DAY 2 – 6 PRACTICE

Emerging Data Modeling and Management Technology: Key-Value and Document Databases

DS&AI : PROFESSIONAL TRAINING COURSE

1 – 3 APRIL 2021 | ASIAN INSTITUTE OF TECHNOLOGY

CHUTIPORN ANUTARIYA | CHUTIPORN AT AIT DOT AC DOT TH

Co-funded by the Erasmus+ Programme of the European Union

DS&AI

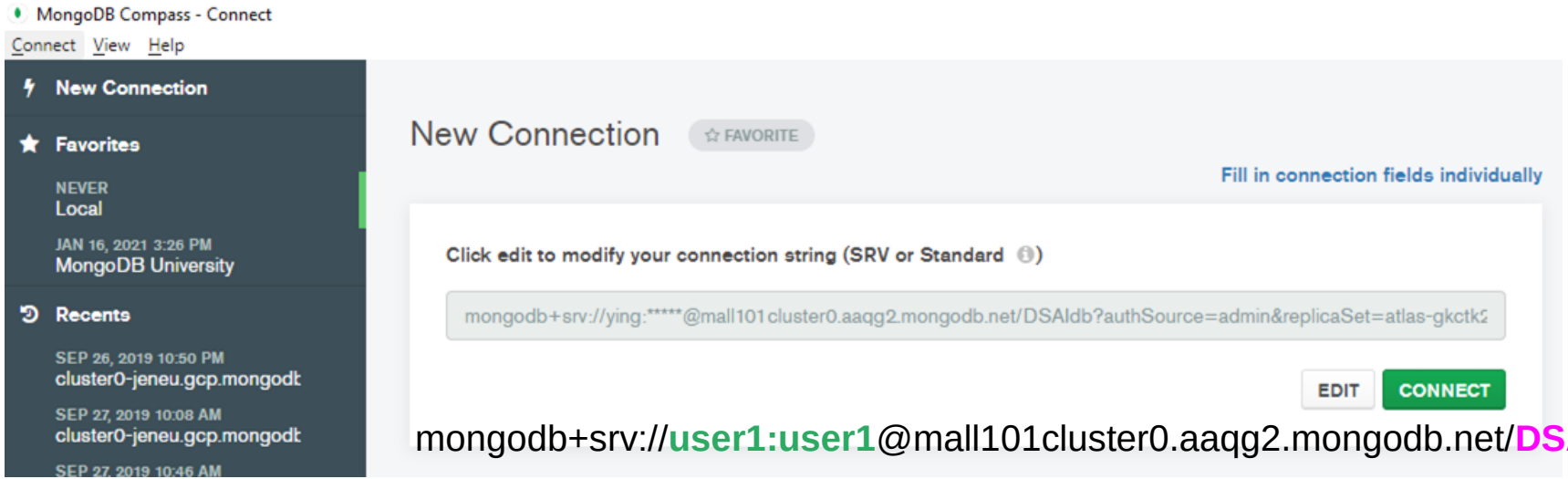
Project Practice: Modeling and Managing

Step 0 : Connect to MongoDB Atlas

- Install MongoDB Compass

  https://downloads.mongodb.com/compass/mongodb-compass-1.25.0-win32-x64.exe

- Connect MongoDb Compass to Atlas Database

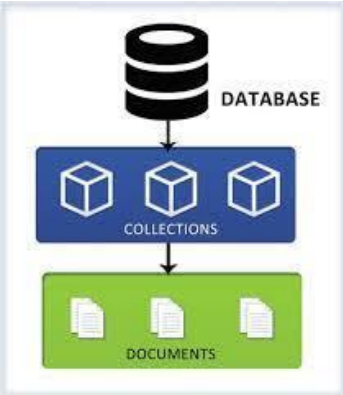


mongodb+srv://**user1:user1**@mall101cluster0.aaqg2.mongodb.net/**DSAIdb**

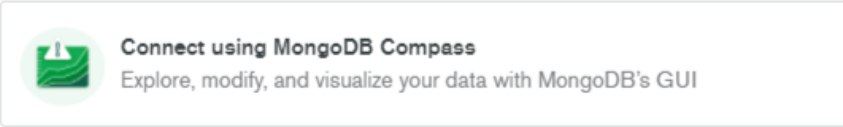
MongoDB Practice Architecture

MongoDB Atlas Cloud Database

- Cluster0 (DBServer)
  - **DSAIdb**
    - sales
    - customers
    - products



MongoDB Clients



mongodb+srv://user1:password1@mall101cluster0.aaqg2.mongodb.net/**DSAIdb**

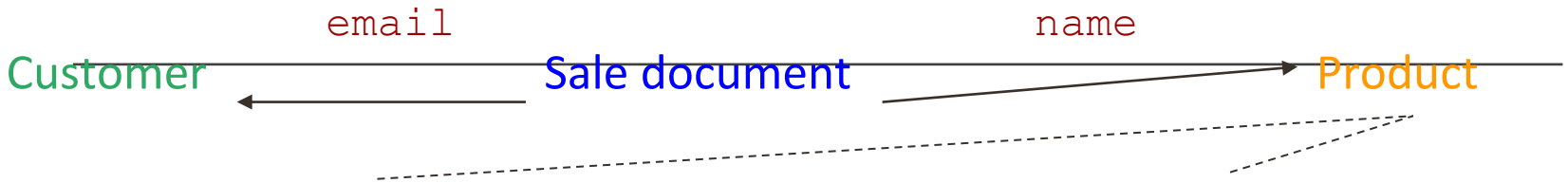
Use Case: U2

Customer searches products

Core Concepts

- Basic Query
- Query on Complex fields
  E.g., nested fields or array fields

Mongodb data model




```
_id: ObjectId("6039cb812014074d58e9d920")
name: "backpack"
description: "A drawstring style crafted fr
tags: Array
    0: "school"
    1: "travel"
    2: "kids"
price: 127.59
quantity: 200
size: Object
    v: "20"
    uom: "l"
```

```
_id: ObjectId("603b4e6d2014074d58e9d924")
name: "pencil2B"
quantity: 25
price: 200
tags: Array
    0: "pencil"
    1: "school"
size: Object
    h: 14
    w: 1
    uom: "cm"
darkness: "2B"
```

Basic Query

SQL statements	MongoDB statements
SELECT * FROM Product	db.product.find()

db.collection.find()

`db.collection.find(query, projection)`

- **query**: document
  - Optional. Specifies selection filter using query operators. To return all documents in a collection, omit this parameter or pass an empty document ({}).
- **projection**: document
  - Optional. Specifies the fields to return in the documents that match the query filter. To return all fields in the matching documents, omit this parameter.

https://docs.mongodb.com/manual/reference/method/db.collection.find/#db.collection.find

View all products

```
> db.products.find({})
< { _id: ObjectId("6039cb812014074d58e9d920"),
    name: 'backpack',
    description: 'A drawstring style crafted fr
    tags: [ 'school', 'travel', 'kids' ],
    price: NumberDecimal("127.59"),
    quantity: 200,
    size: { v: '20', uom: 'l' } }
  { _id: ObjectId("603b4e6d2014074d58e9d923"),
    name: 'canvas',
    quantity: 100,
    price: 500,
    tags: [ 'cotton' ],
```

DSAIdb.products

DOCUMENTS 5 | TOTAL SIZE 718B | AVG. SIZE 144B | INDEXES 1

Documents | Aggregations | Schema | Explain Plan | Indexes

FILTER {} | OPTIONS | FIND

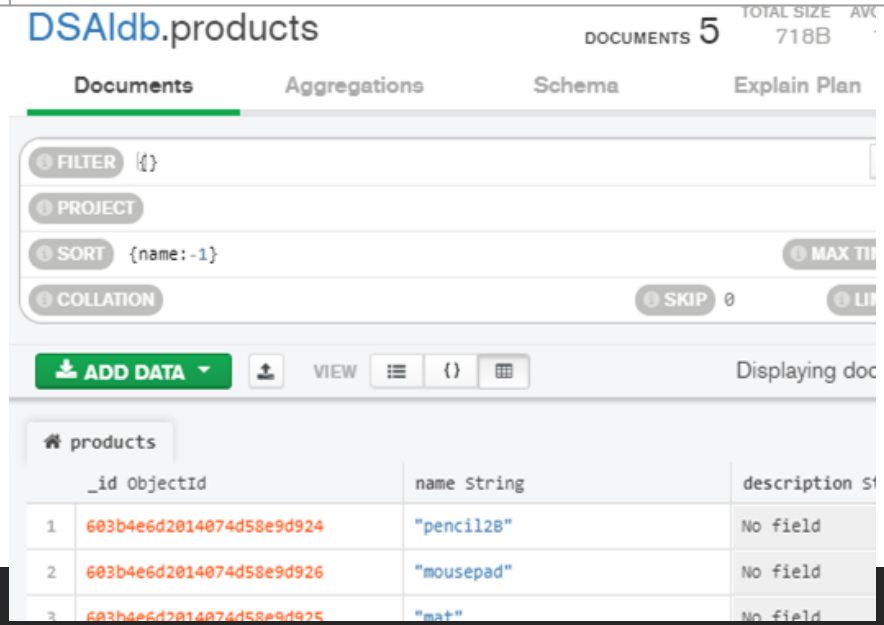
ADD DATA | VIEW | Displaying documents 1 - 5 of 5

```
_id: ObjectId("6039cb812014074d58e9d920")
name: "backpack"
description: "A drawstring style crafted from sturdy black canvas"
> tags: Array
price: 127.59
quantity: 200
> size: Object

_id: ObjectId("603b4e6d2014074d58e9d923")
name: "canvas"
quantity: 100
price: 500
> tags: Array
> size: Object

_id: ObjectId("603b4e6d2014074d58e9d924")
name: "pencil2B"
quantity: 25
price: 200
> tags: Array
> size: Object
darkness: "2B"
```

Order by

Select * from products order by name	db.products.find({}).sort({name:1})
Select * from products order by name desc	db.products.find({}).sort({name:-1})

```
> db.products.find({}).sort({name:1})
< { _id: ObjectId("6039cb812014074d58e9d920"),
    name: 'backpack',
    description: 'A drawstring style crafted fro
    tags: [ 'school', 'travel', 'kids' ],
    price: NumberDecimal("127.59"),
    quantity: 200,
    size: { v: '20', uom: 'l' } }
  { _id: ObjectId("603b4e6d2014074d58e9d923"),
    name: 'canvas',
    quantity: 100,
    price: 500,
    tags: [ 'cotton' ],
```



Find()

db.collection.find(query, projection)

- **query**: document
  - Optional. Specifies selection filter using query operators. To return all documents in a collection, omit this parameter or pass an empty document ({}).
- **projection**: document
  - Optional. Specifies the fields to return in the documents that match the query filter. To return all fields in the matching documents, omit this parameter.

```
db.products.find({name:"backpack"})
```

```
db.products.find({},{_id:0,quantity: 0,tags:0})
```

Projectio: Specified fields to return in the resulting data.

By default, all fields are returned, 0 means **exclude fields** from resulting data.

DSAIdb.products

Documents | Aggregations | Schema | Explain Plan

{ _id:0,
Quantity: 0,
Tags:0 }

Return all fields except _id, quantity, and tags.

FILTER | OPTIONS | FIND

PROJECT {_id:0, quantity:0, tags:0}

SORT | MAX TIME MS 6000

COLLATION | SKIP 0 | LIMIT 0

VIEW | Displaying documents 1 - 1 of 1

```
name: "backpack"
description: "A drawstring style crafted from sturdy black canvas"
price: 127.59
```

db.products.find({}, {_id:0, quantity:0, tags:0 })

Fields to return in the resulting data.

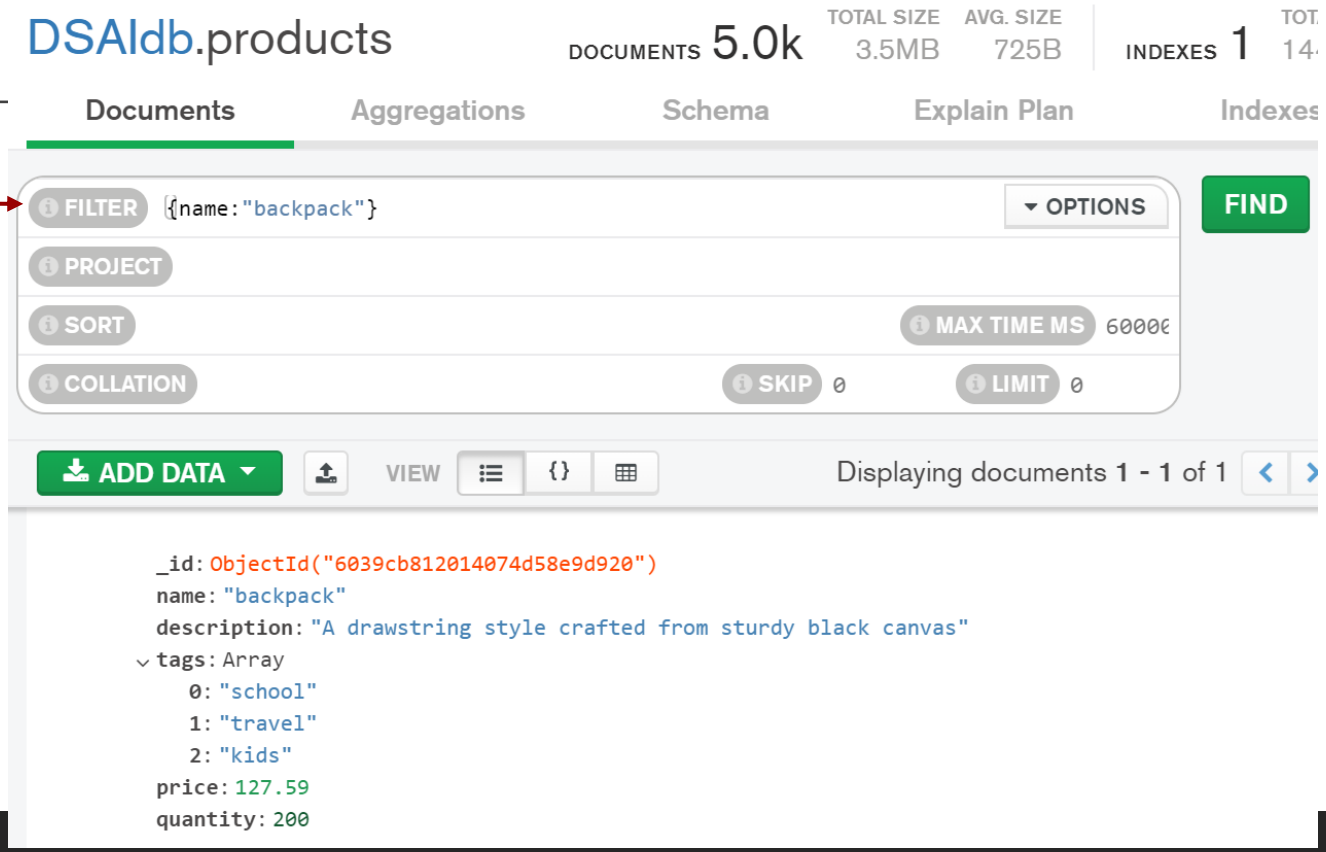
0 = Not Show, 1 = Show

```
>_MongoSH Beta
> db.products.find({},{_id:0,quantity: 0,tags:0})
< { name: 'backpack',
    description: 'A drawstring style crafted from sturdy black canvas',
    price: NumberDecimal("127.59") }
```

Query Specify where condition for product(s) document

Query
{name: "backpack"}

Search product(s) where name is "backpack"



Where cause

db.products.find({name:”backpack”})

```
>_MongoSH Beta

> db.products.find({name:"backpack"})
< { _id: ObjectId("6039cb812014074d58e9d920"),
    name: 'backpack',
    description: 'A drawstring style crafted from sturdy black canvas',
    tags: [ 'school', 'travel', 'kids' ],
    price: NumberDecimal("127.59"),
    quantity: 200 }
```

Query Operators

- $eq: Matches values that are equal to a specified value.
- $gt: Matches values that are greater than a specified value.
- $gte: Matches values that are greater than or equal to a specified value.
- $in: Matches any of the values specified in an array.
- $lt: Matches values that are less than a specified value.
- $lte: Matches values that are less than or equal to a specified value.
- $ne: Matches all values that are not equal to a specified value.
- $nin: Matches none of the values specified in an array.

Recall Product document

products

	_id ObjectId	name String
1	6039cb812014074d58e9d920	"backpack"
2	603b4e6d2014074d58e9d923	"canvas"
3	603b4e6d2014074d58e9d924	"pencil2B"
4	603b4e6d2014074d58e9d925	"mat"
5	603b4e6d2014074d58e9d926	"mousepad"

```
_id: ObjectId("6039cb812014074d58e9d920")
name: "backpack"
description: "A drawstring style crafted from
tags: Array
    0: "school"
    1: "travel"
    2: "kids"
price: 127.59
quantity: 200
size: Object
    v: "20"
    uom: "l"
```

```
_id: ObjectId("603b4e6d2014074d58e9d924")
name: "pencil2B"
quantity: 25
price: 200
tags: Array
    0: "pencil"
    1: "school"
size: Object
    h: 14
    w: 1
    uom: "cm"
darkness: "2B"
```

db.products.find({name: {$ne: "backpack"}})

```
> db.products.find({name: {$ne: "backpack"}})
< { _id: ObjectId("603b4e6d2014074d58e9d923"),
    name: 'canvas',
    quantity: 100,
    price: 500,
    tags: [ 'cotton' ],
    size: { h: 28, w: 35.5, uom: 'cm' } }
  { _id: ObjectId("603b4e6d2014074d58e9d924"),
    name: 'pencil2B',
    quantity: 25,
    price: 200,
    tags: [ 'pencil', 'school' ],
    size: { h: 14, w: 1, uom: 'cm' },
    darkness: '2B' }
```

DSAldb.products

DOCUMENTS 5 TOTAL 71

Documents | Aggregations | Schema | Explai

FILTER {name: {$ne: "backpack"}}

PROJECT

SORT

COLLATION

SKIP 0

ADD DATA | VIEW | Displa

products

	_id ObjectId	name String	quanti
1	603b4e6d2014074d58e9d923	"canvas"	100
2	603b4e6d2014074d58e9d924	"pencil2B"	25
3	603b4e6d2014074d58e9d925	"mat"	85
4	603b4e6d2014074d58e9d926	"mousepad"	25

db.products.find({price: {$lt: 100}})

```
> db.products.find({price: {$lt: 100}})
< { _id: ObjectId("603b4e6d2014074d58e9d925"),
    name: 'mat',
    quantity: 85,
    price: 99,
    tags: [ 'gray' ],
    size: { h: 27.9, w: 35.5, uom: 'cm' } }
  { _id: ObjectId("603b4e6d2014074d58e9d926"),
    name: 'mousepad',
    quantity: 25,
    price: 29,
    tags: [ 'gel', 'blue' ],
    size: { h: 19, w: 22.85, uom: 'in' } }
```

DSAIdb.products

DOCUMENTS

Documents | Aggregations | Schema

FILTER {price: {$lt: 100}}

PROJECT

SORT

COLLATION

SK

ADD DATA | VIEW | {}

products

	_id ObjectId	name String
1	603b4e6d2014074d58e9d925	"mat"
2	603b4e6d2014074d58e9d926	"mousepad"

LIKE

Search products where name like %pen%

SQL statements	MongoDB statements
SELECT * FROM Products WHERE name LIKE “%pen%”	db.products.find({ name : { $regex: /pen/ } })

Result

DSAldb.products

DOCUMENTS 8

Documents | Aggregations | Schema | Ex

FILTER {name:{$regex:/pen/}}

ADD DATA | VIEW | {} | Di

products

	_id ObjectId	name String	qu
1	603b4e6d2014074d58e9d924	"pencil2B"	25
2	603b60852014074d58e9d927	"erasablepen"	25
3	603b610d2014074d58e9d928	"whiteboardpen"	25
4	603b61342014074d58e9d929	"redpen"	25

```
> db.products.find({name:{$regex:/pen/}})
< { _id: ObjectId("603b4e6d2014074d58e9d924"),
    name: 'pencil2B',
    quantity: 25,
    price: 200,
    tags: [ 'pencil', 'school' ],
    size: { h: 14, w: 1, uom: 'cm' },
    darkness: '2B' }
{ _id: ObjectId("603b60852014074d58e9d927"),
  name: 'erasablepen',
  quantity: 250,
  price: 20,
  tags: [ 'pen', 'school', 'red', 'erasable' ],
  size: { h: 14, w: 1, uom: 'cm' },
  color: 'red' }
```

IN

Search products where color is in the list

DSAldb.products

DOCUMENTS 8 | TOTAL SIZE 1.3KB | AVG. SIZE 168B | INDEXES 1

Documents | Aggregations | Schema | Explain Plan | Indexes

FILTER {color: {$in: ["red","blue"]}}

OPTIONS

FIND

PROJECT {_id:0,name:1,color:1}

SORT

MAX TIME MS 60000

COLLATION

SKIP 0

LIMIT 0

VIEW

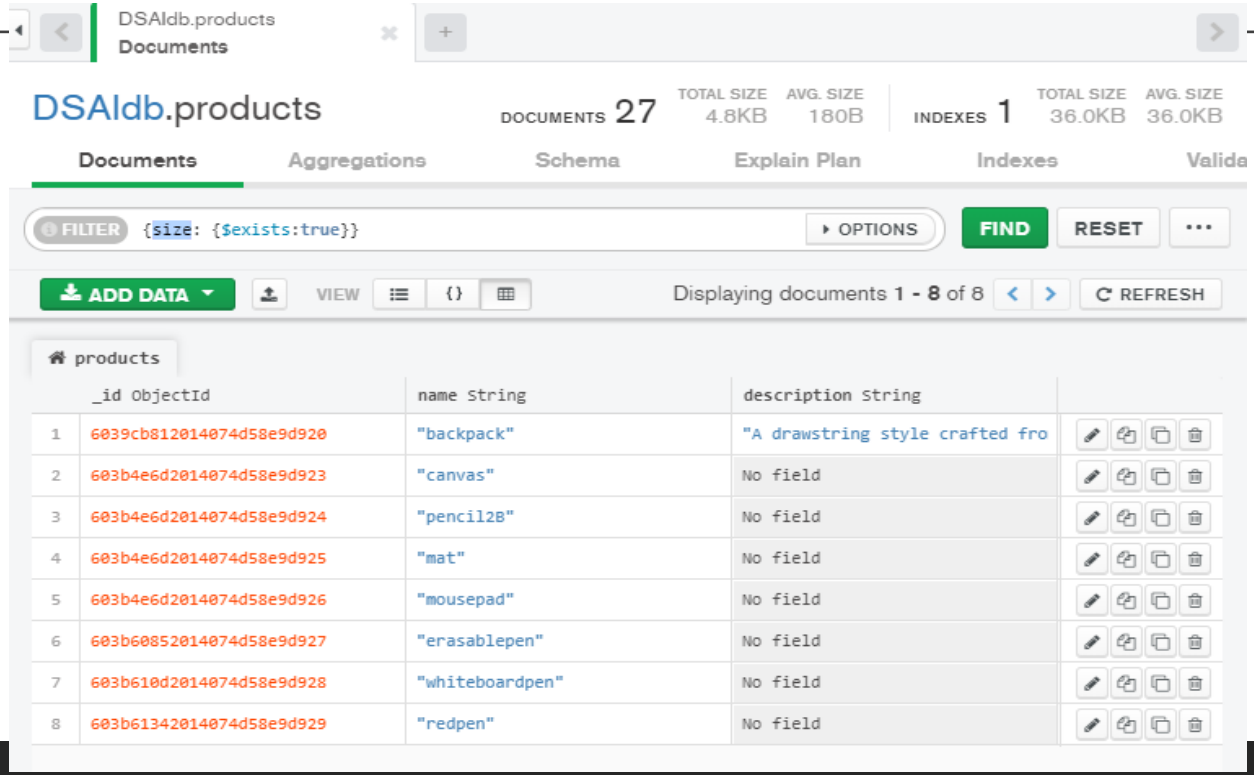
Displaying documents 1 - 2 of 2

products

	name String	color String
1	"erasablepen"	"red"
2	"redpen"	"blue"

EXISTS

Search products where size exists



	_id ObjectId	name String	description String
1	6039cb812014074d58e9d920	"backpack"	"A drawstring style crafted fro
2	603b4e6d2014074d58e9d923	"canvas"	No field
3	603b4e6d2014074d58e9d924	"pencil2B"	No field
4	603b4e6d2014074d58e9d925	"mat"	No field
5	603b4e6d2014074d58e9d926	"mousepad"	No field
6	603b60852014074d58e9d927	"erasablepen"	No field
7	603b610d2014074d58e9d928	"whiteboardpen"	No field
8	603b61342014074d58e9d929	"redpen"	No field

Specify more than one conditions

- $AND: both conditions have to be satisfied
- $OR: only a condition satisfy

AND

Search products where price greater than 500 and less than or equal to 1000.

```
db.products.find({price: {$gt: 50, $lte: 100}})
```

FILTER {price: {$gt: 50, $lte: 100}}

Result

ADD DATA VIEW

```
_id: ObjectId("603b4e6d2014074d58e9d925")
name: "mat"
quantity: 85
price: 99
> tags: Array
> size: Object
```

All products & price

FILTER

PROJECT {_id:0,name:1, price:1}

SORT {price:-1}

COLLATION

VIEW

products

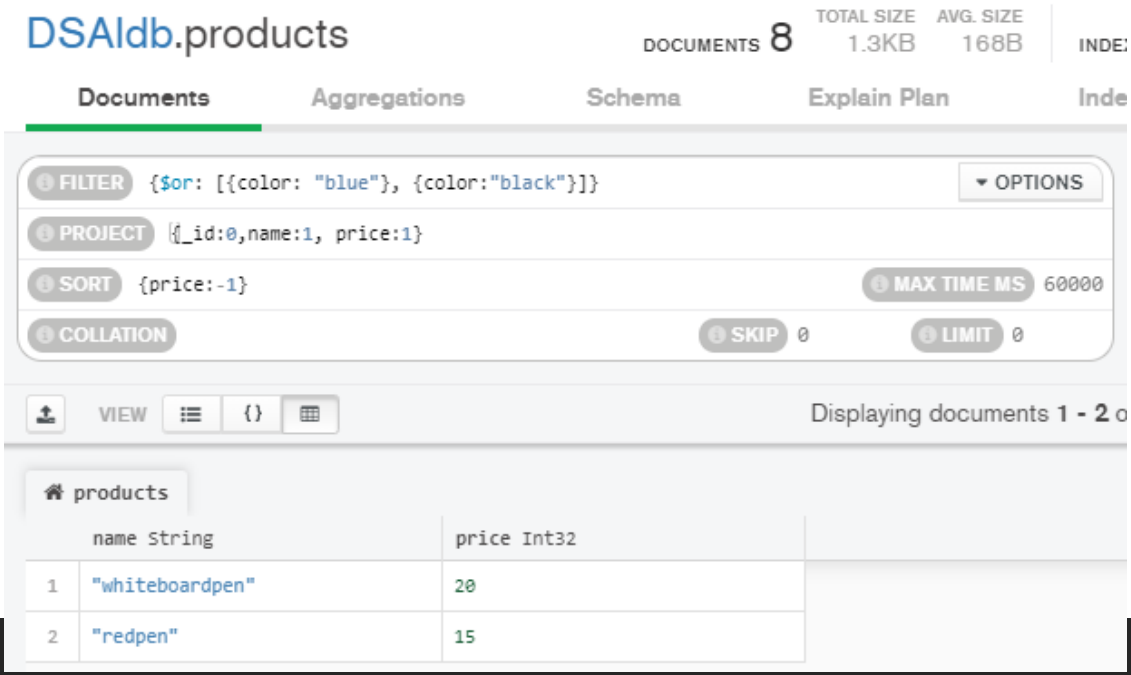
	name String	price Int32
1	"canvas"	500
2	"pencil2B"	200
3	"backpack"	127.59
4	"mat"	99
5	"erasablepen"	50
6	"mousepad"	29
7	"whiteboardpen"	20
8	"redpen"	15

OR

Search blue pen or black pen

```
db.products.find({$or: [{color: "blue"}, {color:"black"}]})
```

Result



QUERY on Complex fields

Nested Fields or array fields

Query Nested Field Uses **dot notation** to access fields in an embedded document:

Array Field
E.x. tags

Nested Field
Ex. size

```
_id: ObjectId("6039cb812014074d58e9d920")
name: "backpack"
description: "A drawstring style crafted from sturdy black canvas"
tags: Array
    0: "school"
    1: "travel"
    2: "kids"
price: 127.59
quantity: 200
size: Object
    v: "20"
    uom: "l"
```

Query Nested Field

Select all products where use in as uom of size

db.products.find({“size.uom”:”in”})

products

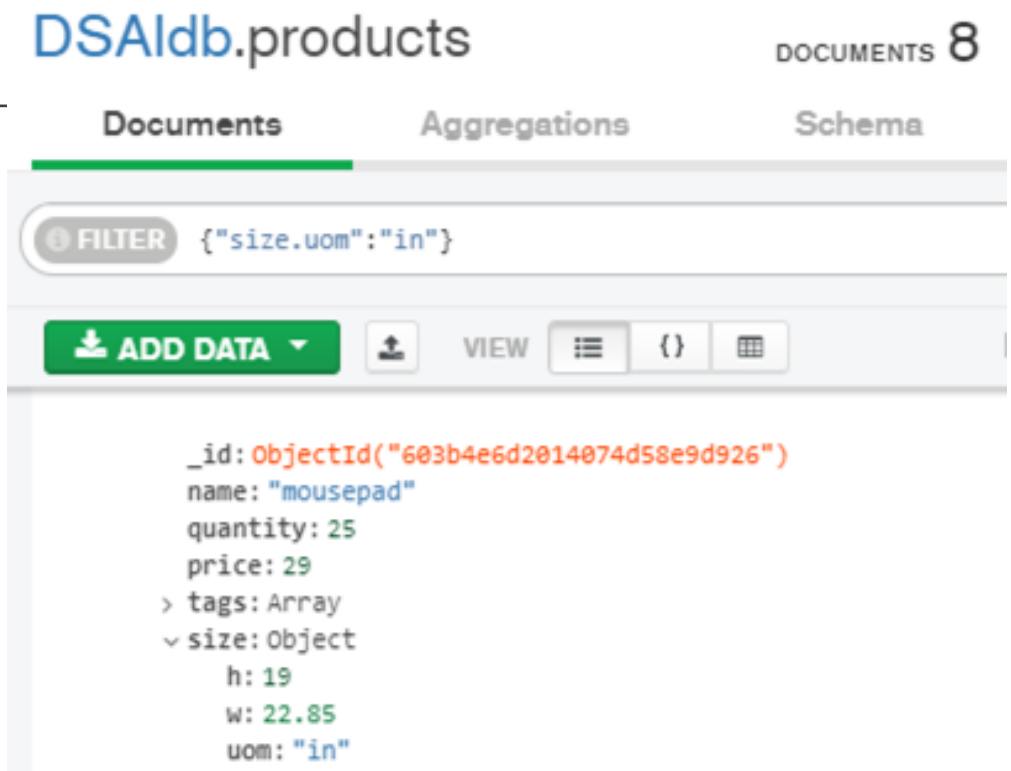
	_id ObjectId	name String
1	6039cb812014074d58e9d920	"backpack"
2	603b4e6d2014074d58e9d923	"canvas"
3	603b4e6d2014074d58e9d924	"pencil2B"
4	603b4e6d2014074d58e9d925	"mat"
5	603b4e6d2014074d58e9d926	"mousepad"
6	603b60852014074d58e9d927	"erasablepen"
7	603b610d2014074d58e9d928	"whiteboardpen"
8	603b61342014074d58e9d929	"redpen"

products | size { }

	_id ObjectId	uom String
1	6039cb812014074d58e9d920	"l"
2	603b4e6d2014074d58e9d923	"cm"
3	603b4e6d2014074d58e9d924	"cm"
4	603b4e6d2014074d58e9d925	"cm"
5	603b4e6d2014074d58e9d926	"in"
6	603b60852014074d58e9d927	"cm"
7	603b610d2014074d58e9d928	"cm"
8	603b61342014074d58e9d929	"cm"

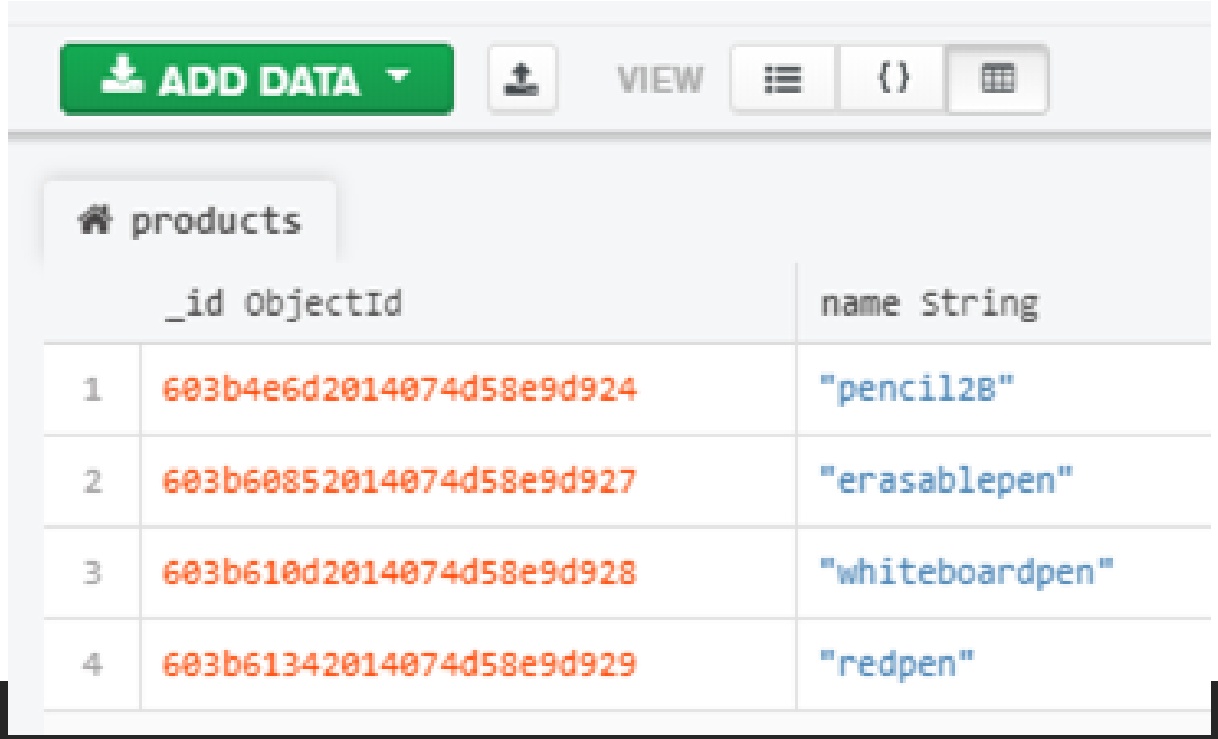
Result

db.products.find
(
  {“size.uom”:”in”}
)



TASK

Select products where their height less than 15 cm and the UOM is in cm.



	_id ObjectId	name String
1	603b4e6d2014074d58e9d924	"pencil2B"
2	603b60852014074d58e9d927	"erasablepen"
3	603b610d2014074d58e9d928	"whiteboardpen"
4	603b61342014074d58e9d929	"redpen"

Query Array Field

Search products for kids using tags

```
> db.products.find({tags:"kids"})
< { _id: ObjectId("6039cb812014074d58e9d920"),
    name: 'backpack',
    description: 'A drawstring style crafted fr
    tags: [ 'school', 'travel', 'kids' ],
    price: NumberDecimal("127.59"),
    quantity: 200,
    size: { v: '20', uom: 'l' } }
```

DSAIdb.products

DOCUMENTS 27 TOTAL SIZE 4.8KB AVG. 18

Documents Aggregations Schema Explain P

FILTER {tags:"kids"}

ADD DATA VIEW Displaying docu

```
_id: ObjectId("6039cb812014074d58e9d920")
name: "backpack"
description: "A drawstring style crafted from sturdy black canvas"
tags: Array
  0: "school"
  1: "travel"
  2: "kids"
price: 127.59
quantity: 200
size: Object
  v: "20"
  uom: "l"
```

Use case: U6

Staff manages Product information

6.1(1) add product information
6.1(2) update bestseller flag for product
6.1(3) delete rejected sale document from sales

Core Concepts

- INSERT Operations
- UPDATE Operations
- DELETE Operations

Insert Documents

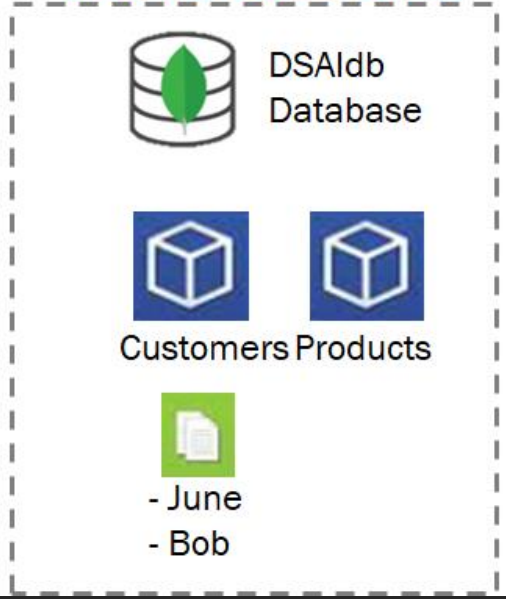
db.collection.insertMany()

Syntax:

```
db.collection.insertMany(
    [ <document 1> , <document 2>, ... ]
)
```

Document Model Architecture

Atlas MongoDB Server



Insert products

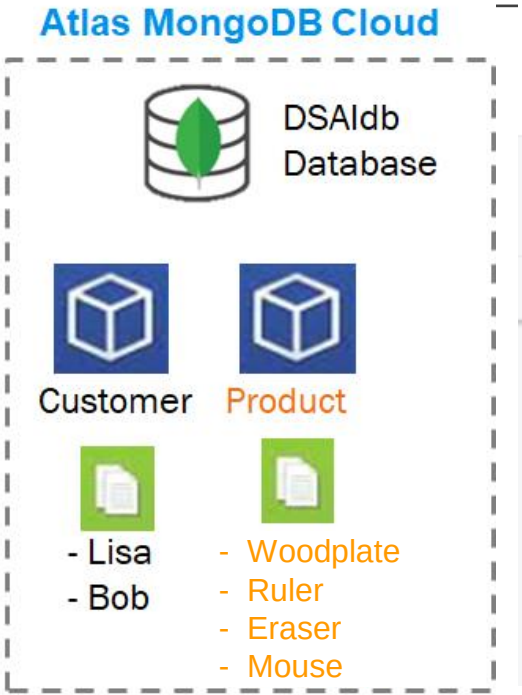
Insert product documents

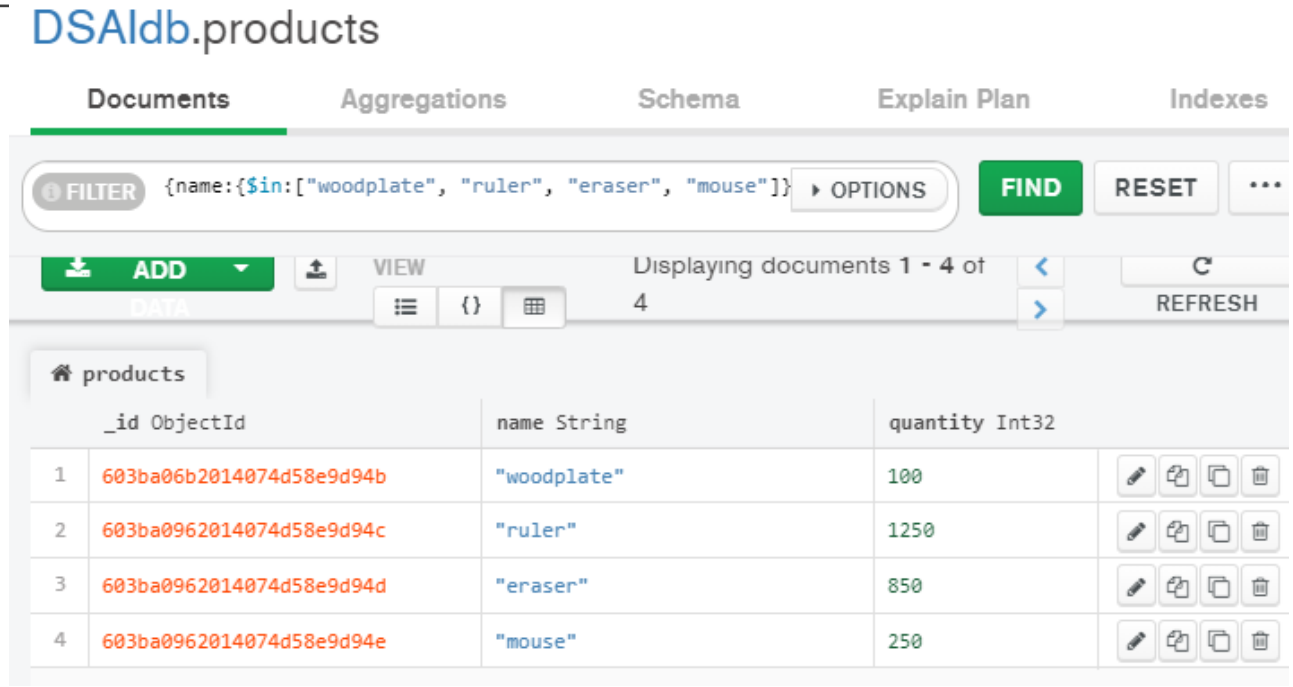
```
db.products.insertOne(
  { name: "woodplate", quantity: 100, price: 500, tags: ["wood",”school”]})
db.products.insertMany([
  { name: "ruler", quantity: 1250, price: 20, tags: ["ruler",”pooh”], size: { h: 30, w: 5, uom: "cm" } },
  { name: “eraser", quantity: 850, price: 9,  tags: ["gray",”eraser”,”pencil”], size: { h: 5, w: 2, uom: "cm" } },
  { name: "mouse", quantity : 250, price: 199, tags: ["wired", "black"] }
])
```

Result

```
> db.products.insertOne({ name: "woodplate", quantity: 100, price: 500, tags: ["wood","school"]})
< { acknowledged: true,
    insertedId: ObjectId("603ba06b2014074d58e9d94b") }
> db.products.insertMany([
      { name: "ruler", quantity: 1250, price: 20, tags: ["ruler","pooh"], size: { h: 30, w: 5, uom
      { name: "eraser", quantity: 850, price: 9,  tags: ["gray","eraser","pencil"], size: { h: 5,
      { name: "mouse", quantity : 250, price: 199, tags: ["wired", "black"] }
  ])
< { acknowledged: true,
    insertedIds:
     { '0': ObjectId("603ba0962014074d58e9d94c"),
       '1': ObjectId("603ba0962014074d58e9d94d"),
       '2': ObjectId("603ba0962014074d58e9d94e") } }
```

Verify the product insertion





	_id ObjectId	name String	quantity Int32
1	603ba06b2014074d58e9d94b	"woodplate"	100
2	603ba0962014074d58e9d94c	"ruler"	1250
3	603ba0962014074d58e9d94d	"eraser"	850
4	603ba0962014074d58e9d94e	"mouse"	250

Update product(s) add “isBestSeller” field

SQL Query	MongoDB command
Alter table products Add isBestSeller boolean + Update products set isBestSeller = FALSE	*Schemaless, no need to define.* *If defined,* db.products.updateMany(← collection {}, ← update filter {$set: {isBestSeller: false}} ← update action
Update products set isBestSeller = TRUE Where _id = “603b60852014074d58e9d927”	db.products.updateMany({_id: ObjectId(“603b60852014074d58e9d927”)}, {$set: {isBestSeller: true}}
	To remove bestseller db.products.updateMany({isBestSeller:true},{ $unset: {"isBestSeller": "true"}})

Result

```
> db.products.updateMany(
      {_id: ObjectId("603b60852014074d58e9d927")},
      {$set: {isBestSeller: true}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
```

Verify DSAIdb.products

Documents | Aggregations | Schema | Explain Plan

FILTER {_id: ObjectId("603b60852014074d58e9d927")} OPTIONS FIND

ADD DATA | VIEW | Displaying documents 1 - 1 of 1

```
_id: ObjectId("603b60852014074d58e9d927")
name: "erasablepen"
quantity: 250
price: 50
> tags: Array
> size: Object
color: "red"
isBestSeller: true
```

$unset

```
> db.products.updateMany( {isBestSeller:true},{ $unset: {"isBestSeller": "true"}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
```

Delete rejected sale documents from sales

SQL Query	MongoDB command
Delete from sales Where status = ‘reject’	db.sales.deleteMany({status: “reject”})

collection

delete filter

```
> db.sales.find({status: "reject"})
< { _id: ObjectId("5bd761dcae323e45a93ccfec"),
    saleDate: 2017-12-03T18:39:48.253Z,
    items:
     [ { name: 'backpack',
         tags: [ 'school', 'travel', 'kids' ],
         price: NumberDecimal("127.59"),
         quantity: 3 },
       { name: 'notepad',
         tags: [ 'office', 'writing', 'school' ],
```

```
> db.sales.deleteMany({status: "reject"})
< { acknowledged: true, deletedCount: 1 }
> db.sales.find({status: "reject"})
<
>
```

Use Case: U7

Staff views daily report

Core Concepts

Aggregate Operations

Join/Lookup

Aggregation

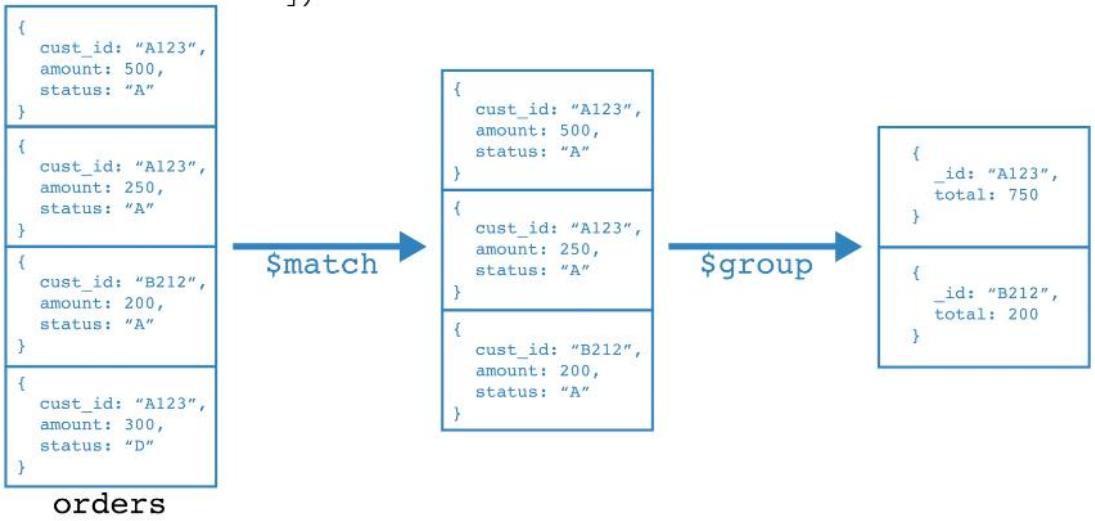
- Aggregation operations group values from multiple documents together and can perform a variety of operations on the grouped data to return a single result.
- SQL terms vs. MongoDB Aggregation Operators

WHERE	`$match`	LIMIT	`$limit`
GROUP BY	`$group`	SUM()	`$sum`
HAVING	`$match`	COUNT()	`$sum` `$sortByCount`
SELECT	`$project`		
ORDER BY	`$sort`	join	`$lookup`

Other simple example

Collection

```
db.orders.aggregate( [
  $match stage ⟶ { $match: { status: “A” } },
  $group stage ⟶ { $group: { _id: “$cust_id”,total:  {$sum: “$amount” } } }
                 ])
```



SUM- count the number of order of each customer

Using Compass

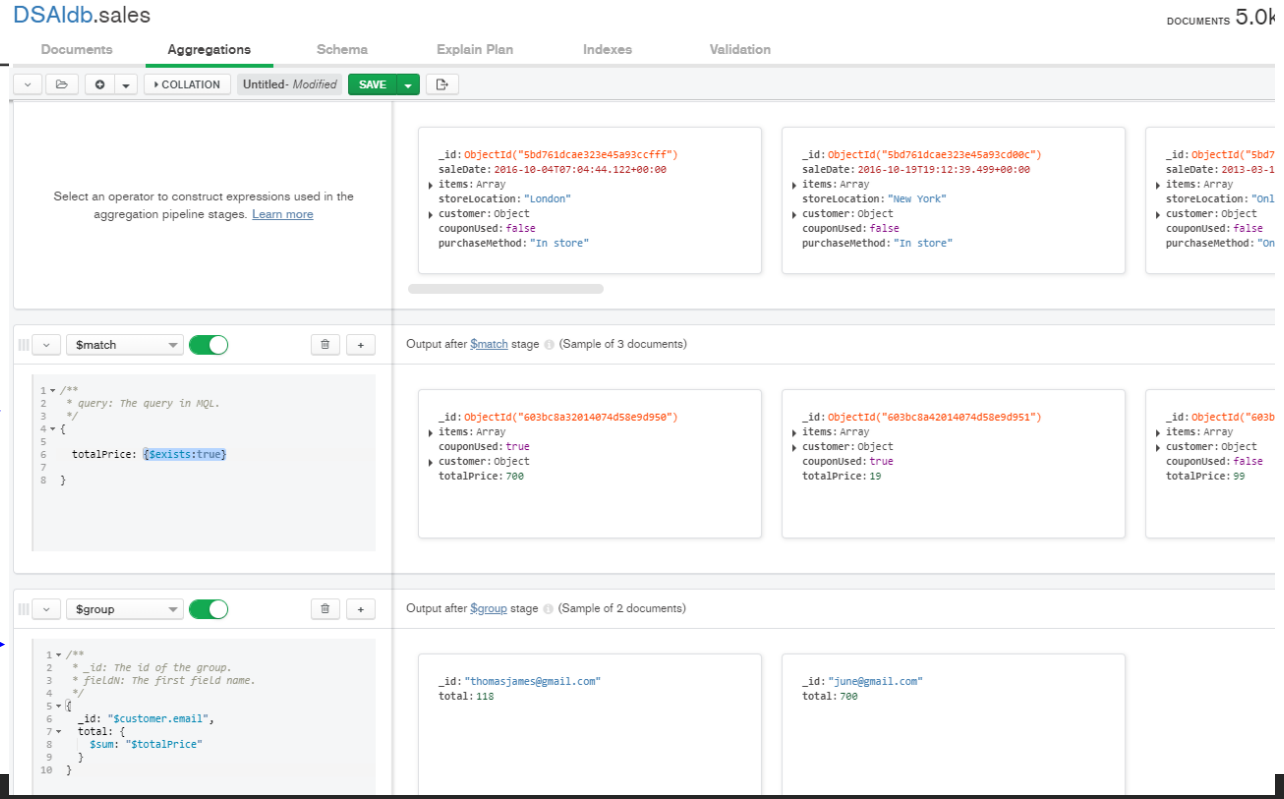
All Data

$match stage

{totalPrice: {$exists:true}}

$group stage

```
{ _id: "$customer.email",
  total: {
    $sum: "$totalPrice" }}
```



SUM- sum total price for each customer

Shell

```
> db.sales.aggregate([{$match:{totalPrice:{$exists:true}}},
  {$group:{_id:"$customer.email", total: { $sum: "$totalPrice"}}}])
< [ { _id: 'june@gmail.com', total: 700 },
    { _id: 'thomasjames@gmail.com', total: 118 } ]
```

Count - count the number of order of each customer

Using Compass

All Data

$match stage

$group stage

DSAldb.sales
Aggregations

DSAldb.sales

DOCUMENTS 5.0k | TOTAL SIZE 4.2MB | AVG. SIZE 873B | INDEXES 1 | TOTAL SIZE 84.0KB | AVG. SIZE 84.0KB

Documents | Aggregations | Schema | Explain Plan | Indexes | Validation

COLLATION | Untitled- *Modified* | SAVE | SAMPLE MODE | AUTO PREVIEW

Select an operator to construct expressions used in the aggregation pipeline stages. Learn more

```
_id: ObjectId("5bd761dcae323e45a93ccfec")
saleDate: 2017-12-03T18:39:48.253+00:00
▸ items: Array
storeLocation: "London"
▸ customer: Object
couponUsed: false
purchaseMethod: "In store"
```

```
_id: ObjectId("5bd761dcae323e45a93
saleDate: 2016-10-04T07:04:44.122+
▸ items: Array
storeLocation: "London"
▸ customer: Object
couponUsed: false
purchaseMethod: "In store"
```

$match

```
/**
 * query: The query in MQL.
 */
{
  couponUsed:true
}
```

Output after $match stage (Sample of 20 documents)

```
_id: ObjectId("5bd761dcae323e45a93cd04e")
saleDate: 2015-08-30T07:37:03.111+00:00
▸ items: Array
storeLocation: "London"
▸ customer: Object
couponUsed: true
purchaseMethod: "In store"
```

```
_id: ObjectId("5bd761dcae323e45a93
saleDate: 2015-01-31T10:38:13.836+
▸ items: Array
storeLocation: "Austin"
▸ customer: Object
couponUsed: true
purchaseMethod: "In store"
```

$group

```
/**
 * _id: The id of the group.
 * fieldN: The first field name.
 */
{
  _id: "$customer.email",
  countOrder: {
    $sum: 1
  }
}
```

Output after $group stage (Sample of 20 documents)

```
_id: "sherry31@yahoo.com"
countOrder: 9
```

```
_id: "thomasjames@gmail.com"
countOrder: 7
```

Count - count the number of order of each customer

Using Shell

```
>_MongoSH Beta
  {$group: {_id:"$customer.email", count:{$sum:1}}}])
< [ { _id: 'sherry31@yahoo.com', count: 9 },
    { _id: 'thomasjames@gmail.com', count: 7 },
    { _id: 'scottjonathan@yahoo.com', count: 11 },
    { _id: 'ble@gmail.com', count: 11 },
    { _id: 'james04@gmail.com', count: 38 },
    { _id: 'colinward@hotmail.com', count: 15 },
    { _id: 'ronald23@hotmail.com', count: 14 },
    { _id: 'cindy86@yahoo.com', count: 23 },
    { _id: 'jdavis@hotmail.com', count: 31 },
    { _id: 'william88@yahoo.com', count: 5 },
    { _id: 'nicholas62@hotmail.com', count: 18 },
    { _id: 'eric79@hotmail.com', count: 25 },
    { _id: 'madison55@gmail.com', count: 23 },
```

Average, Min, Max

Average

```
> db.sales.aggregate([{$match:{totalPrice:{$exists:true}}},
  {$group:{_id:"$customer.email", average: { $avg: "$totalPrice"}}}])
< [ { _id: 'june@gmail.com', average: 700 },
    { _id: 'thomasjames@gmail.com', average: 59 } ]
```

Min

```
> db.sales.aggregate([{$match:{totalPrice:{$exists:true}}},
  {$group:{_id:"$customer.email", minimum: { $min: "$totalPrice"}}}])
< [ { _id: 'thomasjames@gmail.com', minimum: 19 },
    { _id: 'june@gmail.com', minimum: 700 } ]
```

Max

```
> db.sales.aggregate([{$match:{totalPrice:{$exists:true}}},
  {$group:{_id:"$customer.email", maximum: { $max: "$totalPrice"}}}])
< [ { _id: 'june@gmail.com', maximum: 700 },
    { _id: 'thomasjames@gmail.com', maximum: 99 } ]
```

Having

$group

```
/**
 * _id: The id of the group.
 * fieldN: The first field name.
 */
{
  _id: "$customer.email",
  total: {
    $sum: "$totalPrice"
  }
}
```

Output after $group stage (Sample of 2 documents)

```
_id: "thomasjames@gmail.com"
total: 118
```

```
_id: "june@gmail.com"
total: 700
```

$match

```
/**
 * query: The query in MQL.
 */
{
  total:{$gt:150}
}
```

Output after $match stage (Sample of 1 document)

```
_id: "june@gmail.com"
total: 700
```

Sort

$group

```
/**
 * _id: The id of the group.
 * fieldN: The first field name.
 */
{
  _id: "$customer.email",
  total: {
    $sum: "$totalPrice"
  }
}
```

Output after $group stage (Sample of 2 documents)

```
_id: "thomasjames@gmail.com"
total: 118
```

```
_id: "june@gmai
total: 700
```

$sort

```
/**
 * Provide any number of field/order pairs.
 */
{
  total: -1
}
```

Output after $sort stage (Sample of 2 documents)

```
_id: "june@gmail.com"
total: 700
```

```
_id: "thomasjam
total: 118
```

Limit

Before

```
_id: "june@gmail.com"
total: 700
```

```
_id: "thomasjames@gmail.com"
total: 118
```

$limit

```
/**
 * Provide the number of documents to limit.
 */
1
```

Output after $limit stage (Sample of 1 document)

After

```
_id: "june@gmail.com"
total: 700
```

Join/Lookup

- Syntax

```
{
   $lookup:
     {
       from: <collection to join>,
       localField: <field from the input documents>,
       foreignField: <field from the documents of the "from" collection>,
       as: <output array field>
     }
}
```

Join Customer vs. Sale

```
db.sales.aggregate([
  {
        $lookup:
        {
                from: 'customers',
                localField: 'customer.email',
                foreignField: 'email',
                as: 'customer.moreInfo'
        }
  }
])
```

```
_id: ObjectId("5bd761dcae323e45a93ccfff")
saleDate: 2016-10-04T07:04:44.122+00:00
▸ items: Array
storeLocation: "London"
▾ customer: Object
    gender: "M"
    age: 56
    email: "nicholas62@hotmail.com"
    customerRating: 5
  ▾ moreInfo: Array
        _id: ObjectId("5ca4bbcea2dd94ee58162b38")
        name: "Amanda Hammond"
        address: "41554 Wood View
                  New Scott, MO 28699"
        birthdate: 1980-03-12T00:58:32.000+00:00
        email: "nicholas62@hotmail.com"
      ▸ accounts: Array
        password: "christine68"
couponUsed: false
purchaseMethod: "In store"
```

More $Lookup: https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/#pipe._S_lookup

Use Case: U8

Owner views summary report

8.1(1) Get the top 3 popular product
8.1(2) Get total sales group by month

Core Concepts

Aggregate Operations

8.1(1) Get the top 3 popular product

```
_id: ObjectId("5bd761dcae323e45a93ccfff")
saleDate: 2016-10-04T07:04:44.122+00:00
items: Array
  0: Object
    name: "binder"
    tags: Array
    price: 20.08
    quantity: 7
  1: Object
  2: Object
  3: Object
  4: Object
storeLocation: "London"
customer: Object
couponUsed: false
```

$unwind

```
/*...*/
{
  path: "$items"
}
```

```
_id: ObjectId("5bd761dcae323e45a93ccfff")
saleDate: 2016-10-04T07:04:44.122+00:00
items: Object
  name: "binder"
  tags: Array
  price: 20.08
  quantity: 7
storeLocation: "London"
customer: Object
```

```
_id: ObjectId("5bd761dcae323e45a93ccfff")
saleDate: 2016-10-04T07:04:44.122+00:00
items: Object
  name: "envelopes"
  tags: Array
  price: 22.9
  quantity: 3
storeLocation: "London"
customer: Object
```

$group

```
/*...*/
{
  _id: "$items.name",
  total: {
    $sum: "$items.quantity"
  }
}
```

```
_id: "backpack"
total: 6921

_id: "notepad"
total: 20727

_id: "canvas"
total: 1
```

$sort

```
/*...*/
{
  total: -1
}
```

$limit

```
/*...*/
3
```

8.1(2) Get total sales group by month

$group

```
/*[...]*/
{
  _id: {$month: "$saleDate"},
  numberOfSale: {
    $sum: 1
  },
  totalPrice: {
    $sum: "$totalPrice"
  },
  avgPricePerSale: {
    $avg: "$totalPrice"
  }
}
```

```
_id: 1
numberOfSale: 445
totalPrice: 19
avgPricePerSale: 19
```

```
_id: 2
numberOfSale: 360
totalPrice: 700
avgPricePerSale: 700
```

...

Assignment

Use case: U4

Checkout

4.1 Retrieve cart information
4.2 Get latest price
4.3 Add Sales information
4.4 Remove purchased products from cart
4.5 Activate U5 with action='U4' (Write log)

Core Concepts : Put it all together

- Read data in Redis
- Read data from MongoDB
- Insert data to MongoDB
- Delete data in Redis
- Insert data in Redis