



AT82.02

DATA MODELING AND MANAGEMENT

LAB06: DOCUMENT DB (MONGODB)

Outline

- Scenario: TikTock E-Commerce
- MongoDB: Getting started
- MongoDB Practice
 - INSERT
 - UPDATE
 - DELETE
 - Basic Query

Project: RDB to NoSQL migration

Scenario :

TickTock is an office supply company which has several store locations throughout the world and an online store website with relational database.

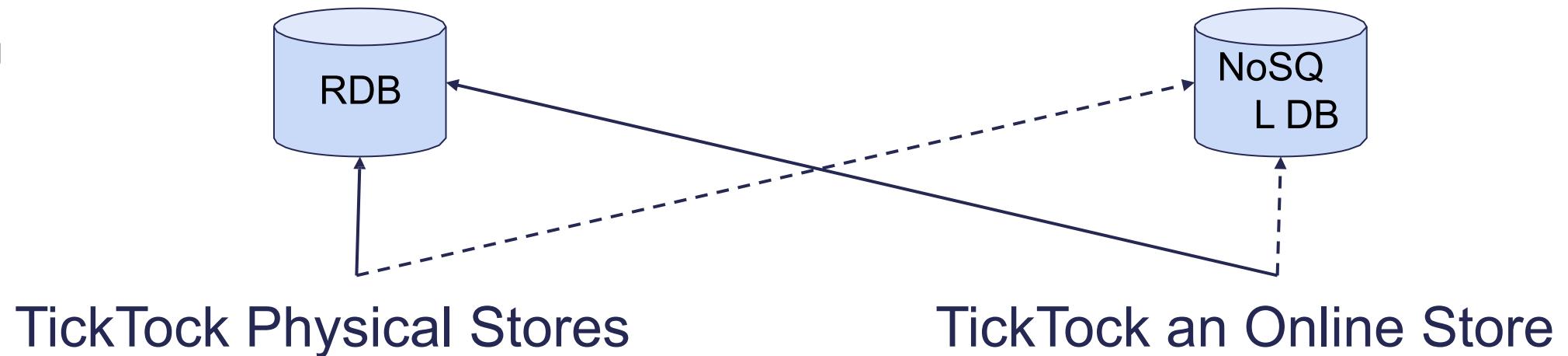


With flexibility of schema, performance and scalability features, TickTock is moving to NOSQL database.

YOU, as a db admin should design and implement query for core functions.

Project: RDB to NoSQL migration

→ Existing
→ Future



TickTock an Online Store



Require query statement for these core functions

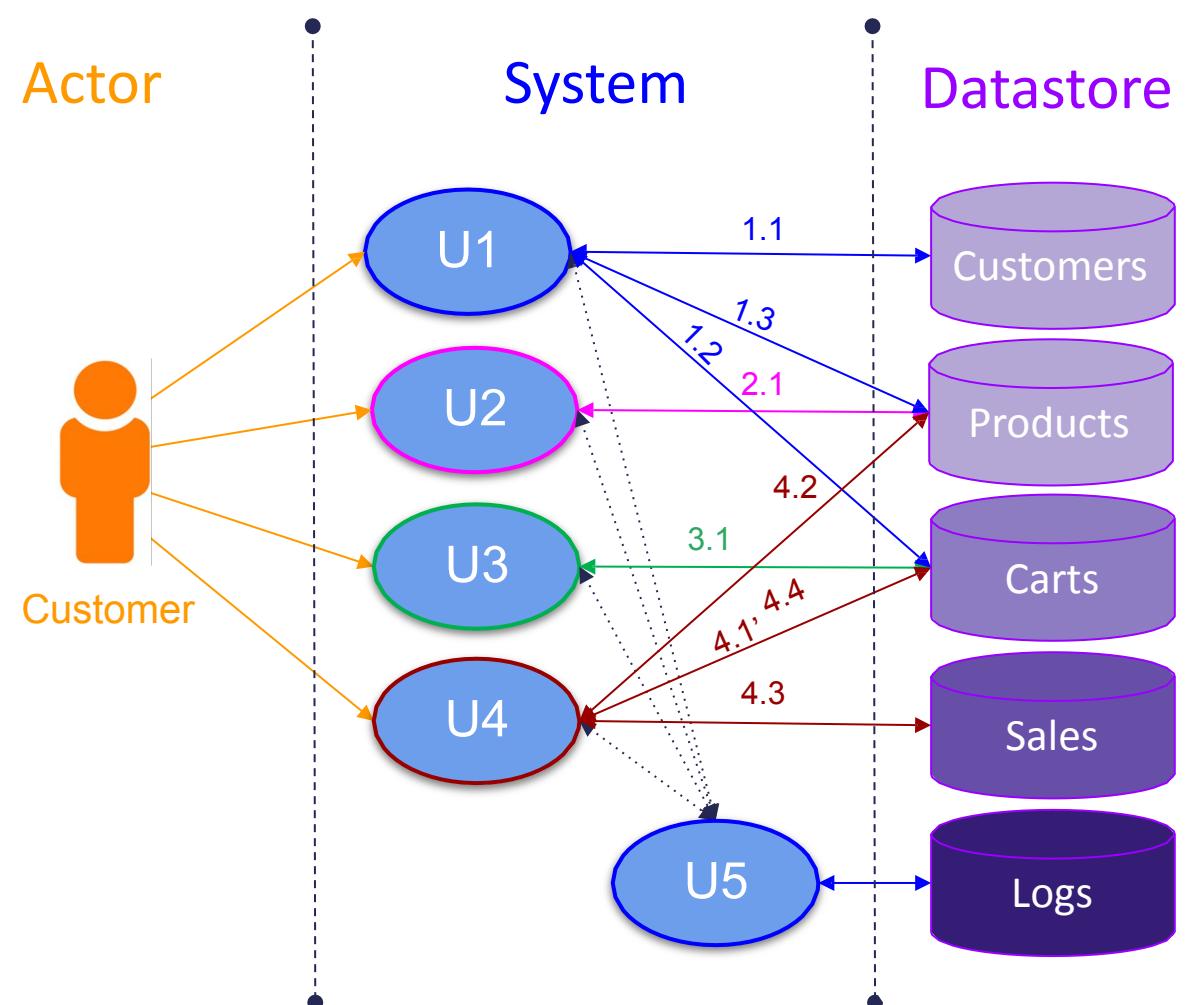
- **Support e-commerce functions for customers:** (U1) Search product, (U2) Add product(s) to cart, and (U3) Checkout
Note that: The payment process is excluded to simplify the system.
- **Support daily operation for store staff:** (U4) Add new products, (U5) Update product information, (U6) Delete duplicate product, (U9) Search insight customer behavior
- **Support summary report for owner:** (U7) View popular product report by month, (U8) Total sales, count , average sales

Existing Data in RDB

Data	Description	Fields
Product data	Information about products	product id, name, description ,price, quantity in stock, product keywords
Customer data	Information about customers	email as username as custid, customer name, address, birthdate, password, bank account
Cart data	Information about cart and tentative purchased products	Cartid, product id, quantity, price
Sales data	Information about order and purchased products	the item(s) purchased, information on the customer who made the purchase, several other details regarding the sale.
Log data	Information about customers' activities	Action datetime, Action type, Action description, customer Action type = {"1}

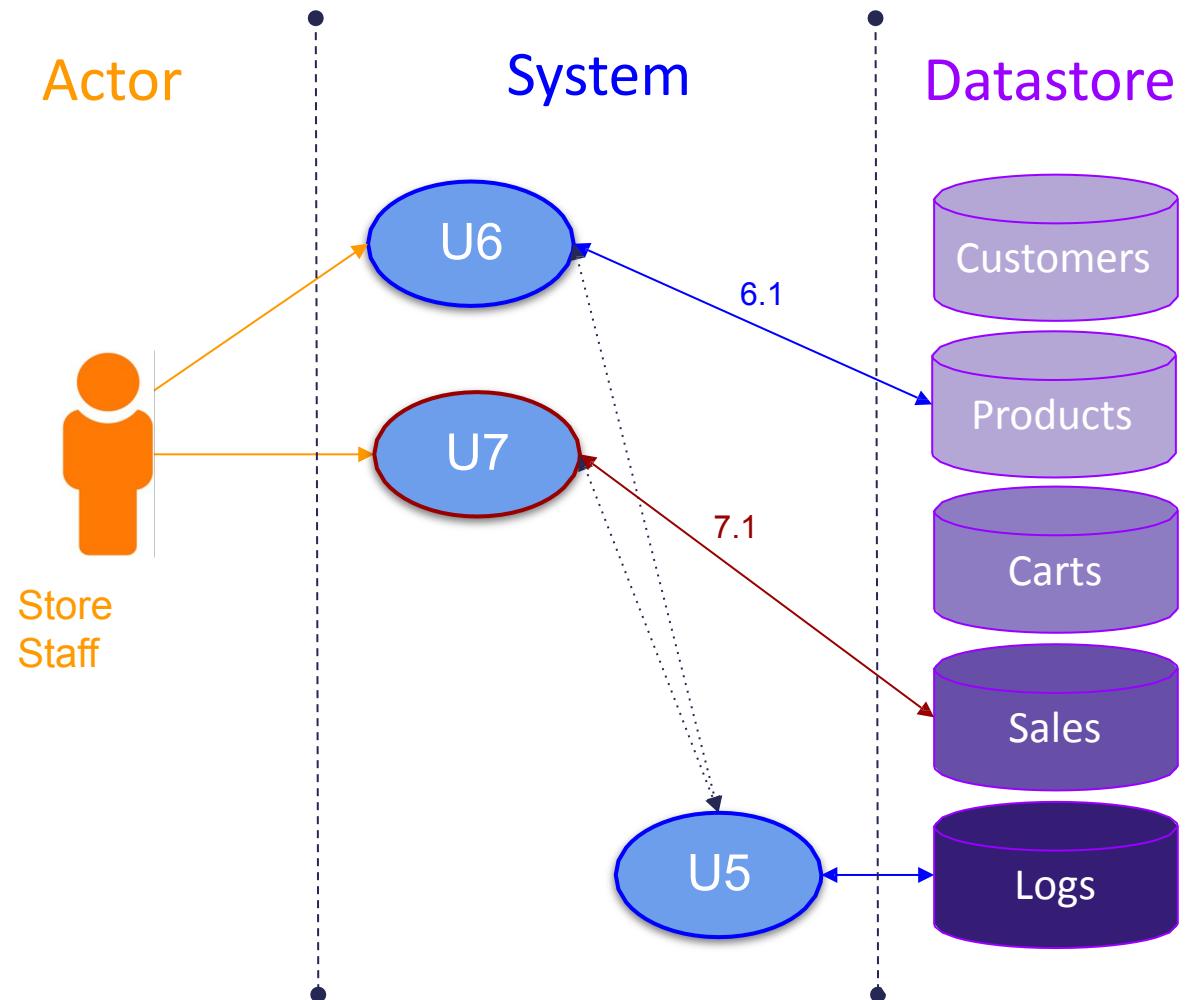
Customer Use Cases & Associated Datastore

Use Cases		Action
U1	Log in	1. Authentication & get customer info. 2. get Cart information 3. get latest price 4. Activate U5 with action='U1'
U2	Find Product	1. Get products by keywords 2. Activate U5 with action='U2'
U3	Update Shopping Cart	3.1(1) add a product to cart 3.1(2) remove products from cart 3.1(3) update quantity 3.2 Activate U5 with action='U3(n)'
U4	Checkout	1. Retrieve cart information 2. Retrieve latest price 3. Add Sales information 4. Remove purchased products from cart 5. Activate U5 with action='U4'
U5	Keep log	Insert action into logs datastore



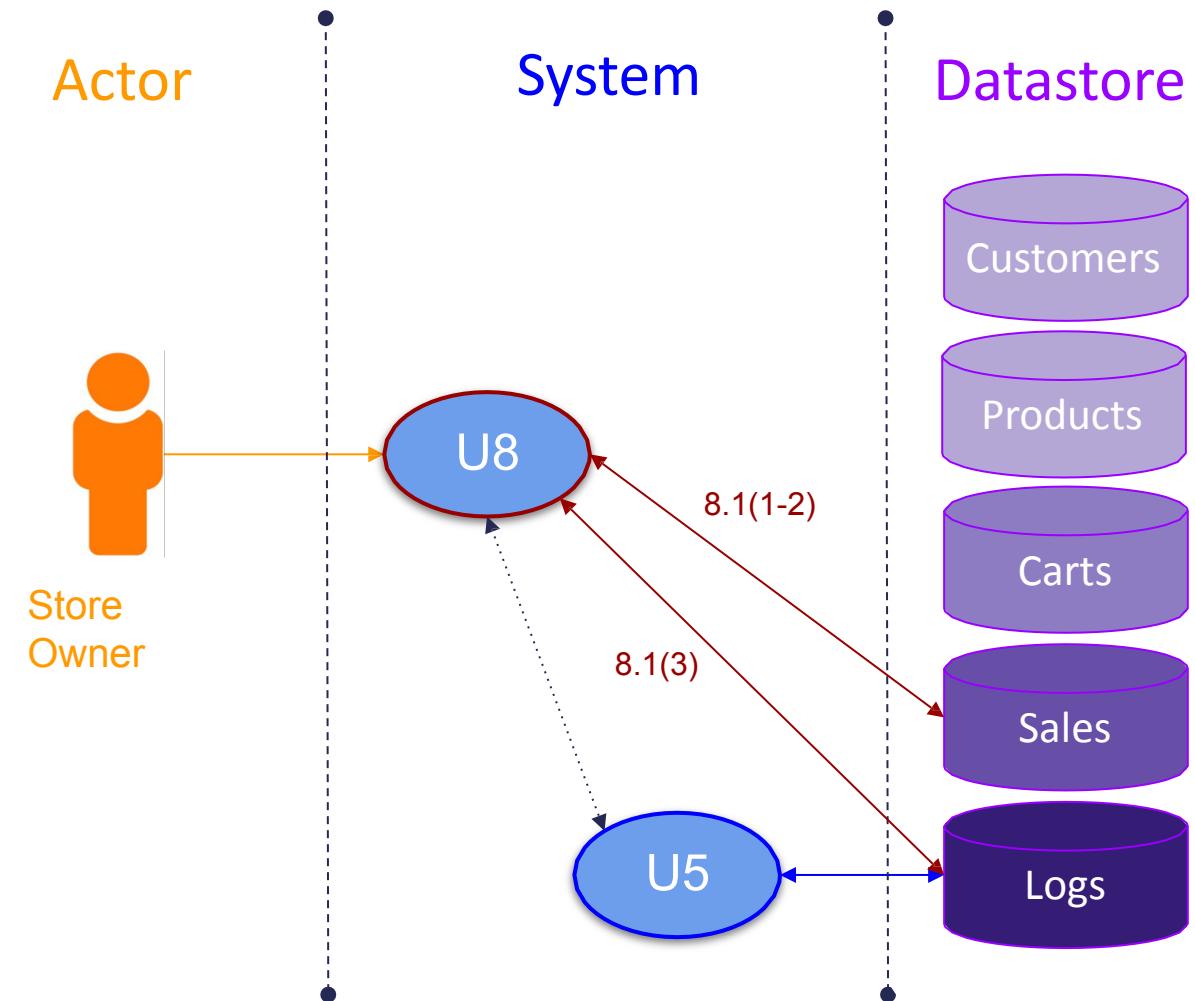
Store Staff Use Cases & Associated Datastore

Use Cases		Action
U6	Update Product and sale information	6.1(1) add a product 6.1(2) update bestseller for products 6.1(3) delete last-5year-sale document from sales 6.2 Activate U5 with action='U6'
U7	View Daily Report	For a given date 7.1(1) Get count sales for a given date 7.1(2) Get Sum total sales 7.1(3) The most/least purchased product 7.1(4) The average purchase cost per customer 7.2 Activate U5 with action='U7'
U5	Keep log	Insert action into logs datastore



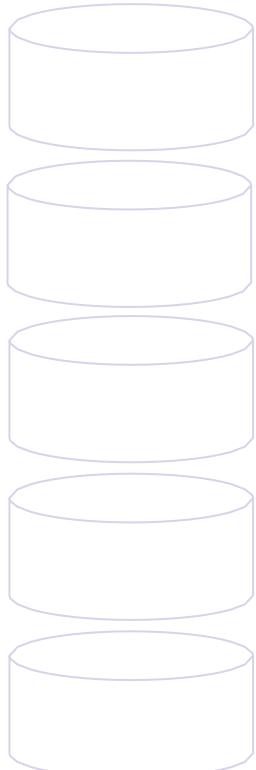
Store Owner Use Cases & Associated Datastore

Use Cases		Action
U8	View Summary Report	<p>Group by month, store type (i.e.,Online, Physical store)</p> <p>8.1(1) Get the most popular product</p> <p>8.1(2) Get total sales, count, average sales</p> <p>8.2 Activate U5 with action='U8'</p>
U5	Keep log	Insert action into logs datastore



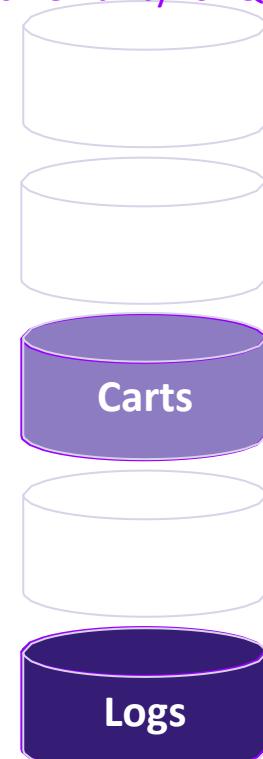
Data Modeling

Our data



Key-value model

- + Higher Speed Read and Write
- Less flexibility for Query



Document model

- Lower Speed Read and Write
- Flexibility for Query





mongoDB

Getting Started

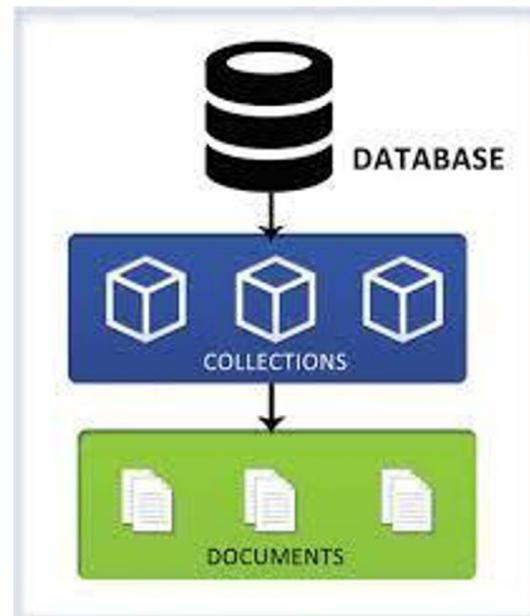
WHAT AND WHY



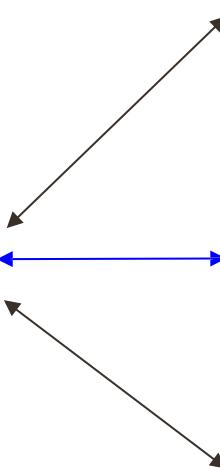
-
- Stores data in **JSON-like documents**.
 - **Fields can vary** from document to document and data structure can be changed over time
 - Distributed database, high availability, horizontal scaling, and geographic distribution
 - Scalability
 - Performance Scale: Sustaining 100,000+ database read and writes per second while maintaining strict latency SLAs
 - Data Scale: Storing 1 billion+ documents in the database
 - Cluster Scale: Distributing the database across 100+ nodes, in multiple data centers

MongoDB Architecture

MongoDB Atlas Cloud Database



MongoDB Clients



Connect with the mongo shell

Interact with your cluster using MongoDB's interactive Javascript interface



Connect using MongoDB Compass

Explore, modify, and visualize your data with MongoDB's GUI



Connect your application

Connect your application to your cluster using MongoDB's native drivers

MongoDB Atlas

The screenshot shows the MongoDB Atlas interface for the 'TickTock' database. The top navigation bar includes 'AIT > DMM2022 > DATABASES', the database name 'TickTock', version '5.0.12', and region 'AWS Singapore (ap-southeast-1)'. The main menu tabs are Overview, Real Time, Metrics, Collections (which is selected), Search, Profiler, Performance Advisor, Online Archive, and Cmd Line Tools. A sidebar on the left lists databases ('TickTock') and collections ('order', 'product', 'user'). The 'Collections' tab displays the 'TickTock.order' collection with storage details: STORAGE SIZE: 28KB, LOGICAL DATA SIZE: 42.21KB, TOTAL DOCUMENTS: 200, INDEXES TOTAL SIZE: 24KB. It features tabs for Find, Indexes, Schema Anti-Patterns (0), Aggregation, and Search Indexes (1). A 'FILTER' button with the query '{ field: 'value' }' is present, along with 'OPTIONS', 'Apply', and 'Reset' buttons. Below this is a section titled 'QUERY RESULTS: 1-20 OF MANY' showing a single document snippet:

```
_id: ObjectId('632bee1ed21620a13ee8af23')
id: 1
> payment: Array
  status: "New"
  userId: 6
> productId: Array
```

Connecting to MongoDB Atlas

- Download and Install MongoDB Compass
<https://www.mongodb.com/try/download/compass>
- Create a new connection

```
mongodb+srv://root:admin@ticktock.et04iht.mongodb.net/test?authSource=admin&replicaSet=atlas-ct18hr-shard-0&readPreference=primary&ssl=true
```

MongoDB Compass

The screenshot shows the MongoDB Compass interface. On the left, there's a sidebar with the connection details "ticktock.et04iht.mongodb.net", a "HOSTS" section listing shard hosts, a "CLUSTER" section showing a Replica Set with 3 Nodes, and sections for "EDITION" (MongoDB 5.0.12 Enterprise) and "My Queries". Below these are tabs for "TickTock" and "MongoShell". The main area is titled "Collections" and lists three collections: "order", "product", and "user". Each collection card provides statistics: Storage size, Documents count, Avg. document size, Indexes count, and Total index size. The "MongoShell" tab at the bottom is highlighted with an orange box and an arrow pointing to it from the right.

Collection	Storage size	Documents	Avg. document size	Indexes	Total index size
order	28.67 kB	200	216.00 B	1	24.58 kB
product	20.48 kB	50	236.00 B	1	20.48 kB
user	20.48 kB	15	141.00 B	1	20.48 kB

>_MONGOSH

Atlas atlas-ct18hr-shard-0 [primary] test >

MongoShell

MongoDB Practice

- Managing Product Information (U6)
 - INSERT
 - UPDATE
 - DELETE
- Searching Products (U2)
 - Basic Query

Insert Documents

[`db.collection.insertMany\(\)`](#)

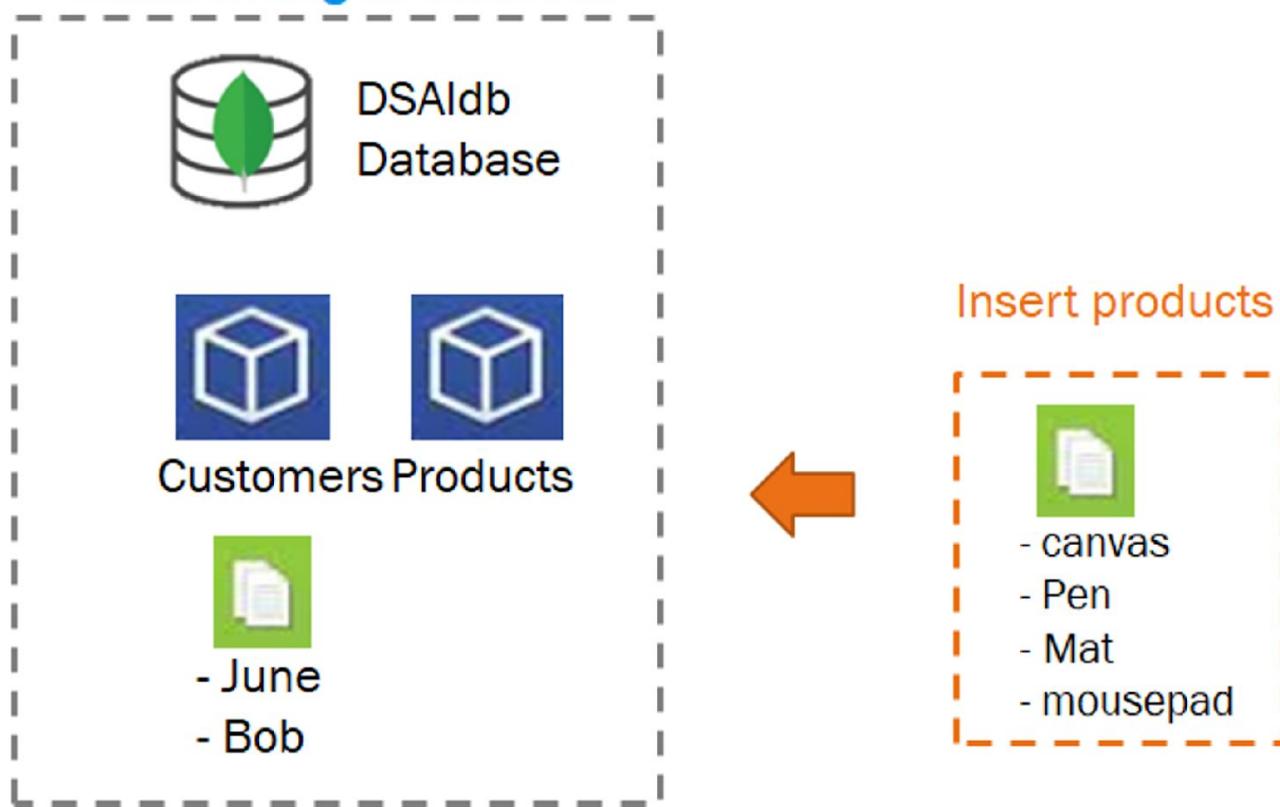
Syntax:

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

Document Model Architecture



Atlas MongoDB Server



Insert product documents

```
// Insert only one document
db.product.insertOne(
    { name: "woodplate", quantity: 100, price: 500, tags: ["wood", "school"] }
)
// Insert multiple documents
db.product.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"]
}
])
```

Update product(s) add “isBestSeller” field

SQL Query	MongoDB command
Alter table products Add isBestSeller boolean + Update products set isBestSeller = FALSE	<i>Schemaless, no need to define. If defined,</i> db.products.updateMany({}, {\$set: {isBestSeller: false}})
Update products set isBestSeller = TRUE Where _id = “603b60852014074d58e9d927”	db.products.updateMany({_id: ObjectId(“603b60852014074d58e9d927”)}, {\$set: {isBestSeller: true}})
	<i>To remove bestseller</i> 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 }
```

\$unset

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

Find Confirmed Order from Order

SQL Query	MongoDB command
Select from Order Where status = 'Confirmed'	db.order.find ({status:"Confirmed"})

```
> db.order.find({status:"Confirmed"})
< { _id: ObjectId("632bee1ed21620a13ee8af24"),
  id: 2,
  payment:
    [ { paymentMethod: 'Cash On Delivery',
        paymentStatus: 'Completed',
        paymentId: 'Pending',
        total: '$5806.13' } ],
  status: 'Confirmed',
  userId: 9,
  productId: [ 13, 89, 81 ] }
{ _id: ObjectId("632bee1ed21620a13ee8af25"),
  id: 3,
  payment:
    [ { paymentMethod: 'QR',
        paymentStatus: 'Pending',
        paymentId: 'Completed',
        total: '$6186.82' } ],
  status: 'Confirmed',
```

Delete Confirmed Order from Order

```
db.Order.deleteMany(  
  {status: "Confirmed"})
```

Your Turn

1. Add your own 4 products using `insertMany()`
2. Try to update your product information by adding a `quantity` field. Define different quantity for each product.

MongoDB Practice

- Managing Product Information (U6)
- Searching Products (U2)
 - Basic Query

Basic Query

SQL statements	MongoDB statements
<code>SELECT * FROM Product</code>	<code>db.product.find([<query>, <projection>]);</code>

View all products

```
db.product.find({ })
```

```
> db.product.find({})
< { _id: ObjectId("632d29d89c05f50cc5b24164"),
  id: 1,
  name: 'Leather Sofa Set 6 Seater',
  status: 'Available',
  description:
    [ { countrOfOrigin: 'Japan',
        material: 'Oak Wood',
        dimension: [ { height: 11, width: 4, weight: 12049 } ] },
      price: 615.82 },
  { _id: ObjectId("632d29d89c05f50cc5b24165"),
    id: 2,
    name: 'Dining Table 4 Seater',
    status: 'Available',
    description:
      [ { countrOfOrigin: 'Srilanka',
          material: 'Cashmire Wood',
          dimension: [ { height: 14, width: 2, weight: 6324 } ] },
        price: 921.23 }
```

Order By

SELECT * FROM products ORDER BY name	db.product.find({}).sort({name: 1 })
SELECT * FROM products ORDER BY name DESC	db.product.find({}).sort({name: -1 })

Find()

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

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

◎ **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

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

- 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.

Projection

Fields to return in the resulting data. 0 = Not Show, 1 = Show

```
> db.product.find({}, {price:0, userId: 0})
< { _id: ObjectId("632bf1ded21620a13ee8aff3"),
  id: 1,
  name: 'Leather Sofa Set 6 Seater',
  status: 'Available',
  description:
    [ { countrOfOrigin: 'Japan',
        material: 'Oak Wood',
        dimension: [ { height: 11, width: 4, weight: 12049 } ] } ]
{ _id: ObjectId("632bf1ded21620a13ee8aff4"),
  id: 2,
  name: 'Dining Table 4 Seater',
  status: 'Available',
  description:
    [ { countrOfOrigin: 'Srilanka',
        material: 'Cashmire Wood',
        dimension: [ { height: 14, width: 2, weight: 6324 } ] } ] }
```

Query

```
db.product.find({name: "Wooden Chair"})
```

```
> db.product.find({name: "Wooden Chair"})
< { _id: ObjectId("632bf1ded21620a13ee8affe"),
  id: 12,
  name: 'Wooden Chair',
  status: 'Available',
  description:
    [ { countr0f0rigin: 'Mongolia',
        material: 'Synthetic',
        dimension: [ { height: 10, width: 8, weight: 2689 } ] },
      price: '$839.36' },
  { _id: ObjectId("632bf1ded21620a13ee8b005"),
    id: 19,
    name: 'Wooden Chair',
    status: 'Unavailable',
    description:
      [ { countr0f0rigin: 'Bangladesh',
          material: 'Synthetic',
          dimension: [ { height: 4, width: 3, weight: 5517 } ] } ],
    ... ] }
```

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.

Query Operators : Example

```
$ne : not equal
db.product.find(
{
  name: {$ne: "Wooden
  Chair"}
}
);
```

```
> db.product.find({name: {$ne: "Wooden Chair"}});
< { _id: ObjectId("632bf1ded21620a13ee8aff3"),
  id: 1,
  name: 'Leather Sofa Set 6 Seater',
  status: 'Available',
  description:
    [ { countr0f0rigin: 'Japan',
      material: 'Oak Wood',
      dimension: [ { height: 11, width: 4, weight: 12049 } ] },
      price: '$615.82' }
{ _id: ObjectId("632bf1ded21620a13ee8aff4"),
  id: 2,
  name: 'Dining Table 4 Seater',
  status: 'Available',
  description:
    [ { countr0f0rigin: 'Srilanka',
      material: 'Cashmire Wood',
      dimension: [ { height: 14, width: 2, weight: 6324 } ] },
      price: '$921.23' }
```

Query Operators : Example

```
$lt : less than  
db.order.find  
({userId:  
{$lt: 5}  
})
```

```
< { _id: ObjectId("632bee1ed21620a13ee8af2f"),  
    id: 13,  
    payment:  
      [ { paymentMethod: 'Credit Card',  
          paymentStatus: 'Completed',  
          paymentId: 'Completed',  
          total: '$9695.96' } ],  
    status: 'Confirmed',  
    userId: 4,  
    productId: [ 12 ] }  
{ _id: ObjectId("632bee1ed21620a13ee8af38"),  
  id: 22,  
  payment:  
    [ { paymentMethod: 'Credit Card',  
        paymentStatus: 'Completed',  
        paymentId: 'Completed',  
        total: '$9454.54' } ],  
  status: 'Delivered',  
  userId: 4,  
  productId: [ 26, 16, 29 ] }  
{ _id: ObjectId("632bee1ed21620a13ee8af3a"),
```

Querying with RegEx

```
db.product.find
  ({
    name: {
      $regex:
        /Chair/
    }
});
```

```
> db.product.find({name: {$regex: /Chair/}});
< { _id: ObjectId("632bf1ded21620a13ee8aff5"),
  id: 3,
  name: 'Rocking Chair',
  status: 'Available',
  description:
    [ { countrOfOrigin: 'Nepal',
      material: 'Synthetic',
      dimension: [ { height: 13, width: 6, weight: 5048 } ] } ],
  price: '$889.70' }

{ _id: ObjectId("632bf1ded21620a13ee8aff6"),
  id: 4,
  name: 'Rocking Chair',
  status: 'Available',
  description:
    [ { countrOfOrigin: 'Thailand',
      material: 'Synthetic',
      dimension: [ { height: 7, width: 8, weight: 7184 } ] } ],
  price: '$735.67' }

{ _id: ObjectId("632bf1ded21620a13ee8aff7"),
  id: 5,
  name: 'Office Wooden Chair',
```

IN

Search for Order status for Order that have product Id 20 or 22

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER `{productId: {$in: [20, 22]}}` **OPTIONS** **FIND** **RESET** **...**

PROJECT `{productId: 1, "payment.paymentStatus": 1}`

SORT `{ field: -1 } or [[field', -1]]` **MAX TIME MS** 60000

COLLATION `{ locale: 'simple' }` **SKIP** 0 **LIMIT** 0

VIEW **grid** **refresh** **refresh** Displaying documents 1 - 17 of 17 **first** **last** **refresh**

```
_id: ObjectId('632bee1ed21620a13ee8af23')
payment: Array
  0: Object
    paymentStatus: "Pending"
productId: Array
  0: 20
  1: 22

_id: ObjectId('632bee1ed21620a13ee8af32')
payment: Array
  0: Object
    paymentStatus: "Completed"
productId: Array
  0: 14
  1: 20
  2: 99
```

EXISTS

Search products where size exists

MongoDB Studio																																																	
Documents	Aggregations	Schema	Explain Plan	Indexes	Validate																																												
<button> FILTER {size: {\$exists:true}}</button>	<button> OPTIONS </button>	<button> FIND </button>	<button> RESET </button>	<button> ... </button>																																													
<button> ADD DATA </button>	<button> VIEW </button>	<button> { } </button>	<button> { } </button>	Displaying documents 1 - 8 of 8	<button> < </button> <button> > </button> <button> REFRESH </button>																																												
<table border="1"><thead><tr><th>products</th><th>_id ObjectId</th><th>name String</th><th>description String</th><th>Actions</th></tr></thead><tbody><tr><td>1</td><td>6039cb812014074d58e9d920</td><td>"backpack"</td><td>"A drawstring style crafted fro</td><td> </td></tr><tr><td>2</td><td>603b4e6d2014074d58e9d923</td><td>"canvas"</td><td>No field</td><td> </td></tr><tr><td>3</td><td>603b4e6d2014074d58e9d924</td><td>"pencil2B"</td><td>No field</td><td> </td></tr><tr><td>4</td><td>603b4e6d2014074d58e9d925</td><td>"mat"</td><td>No field</td><td> </td></tr><tr><td>5</td><td>603b4e6d2014074d58e9d926</td><td>"mousepad"</td><td>No field</td><td> </td></tr><tr><td>6</td><td>603b60852014074d58e9d927</td><td>"erasablepen"</td><td>No field</td><td> </td></tr><tr><td>7</td><td>603b610d2014074d58e9d928</td><td>"whiteboardpen"</td><td>No field</td><td> </td></tr><tr><td>8</td><td>603b61342014074d58e9d929</td><td>"redpen"</td><td>No field</td><td> </td></tr></tbody></table>					products	_id ObjectId	name String	description String	Actions	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	
products	_id ObjectId	name String	description String	Actions																																													
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.product.find({  
    price: {  
        $gt: 500, $lte: 1000  
    }  
})
```

Result

TickTock.product

50 1
DOCUMENTS INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER {price: {\$gt:500, \$lte:1000}} OPTIONS FIND RESET ...

ADD DATA VIEW Displaying documents 1 - 20 of 50 < > C REFRESH

product	id Int32	name String	status String	description Array	price Double	Actions			
1 05f50cc...	1	"Leather Sofa Set 6 Seater"	"Available"	[] 1 elements	615.82				
2 05f50cc...	2	"Dining Table 4 Seater"	"Available"	[] 1 elements	921.23				
3 05f50cc...	3	"Rocking Chair"	"Available"	[] 1 elements	889.7				
4 05f50cc...	4	"Rocking Chair"	"Available"	[] 1 elements	735.67				
5 05f50cc...	5	"Office Wooden Chair"	"Available"	[] 1 elements	966.37				
6 05f50cc...	6	"Office Chair with Wheels"	"Available"	[] 1 elements	698.06				
7 05f50cc...	7	"Dining Table 4 Seater"	"Available"	[] 1 elements	682.11				
8 05f50cc...	8	"Leather Sofa Set 6 Seater"	"Available"	[] 1 elements	932.61				
9 05f50cc...	9	"Vase"	"Available"	[] 1 elements	828.37				
10 05f50cc...	10	"Office Wooden Chair"	"Unavailable"	[] 1 elements	610.47				
11 05f50cc...	11	"Dining Table 6 Seater"	"Unavailable"	[] 1 elements	716.71				

OR

Search product from Japan or Thailand

```
db.product.find({$or: [{"description.countryOfOrigin": "Japan"}, {"description.countryOfOrigin": "Thailand"}]})
```

Result

```
> db.product.find({$or: [{"description.countryOfOrigin": "Japan"}, {"description.countryOfOrigin": "Thailand"}]})  
< { _id: ObjectId("632d29d89c05f50cc5b24164"),  
    id: 1,  
    name: 'Leather Sofa Set 6 Seater',  
    status: 'Available',  
    description:  
      [ { countryOfOrigin: 'Japan',  
          material: 'Oak Wood',  
          dimension: [ { height: 11, width: 4, weight: 12049 } ] } ],  
    price: 615.82 }  
{ _id: ObjectId("632d29d89c05f50cc5b24167"),  
    id: 4,  
    name: 'Rocking Chair',  
    status: 'Available',  
    description:  
      [ { countryOfOrigin: 'Thailand',  
          material: 'Synthetic',  
          dimension: [ { height: 7, width: 8, weight: 7184 } ] } ],  
    price: 735.67 }
```

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

```
_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"
```

Nested
Field Ex.
size

Query Nested Field

Select all products where height dimension is exactly 11

```
db.product.find({"description.dimension.height":11})
```

TickTock.product 50 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER `{"description.dimension.height":11}` ▼ OPTIONS

PROJECT `{ field: 0 }`

SORT `{ field: -1 } or [[('field', -1]]` MAX TIME MS 60000

COLLATION `{ locale: 'simple' }` SKIP 0 LIMIT 0

FIND **RESET** **...**

ADD DATA VIEW {} [] Displaying documents 1 - 2 of 2 < > REFRESH

```
_id: ObjectId('632d29d89c05f50cc5b24164')
id: 1
name: "Leather Sofa Set 6 Seater"
status: "Available"
description: Array
  0: Object
    countrOfOrigin: "Japan"
    material: "Oak Wood"
    dimension: Array
      0: Object
        height: 11
        width: 4
        weight: 12049
price: 615.82
```

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' } }
```

The screenshot shows the MongoDB Compass interface with the 'Documents' tab selected. A filter bar at the top has the query '{tags:"kids"}' applied. Below the filter are buttons for 'ADD DATA', 'VIEW', and document navigation icons. To the right, it says 'Displaying docu'. The main area shows a single document result:

```
_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"
```

LookUp & Aggregate Queries

TickTock.order 200 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Pipeline \$lookup \$match Explain Export Run More Options ▾

Untitled – modified SAVE + CREATE NEW EXPORT TO LANGUAGE AUTO PREVIEW

\$lookup Output after \$lookup stage (Sample of 10 documents)

```
1 /**
2  * from: The target collection.
3  * localField: The local join field.
4  * foreignField: The target join field.
5  * as: The name for the results.
6  * pipeline: Optional pipeline to run on the foreign
7  * let: Optional variables to use in the pipeline
8 */
9 {
10  from: "user",
11  localField: "userId",
12  foreignField: "userId",
13  as: "userKey"
14 }
```

_id: ObjectId('632d44a8db2c6cf4f0993f0e')
id: 1
> payment: Array
status: "New"
userId: 6
> productId: Array
> userKey: Array
 0: Object
 _id: ObjectId('632be522d21620a13ee8af15'
 userId: 6
 name: "Mia Lynn"

_id: ObjectId('632d44a8db2c6cf4f0993f0f')
id: 2
> payment: Array
status: "Confirmed"
userId: 9
> productId: Array
> userKey: Array

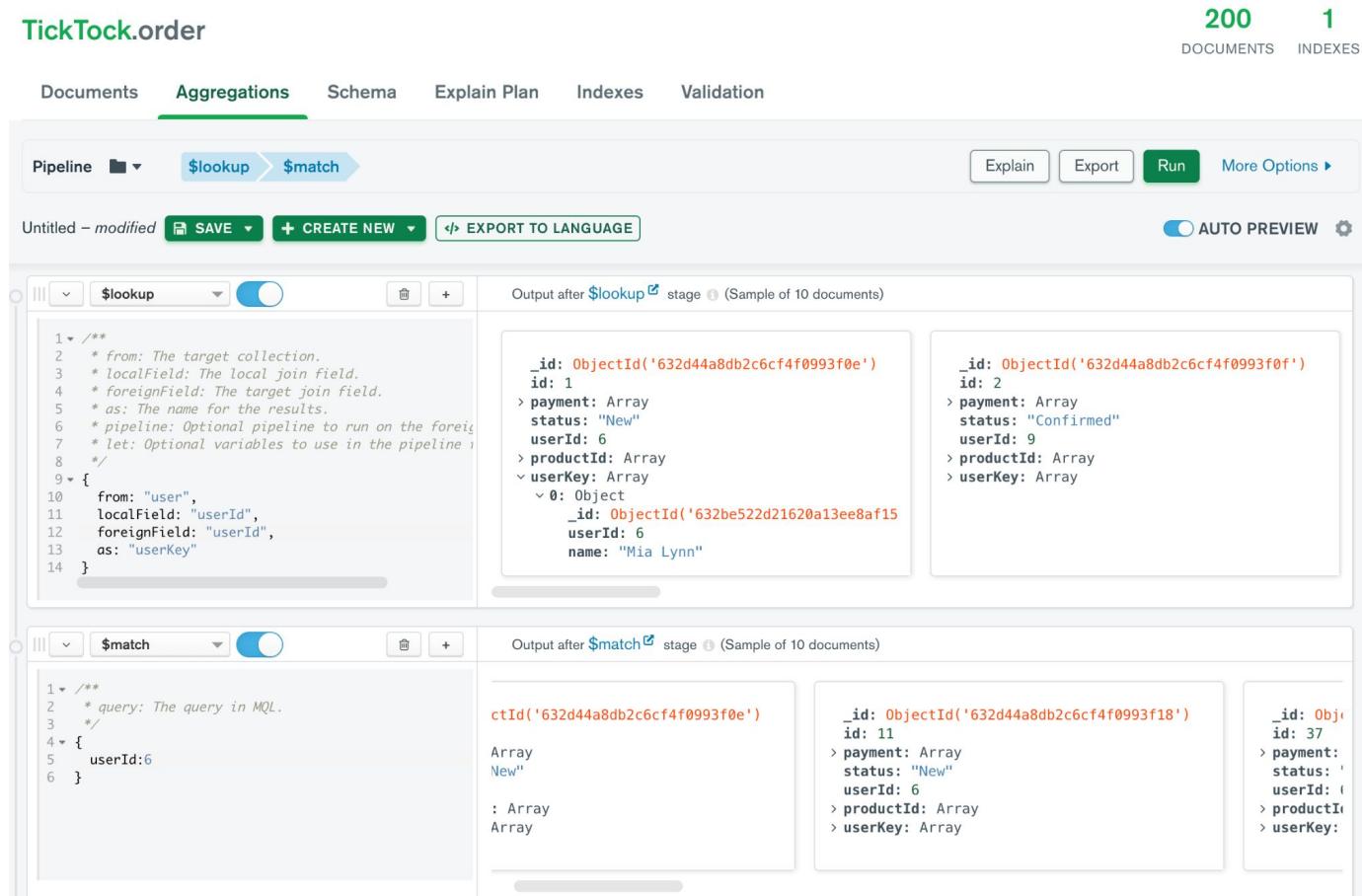
\$match Output after \$match stage (Sample of 10 documents)

```
1 /**
2  * query: The query in MQL.
3  */
4 {
5  userId: 6
6 }
```

ctId('632d44a8db2c6cf4f0993f0e')
Array
New"
: Array
Array

_id: ObjectId('632d44a8db2c6cf4f0993f18')
id: 11
> payment: Array
status: "New"
userId: 6
> productId: Array
> userKey: Array

_id: Obj...
id: 37
> payment:
status: '
userId: '
> productId:
> userKey:



LookUp & Aggregate Queries

List the count of product on the basis of their status.

```
db.product.aggregate([
  {
    $group: {
      _id: "$status",
      count: {
        $sum: 1
      },
      totalValue: {
        $sum: '$price'
      }
    }
  }
]);
```

TickTock.product 50 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

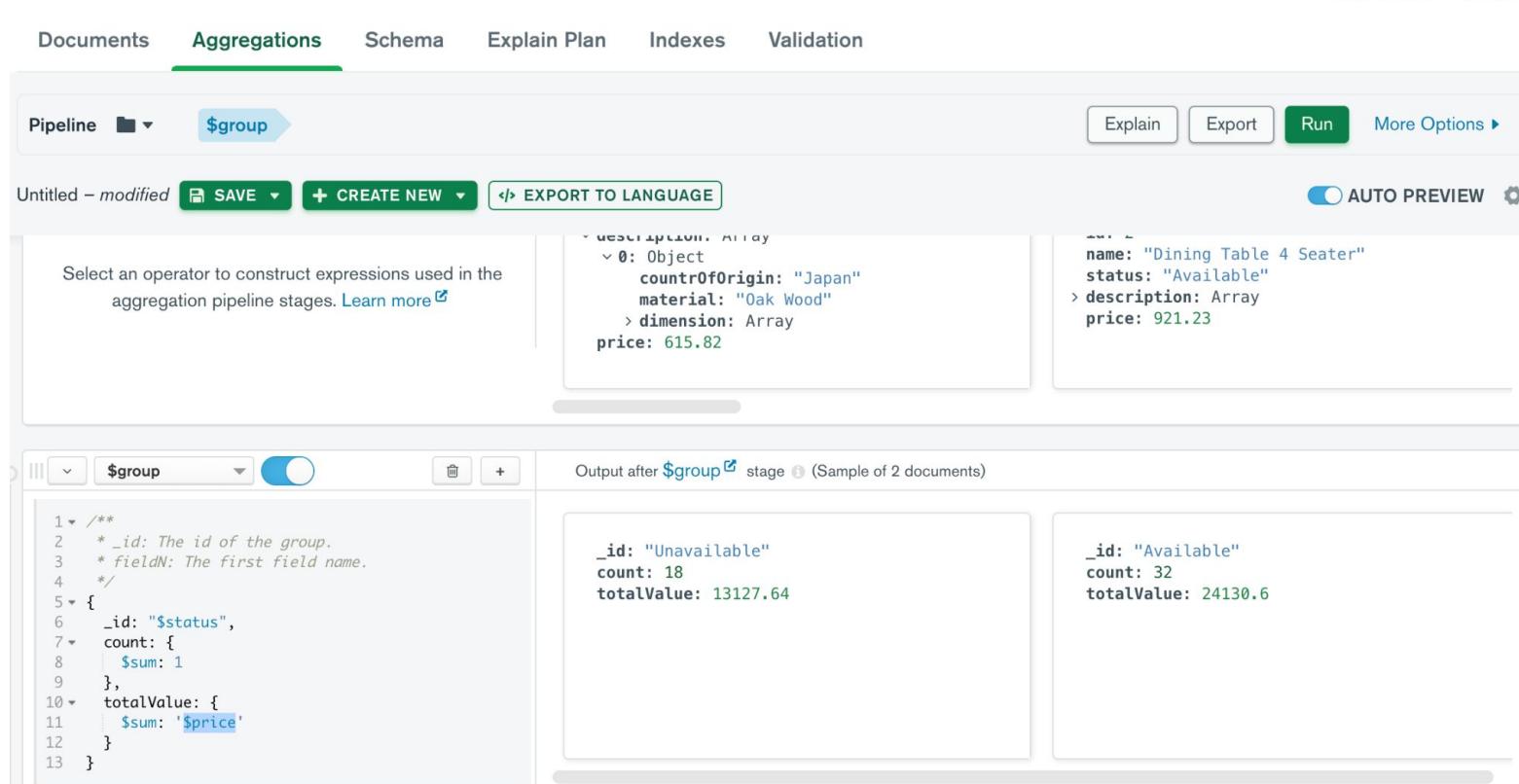
Pipeline \$group Explain Export Run More Options ▾

Untitled – modified SAVE + CREATE NEW EXPORT TO LANGUAGE AUTO PREVIEW

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

Output after \$group stage (Sample of 2 documents)

_id	count	totalValue
Unavailable	18	13127.64
Available	32	24130.6



Your Turn

1. Show all products sorted by quantity in descending order.
2. Show all products with a specific tag, ordered by price ascendingly.
3. Select products where their height less than 15 cm and the UOM is in cm.

Additional References

1. Create your own MongoDB server on Atlas
 - <https://docs.atlas.mongodb.com/getting-started/>
 - <https://www.youtube.com/watch?v=S4fi6Qux-4g>
2. Using MongoDB plugin on VS Code
 - <https://docs.mongodb.com/mongodb-vscode/install/>
 - <https://www.youtube.com/watch?v=g1EjixJht6g&t=424s>
3. Using MongoDB in Python
 - <https://docs.mongodb.com/drivers/python/>
 - https://www.youtube.com/watch?v=rE_bJI2GAY8

The background features a complex, abstract pattern composed of black and white squares. Some squares are standard checkered, while others are distorted into various wavy, curved, and twisted shapes, creating a sense of depth and motion.

Thank you.
