompanyName: "$Cust.CompanyName",
        Year: "$_id.Year",
        Month: "$_id.Month",
        TotalSales: 1
      }
    },

    // grupowanie wyników po kliencie
    {
      $group: {
        _id: "$CustomerID",
        CompanyName: { $first: "$CompanyName" },
        Sale: {
          $push: {
            Year: "$Year",
            Month: "$Month",
            TotalSales: "$TotalSales"
```

```
        }
      }
    }
  },

  {
    $project: {
      _id: 1,
      CustomerID: "$_id",
      CompanyName: 1,
      Sale: 1
    }
  }
]);
```

Na bazie kolekcji OrdersInfo

```
db.OrdersInfo.aggregate([
  // rozwijanie dokumentów, ponieważ będzie następowało grupowanie po ich
atrybutach (miesiac, rok)
  { $unwind: "$Orderdetails" },

  // wybór potrzebnych atrybutów
  {
    $project: {
      CustomerID: "$Customer.CustomerID",
      CompanyName: "$Customer.CompanyName",
      Value: "$Orderdetails.Value",
      Year: { $year: "$Dates.OrderDate" },
      Month: { $month: "$Dates.OrderDate" }
    }
  },

  // Suma wartości dla klienta-roku-miesiąc - tu niestety nie dało się użyć $first
  {
    $group: {
      _id: {
        CustomerID: "$CustomerID",
        Year: "$Year",
        Month: "$Month"
      },
      CompanyName: { $first: "$CompanyName" },
      TotalSales: { $sum: "$Value" }
    }
  },

  // wybór i przerzutowanie pól z $_id do pól z wartościami
  {
    $project: {
      CustomerID: "$_id.CustomerID",
      Year: "$_id.Year",
```

```
      Month: "$_id.Month",
      TotalSales: 1,
      CompanyName: 1
    }
  },

  // grupowanie po kliencie
  {
    $group: {
      _id: "$CustomerID",
      CompanyName: { $first: "$CompanyName" },
      Sale: {
        $push: {
          Year: "$Year",
          Month: "$Month",
          TotalSales: "$TotalSales"
        }
      }
    }
  },

  {
    $project: {
      _id: 1,
      CustomerID: "$_id",
      CompanyName: 1,
      Sale: 1,
    }
  }
]);
```

Na bazie kolekcji CustomerInfo

```
db.CustomerInfo.aggregate([
  // rozwinięcie, listy dokumentów orders, ponieważ chcemy grupować bo ich
atrybutach
  { $unwind: "$Orders" },

  // wybór i oznaczenie potrzebnych atrybutów
  {
    $project: {
      CustomerID: "$CustomerID",
      CompanyName: "$CompanyName",
      OrderValue: "$Orders.OrderTotal",
      Year: { $year: "$Orders.Dates.OrderDate" },
      Month: { $month: "$Orders.Dates.OrderDate" }
    }
  },

  // Suma wartości dla klienta-roku-miesiąc - tu niestety nie dało się użyć $first
  {
```

```
      $group: {
        _id: {
          CustomerID: "$CustomerID",
          Year: "$Year",
          Month: "$Month"
        },
        TotalSales: { $sum: "$OrderValue" },
        CompanyName: { $first: "$CompanyName"}
      }
    },

    // wybór i przerzutowanie pól z $_id do pól z wartościami
    {
      $project: {
        CustomerID: "$_id.CustomerID",
        CompanyName: 1,
        Year: "$_id.Year",
        Month: "$_id.Month",
        TotalSales: 1
      }
    },

    // grupowanie wyników po kliencie
    {
      $group: {
        _id: "$CustomerID",
        CompanyName: { $first: "$CompanyName" },
        Sale: {
          $push: {
            Year: "$Year",
            Month: "$Month",
            TotalSales: "$TotalSales"
          }
        }
      }
    },

    {
      $project: {
        _id: 1,
        CustomerID: "$_id",
        CompanyName: 1,
        Sale: 1
      }
    }
  ]);
```

Komentarz

Przykładowy wynik:

```
{
    "_id": "OCEAN",
    "CompanyName": "Océano Atlántico Ltda.",
    "Sale": [
      {
        "Year": 1998,
        "Month": 3,
        "TotalSales": 3001
      },
      {
        "Year": 1997,
        "Month": 1,
        "TotalSales": 319.20000000000005
      },
      {
        "Year": 1997,
        "Month": 5,
        "TotalSales": 110
      },
      {
        "Year": 1998,
        "Month": 2,
        "TotalSales": 30
      }
    ],
    "CustomerID": "OCEAN"
  }
```


Budowa zapytań

Z użyciem oryginalnych kolekcji

kroki zapytania:

- dołączenie kolekcji `ordersDetails`, żeby dostać informacje potrzebne do wyliczenia kosztu zamówienia
- wyliczenie kosztu
- agregacja z sumą po kliencie, miesiącu, roku
- dołożenie informcji o kliencie (innej niż `_id`)
- rozkpakowanie pól po których grupowaliśmy
- grupowanie wyłącznie po kliencie

Jako, że w kolekcji `orders` nie ma wszystkich potrzebnych informacji, trzeba było użyć operacji `$lookup`, żeby móc je potem wyświetlić, co znowu sprawia, że bardzo ciężko jest zauważyć bezpośrednie zależności pomiedzy atrybutami i sprawia, że zapytanie jest w ogólny sposób nieczytelne i niefektywne

Z uzyciem kolekcji `OrdersInfo`

kroki zapytania

- rozwinięcie listy `Orderdetails` na osobne dokumenty
- agregacja z sumą po kliencie, miesiącu, roku
- dołożenie informcji o kliencie (innej niż `_id`)
- rozkpakowanie pól po których grupowaliśmy
- grupowanie wyłącznie po kliencie

Zapytanie było bardzo podobne do poprzedniego, tylko z racji, że wszystkie informacje zawierały się w jednej kolekcji można było uniknąć operacji `$lookup`, czyniąc zapytanie czytelniejsze i efektywniejsze.

Z uzyciem kolekcji `CustomerInfo`

kroki zapytania

- rozwinięcie listy `Order` na osobne dokumenty
- agregacja z sumą po kliencie, miesiącu, roku
- dołożenie informcji o kliencie (innej niż `_id`)
- rozkpakowanie pól po których grupowaliśmy
- grupowanie wyłącznie po kliencie

Zapytanie niemal indentyczne, jak poprzednie z tą różnicą, że w kolekcji było więcej przydatnych informacji, co pozwoliło jeszcze bardziej uprościć formułę i logikę.

Porównianie

Wszystkie zapytania zwróciły ten sam, poprawny wynik — co było założonym celem. Jednak sposób ich konstrukcji znacząco się różnił pod względem złożoności oraz czytelności. W przypadku zapytania opartego na oryginalnych kolekcjach, konieczne było użycie wielu operacji `$lookup`, ponieważ żadna z kolekcji samodzielnie nie zawierała wszystkich potrzebnych informacji. To nie tylko wydłużyło pipeline, ale też znacznie obniżyło jego przejrzystość oraz efektywność. Z kolei zapytanie z użyciem kolekcji OrdersInfo było znacznie prostsze — wszystkie wymagane dane były dostępne w jednym miejscu, co pozwoliło uniknąć kosztownych dołączeń i sprawiło, że zapytanie było bardziej intuicyjne. Najbardziej przejrzyste okazało się jednak zapytanie z wykorzystaniem CustomerInfo. Chociaż początkowo wydawało się, że zagnieżdżenie zamówień w tablicy może wymusić dodatkowe operacje (np. `$unwind`), to finalnie pipeline był równie prosty jak w przypadku OrdersInfo, a nawet bardziej czytelny — dzięki logicznemu pogrupowaniu informacji na poziomie klienta i ich bezpośredniej dostępności. Podsumowując — im więcej informacji wstępnie przekształcimy i zagnieżdżimy w dokumentach (czyli im bliżej końcowej struktury), tym prostsze, krótsze i bardziej efektywne stają się zapytania.

```
---

Punktacja:

|         |     |
| ------- | --- |
| zadanie | pkt |
| 1       | 3   |
| 2       | 3   |
| 3       | 3   |
```

```
| 4       | 3   |
| razem   | 12   |
```