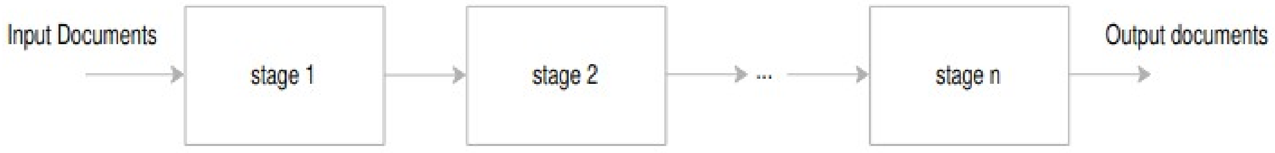
Labcycle 6
MongoDB aggregation operations

MongoDB aggregation operations allow us to process multiple documents and return the calculated results. Typically, we use aggregation operations to group documents by specific field values and perform aggregations on the grouped documents to return computed results. For example, we can use aggregation operations to take a list of sales orders and calculate the total sales amounts grouped by the products. To perform aggregation operations, we use aggregation pipelines. An aggregation pipeline contains one or more stages that process the input documents:



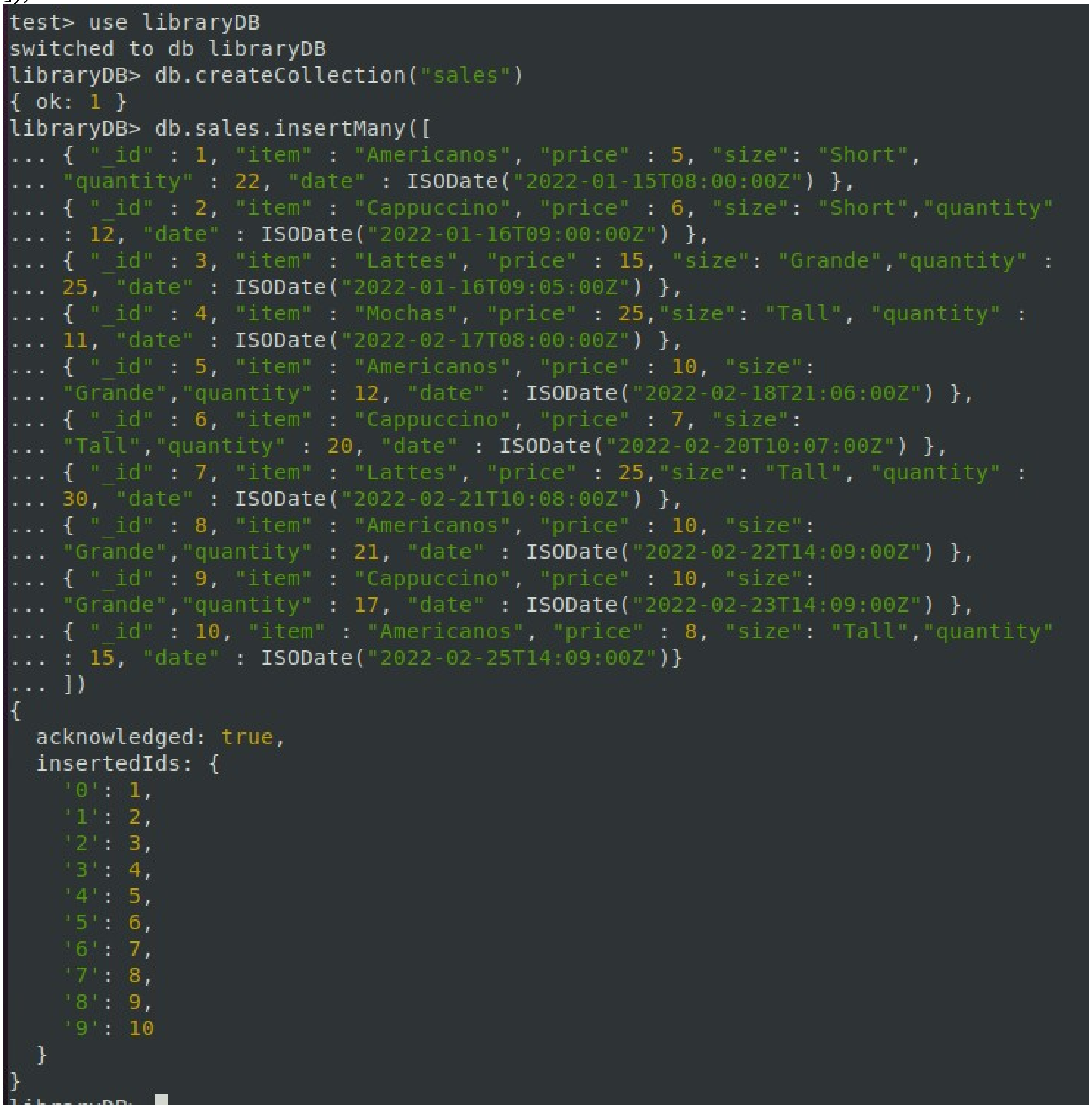
Each stage in the aggregation pipeline performs an operation on the input documents and returns the output documents. The output documents are then passed to the next stage. The final stage returns the calculated result.

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:...}]);
```
Use the following 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") },
```

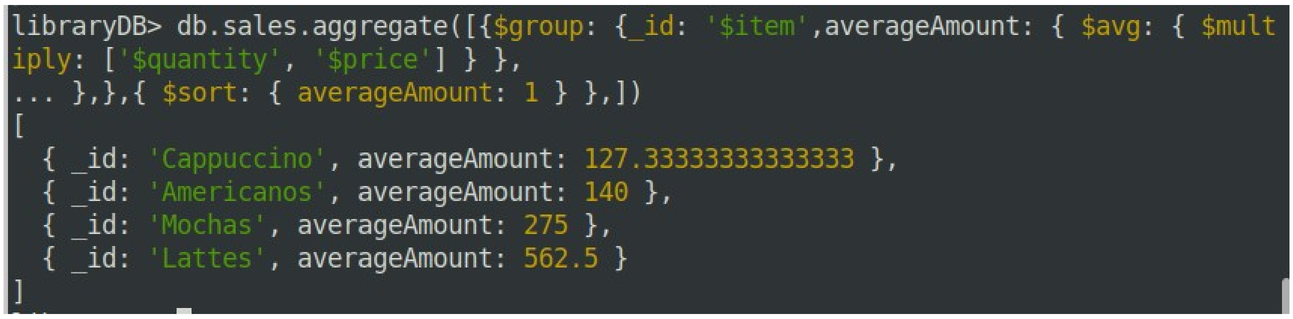
{ "_id" : 10, "item" : "Americanos", "price" : 8, "size": "Tall","quantity"
: 15, "date" : ISODate("2022-02-25T14:09:00Z")}
]);



```
test> use libraryDB
switched to db libraryDB
libraryDB> db.createCollection("sales")
{ ok: 1 }
libraryDB> 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")}
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': 1,
    '1': 2,
    '2': 3,
    '3': 4,
    '4': 5,
    '5': 6,
    '6': 7,
    '7': 8,
    '8': 9,
    '9': 10
  }
}
```

1. Groups the documents by the item field and use the $avg to calculate the average amount for each group.

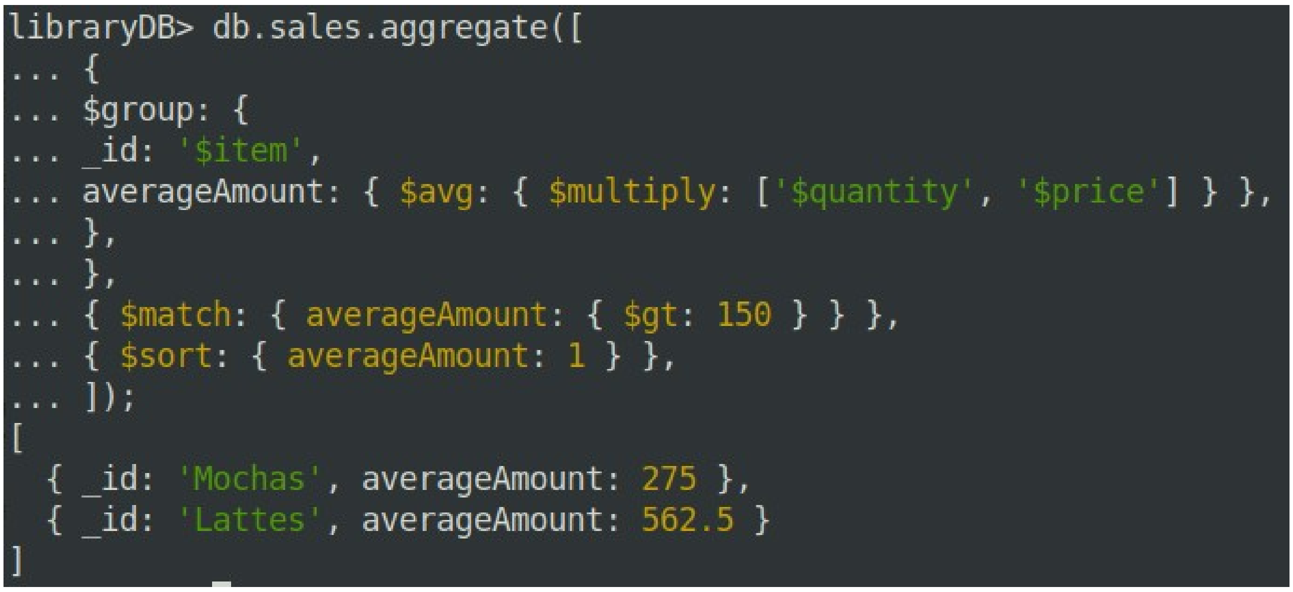
db.sales.aggregate([{$group: {_id: '$item',averageAmount: { $avg: { $multiply: ['$quantity', '$price'] } },},},{ $sort: { averageAmount: 1 } },])



```
libraryDB> db.sales.aggregate([{$group: {_id: '$item',averageAmount: { $avg: { $mult
iply: ['$quantity', '$price'] } },
... },},{ $sort: { averageAmount: 1 } },])
[
  { _id: 'Cappuccino', averageAmount: 127.33333333333333 },
  { _id: 'Americanos', averageAmount: 140 },
  { _id: 'Mochas', averageAmount: 275 },
  { _id: 'Lattes', averageAmount: 562.5 }
]
```

2.Calculate the average amount per group and returns the group with the average amount greater than 150.

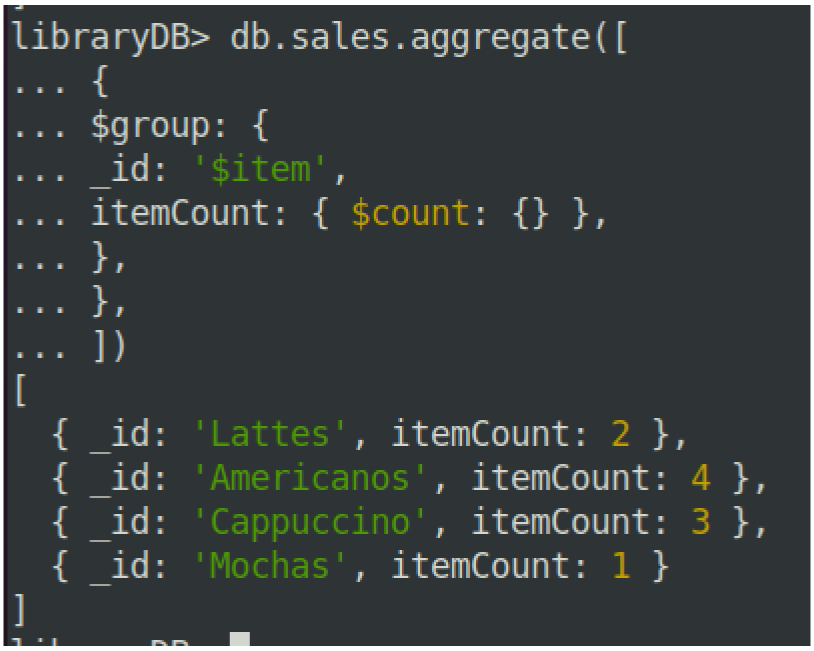
db.sales.aggregate([{$group: {_id: '$item',averageAmount: { $avg: { $multiply: ['$quantity', '$price'] } },},},{ $match: { averageAmount: { $gt: 150 } } },{ $sort: { averageAmount: 1 } },]);



```
libraryDB> db.sales.aggregate([
... {
... $group: {
... _id: '$item',
... averageAmount: { $avg: { $multiply: ['$quantity', '$price'] } },
... },
... },
... { $match: { averageAmount: { $gt: 150 } } },
... { $sort: { averageAmount: 1 } },
... ]);
[
  { _id: 'Mochas', averageAmount: 275 },
  { _id: 'Lattes', averageAmount: 562.5 }
]
```

3. Return the number of items in the sales collection.

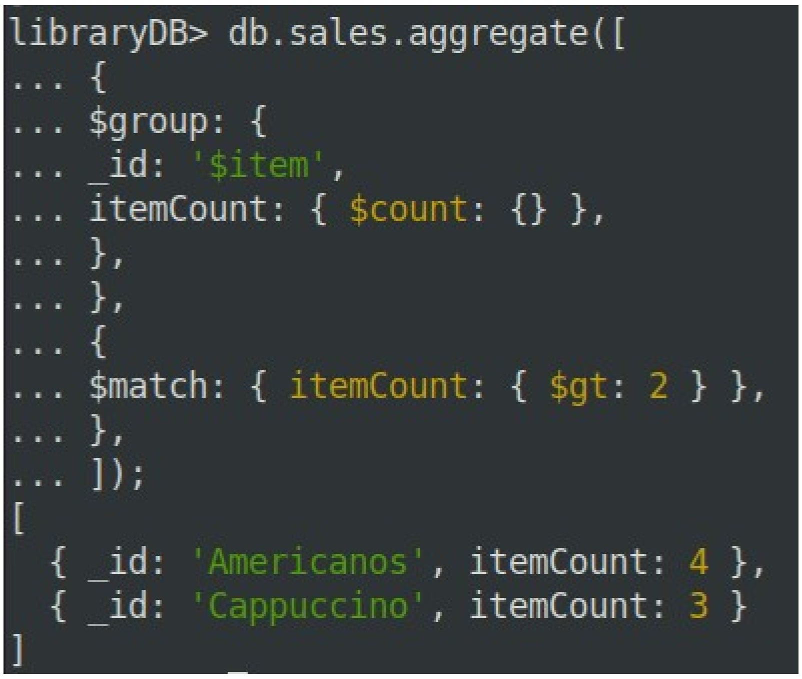
db.sales.aggregate([{$group: {_id: '$item',itemCount: { $count: {} },},},])



```
libraryDB> db.sales.aggregate([
... {
... $group: {
... _id: '$item',
... itemCount: { $count: {} },
... },
... },
... ])
[
  { _id: 'Lattes', itemCount: 2 },
  { _id: 'Americanos', itemCount: 4 },
  { _id: 'Cappuccino', itemCount: 3 },
  { _id: 'Mochas', itemCount: 1 }
]
```

4. Calculate the number of documents per item and returns the item with a count greater than two.

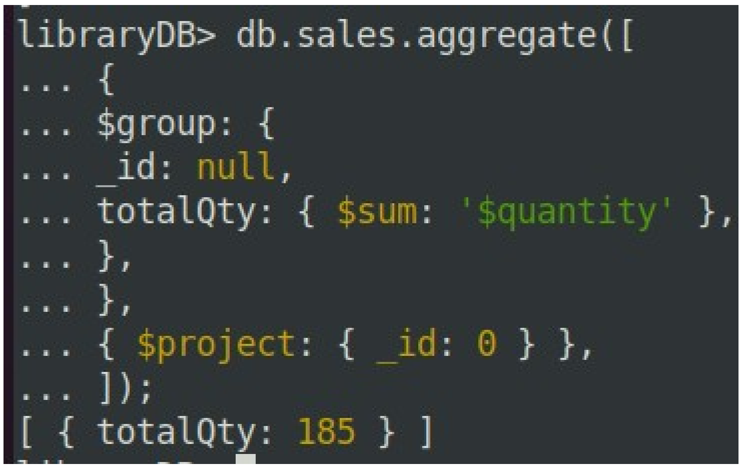
db.sales.aggregate([{$group: {_id: '$item',itemCount: { $count: {} },},},{
$match: { itemCount: { $gt: 2 } },},]);



```
libraryDB> db.sales.aggregate([
... {
... $group: {
... _id: '$item',
... itemCount: { $count: {} },
... },
... },
... {
... $match: { itemCount: { $gt: 2 } },
... },
... ]);
[
  { _id: 'Americanos', itemCount: 4 },
  { _id: 'Cappuccino', itemCount: 3 }
]
```

5. Calculates the total quantity of coffee sales in the sales collection.

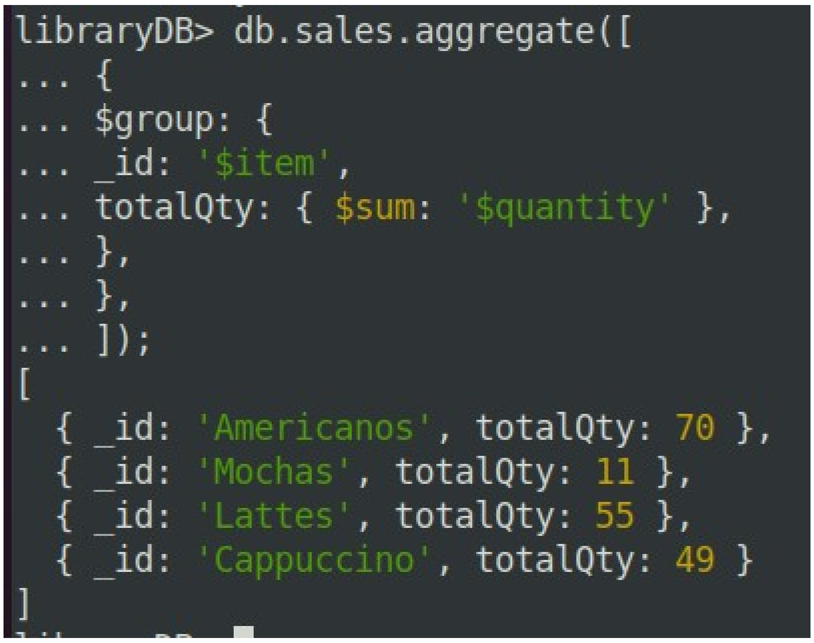
db.sales.aggregate([{$group: {_id: null,totalQty: { $sum: '$quantity' },},},{ $project: { _id: 0 } },])



```
libraryDB> db.sales.aggregate([
... {
... $group: {
... _id: null,
... totalQty: { $sum: '$quantity' },
... },
... },
... { $project: { _id: 0 } },
... ]);
[ { totalQty: 185 } ]
```

6. Calculate the sum of quantity grouped by items.

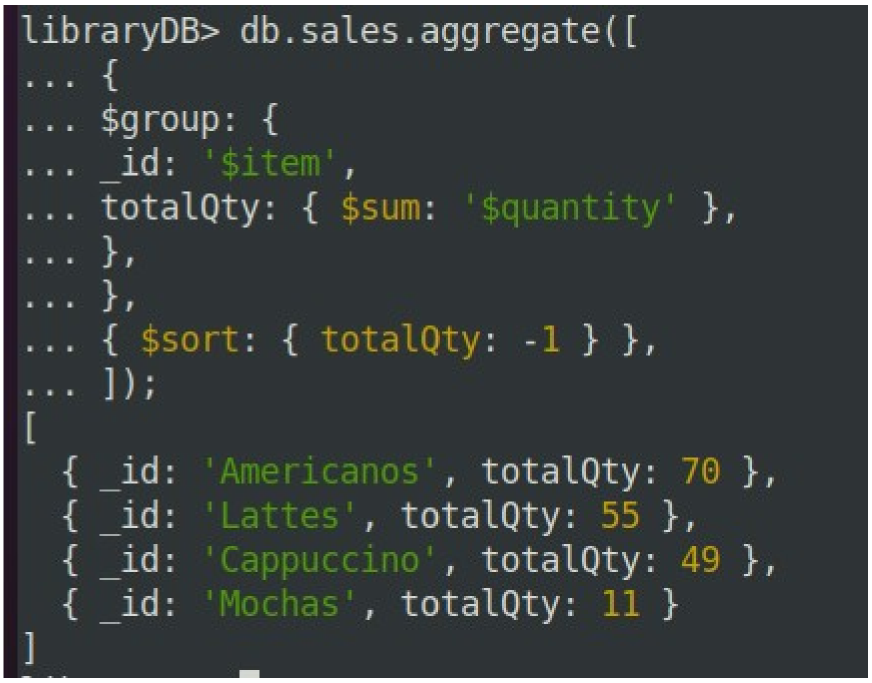
db.sales.aggregate([{$group: {_id: '$item',totalQty: { $sum: '$quantity' },},},]);



```
libraryDB> db.sales.aggregate([
... {
... $group: {
... _id: '$item',
... totalQty: { $sum: '$quantity' },
... },
... },
... ]);
[
  { _id: 'Americanos', totalQty: 70 },
  { _id: 'Mochas', totalQty: 11 },
  { _id: 'Lattes', totalQty: 55 },
  { _id: 'Cappuccino', totalQty: 49 }
]
```

7.Returns the total quantity of each item and sorts the result documents by the totalQty in descending order.

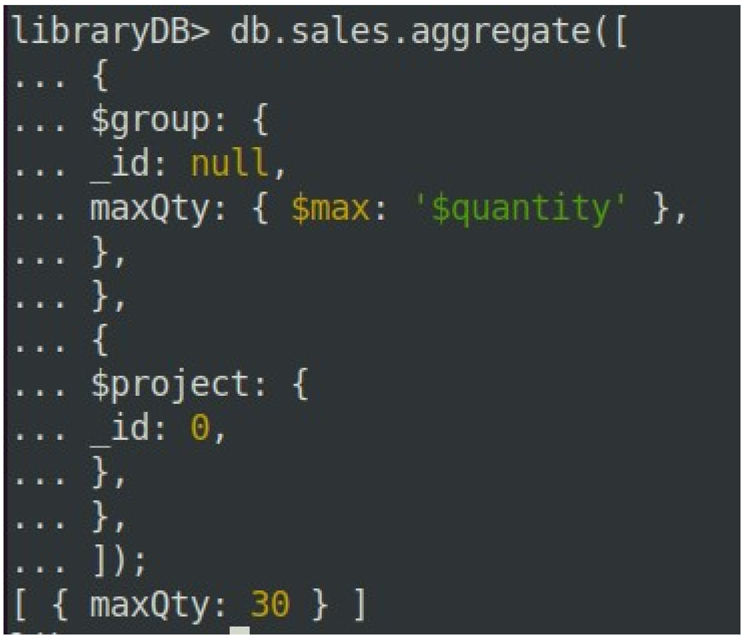
db.sales.aggregate([{$group: {_id: '$item',totalQty: { $sum: '$quantity' },},},
{ $sort: { totalQty: -1 } },]);



```
libraryDB> db.sales.aggregate([
... {
... $group: {
... _id: '$item',
... totalQty: { $sum: '$quantity' },
... },
... },
... { $sort: { totalQty: -1 } },
... ]);
[
  { _id: 'Americanos', totalQty: 70 },
  { _id: 'Lattes', totalQty: 55 },
  { _id: 'Cappuccino', totalQty: 49 },
  { _id: 'Mochas', totalQty: 11 }
]
```

8. Find the maximum quantity from all the sales documents.

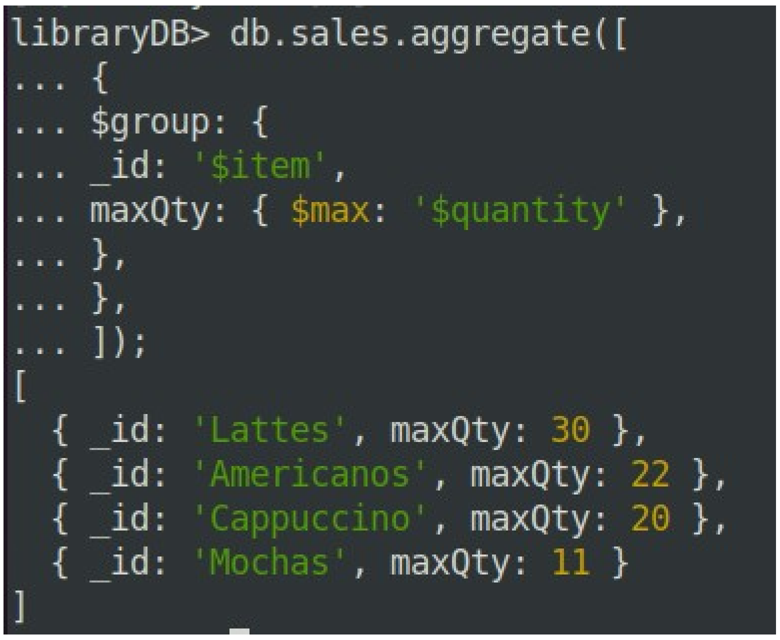
db.sales.aggregate([{$group: {_id: null,maxQty: { $max: '$quantity' },},},{$project: {_id: 0,},},]);



```
libraryDB> db.sales.aggregate([
... {
... $group: {
... _id: null,
... maxQty: { $max: '$quantity' },
... },
... },
... {
... $project: {
... _id: 0,
... },
... },
... ]);
[ { maxQty: 30 } ]
```

9. Group documents in the item field and returns the maximum quantity per group of documents.

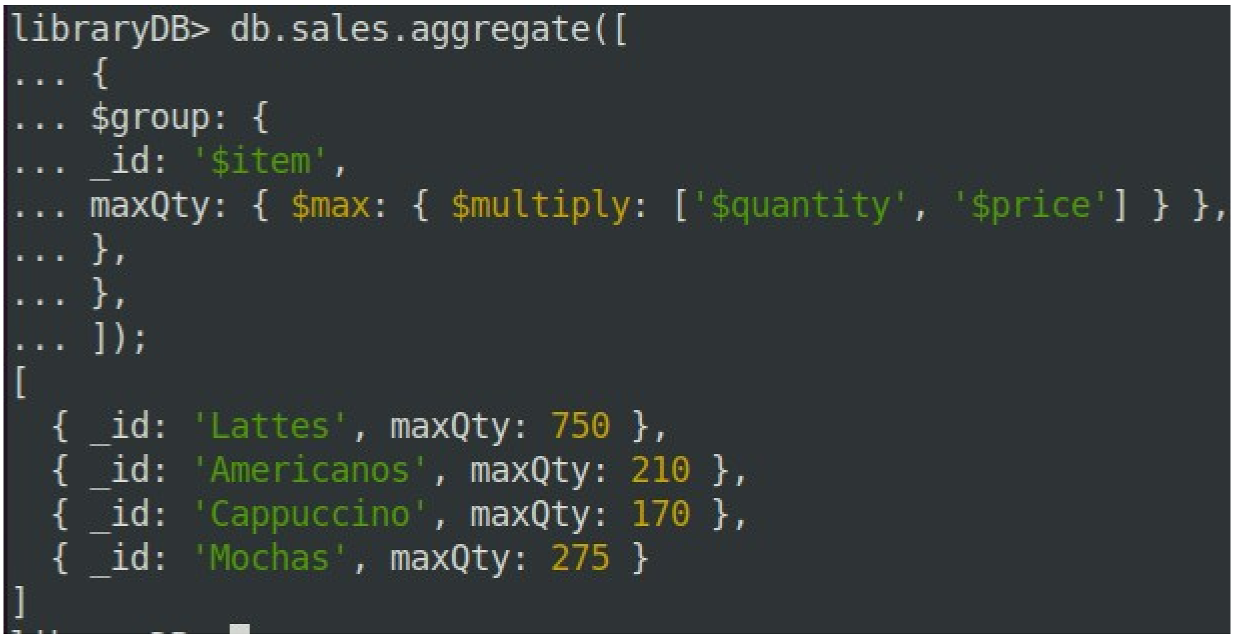
db.sales.aggregate([{$group: {_id: '$item',maxQty: { $max: '$quantity' },},},]);



```
libraryDB> db.sales.aggregate([
... {
... $group: {
... _id: '$item',
... maxQty: { $max: '$quantity' },
... },
... },
... ]);
[
  { _id: 'Lattes', maxQty: 30 },
  { _id: 'Americanos', maxQty: 22 },
  { _id: 'Cappuccino', maxQty: 20 },
  { _id: 'Mochas', maxQty: 11 }
]
```

**10. Groups the documents by the item field and returns the maximum amount for each group of
sales.**

db.sales.aggregate([{$group: {_id: '$item',maxQty: { $max: { $multiply: ['$quantity', '$price'] }},
},},]);



```
libraryDB> db.sales.aggregate([
... {
... $group: {
... _id: '$item',
... maxQty: { $max: { $multiply: ['$quantity', '$price'] } },
... },
... },
... ]);
[
  { _id: 'Lattes', maxQty: 750 },
  { _id: 'Americanos', maxQty: 210 },
  { _id: 'Cappuccino', maxQty: 170 },
  { _id: 'Mochas', maxQty: 275 }
]
```