Practical 5: MongoDB aggregation operations

The operations on each stage can be one of the following:

- $project – select fields for the output documents.
- $match – select documents to be processed.
- $limit – limit the number of documents to be passed to the next stage.
- $skip – skip a specified number of documents.
- $sort – sort documents.
- $group – group documents by a specified key.

The following shows the syntax for defining an aggregation pipeline:

db.collection.aggregate([{ $match:...},{$group:...},{$sort:...}]);

In this syntax:

- First, call the aggregate() method on the collection.
- Second, pass an array of documents, where each document describes a stage in the pipeline.

MongoDB aggregation example

First, switch to the coffeeshop database that stores the coffee sales:

– use coffeeshop

Second, insert documents into the sales collection:

– db.sales.insertMany([

{ "_id" : 1, "item" : "Americanos", "price" : 5, "size": "Short", "quantity" : 22, "date" : ISODate("2022-01-15T08:00:00Z") },

{ "_id" : 2, "item" : "Cappuccino", "price" : 6, "size": "Short","quantity" : 12, "date" : ISODate("2022-01-16T09:00:00Z") },

{ "_id" : 3, "item" : "Lattes", "price" : 15, "size": "Grande","quantity" : 25, "date" : ISODate("2022-01-16T09:05:00Z") },

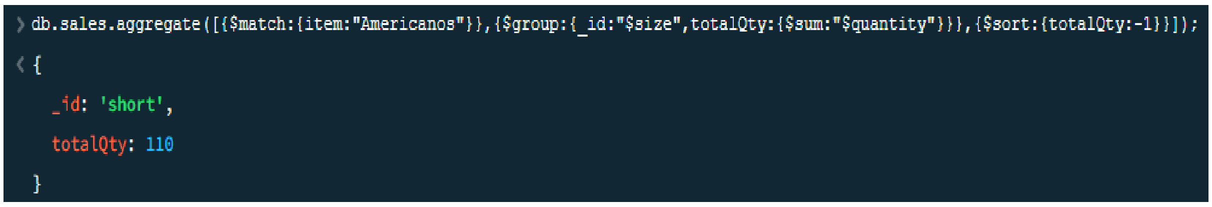
{ "_id" : 4, "item" : "Mochas", "price" : 25,"size": "Tall", "quantity" : 11, "date" : ISODate("2022-02-17T08:00:00Z") },

```
{ "_id" : 5, "item" : "Americanos", "price" : 10, "size": "Grande","quantity" :
12, "date" : ISODate("2022-02-18T21:06:00Z") },

{ "_id" : 6, "item" : "Cappuccino", "price" : 7, "size": "Tall","quantity" : 20,
"date" : ISODate("2022-02-20T10:07:00Z") },

{ "_id" : 7, "item" : "Lattes", "price" : 25,"size": "Tall", "quantity" : 30, "date" :
ISODate("2022-02-21T10:08:00Z") },

{ "_id" : 8, "item" : "Americanos", "price" : 10, "size": "Grande","quantity" :
21, "date" : ISODate("2022-02-22T14:09:00Z") },

{ "_id" : 9, "item" : "Cappuccino", "price" : 10, "size": "Grande","quantity" :
17, "date" : ISODate("2022-02-23T14:09:00Z") },

{ "_id" : 10, "item" : "Americanos", "price" : 8, "size": "Tall","quantity" : 15,
"date" : ISODate("2022-02-25T14:09:00Z")}

]);
```

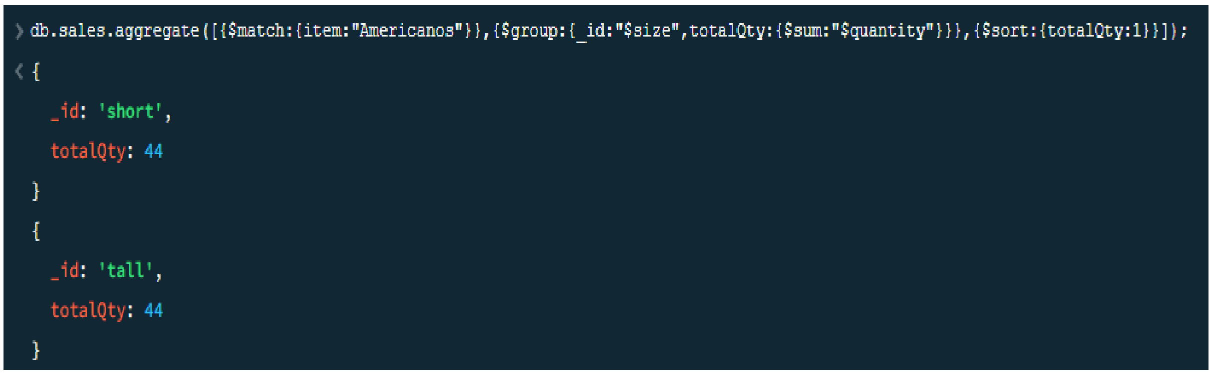
Third, use an aggregation pipeline to filter the sales by the Americanos, calculate the sum of quantity grouped by sizes, and sort the result document by the total quantity in descending order.

```
db.sales.aggregate([

        {

                $match: { item: "Americanos" }

        },

        {

                $group: {

                        _id: "$size",

                        totalQty: {$sum: "$quantity"}

                }

        },

        {

                $sort: { totalQty : -1}
```

```
          }
]);
[
  { _id: 'Grande', totalQty: 33 },
  { _id: 'Short', totalQty: 22 },
  { _id: 'Tall', totalQty: 15 }
]
```



```
> db.sales.aggregate([{$match:{item:"Americanos"}},{$group:{_id:"$size",totalQty:{$sum:"$quantity"}}},{$sort:{totalQty:-1}}]);
< {
    _id: 'short',
    totalQty: 110
  }
```



```
> db.sales.aggregate([{$match:{item:"Americanos"}},{$group:{_id:"$size",totalQty:{$sum:"$quantity"}}},{$sort:{totalQty:1}}]);
< {
    _id: 'short',
    totalQty: 44
  }
  {
    _id: 'tall',
    totalQty: 44
  }
```