Objectives

After completing this lab you will be able to:

- Describe simple aggregation operators that process and compute data such as $sort, $limit, $group, $sum, $min, $max, and $avg
- Combine operators to create multi-stage aggregation pipelines
- Build aggregation pipelines that draw insights about the data by returning aggregated values

Load sample data into the **training** database.

```
test> use training
... db.marks.insert({"name":"Ramesh","subject":"maths","marks":87})
... db.marks.insert({"name":"Ramesh","subject":"english","marks":59})
... db.marks.insert({"name":"Ramesh","subject":"science","marks":77})
... db.marks.insert({"name":"Rav","subject":"maths","marks":62})
... db.marks.insert({"name":"Rav","subject":"english","marks":83})
... db.marks.insert({"name":"Rav","subject":"science","marks":71})
... db.marks.insert({"name":"Alison","subject":"maths","marks":84})
... db.marks.insert({"name":"Alison","subject":"english","marks":82})
... db.marks.insert({"name":"Alison","subject":"science","marks":86})
... db.marks.insert({"name":"Steve","subject":"maths","marks":81})
... db.marks.insert({"name":"Steve","subject":"english","marks":89})
... db.marks.insert({"name":"Steve","subject":"science","marks":77})
... db.marks.insert({"name":"Jan","subject":"english","marks":0,"reason":"absent"})
switched to db training
```

Exercise 2 - Limiting the rows in the output

Using the $limit operator we can limit the number of documents printed in the output.
This command will print only 2 documents from the **marks** collection.

```
training> use training
... db.marks.aggregate([{"$limit":2}])
already on db training
```

Exercise 3 - Sorting based on a column

sorts the documents based on field marks in ascending order

```
training> db.marks.aggregate([{"$sort":{"marks":1}}])
```

Exercise 4 - Sorting and limiting

Aggregation usually involves using more than one operator.
A pipeline consists of one or more operators declared inside an array.
The operators are comma separated.
Mongodb executes the first operator in the pipeline and sends its output to the next operator.

Let us create a two stage pipeline that answers the question “What are the top 2 marks?”.

```
test> db.marks.aggregate([

... {"$sort":{"marks":-1}},

... {"$limit":2}

... ])

... ```

... node:internal/readline/emitKeypressEvents:74

          throw err;

          ^



TypeError: Generator is already running

   at emitKeys.next (<anonymous>)

   at LineByLineInput.onData (node:internal/readline/emitKeypressEvents:64:36)

   at LineByLineInput.emit (node:events:524:28)

   at LineByLineInput.emit (node:domain:489:12)
```

```
    at addChunk (node:internal/streams/readable:561:12)

    at readableAddChunkPushByteMode (node:internal/streams/readable:512:3)

    at Readable.push (node:internal/streams/readable:392:5)

    at LineByLineInput.push (eval at module.exports (node:lib-boxednode/mongosh:103:20), <anonymous>:11:3666768)

    at LineByLineInput._flush (eval at module.exports (node:lib-boxednode/mongosh:103:20), <anonymous>:11:3666507)

    at LineByLineInput.nextLine (eval at module.exports (node:lib-boxednode/mongosh:103:20), <anonymous>:11:3665516)



Node.js v20.19.5
```

The above query is equivalent to the below sql query.

```
db.marks.aggregate([

{"$sort":{"marks":-1}},

{"$limit":2}

])
```

# Exercise 5 - Group by

The operator $group by, along with operators like $sum, $avg, $min, $max, allows us to perform grouping operations.

This aggregation pipeline prints the average marks across all subjects.

```



db.marks.aggregate([

{

    "$group":{

        "_id":"$subject",
        "average":{"$avg":"$marks"}

        }

}

])
```

```
SELECT subject, average(marks)
FROM marks
GROUP BY subject
```

Exercise 6 - Putting it all together

Now let us put together all the operators we have learnt to answer the question. “Who are the top 2 students by average marks?”
This involves:

- finding the average marks per student.
- sorting the output based on average marks in descending order.
- limiting the output to two documents.

```
test> db.marks.aggregate([
... {
...     "$group":{
...         "_id":"$name",
...         "average":{"$avg":"$marks"}
...         }
... },
... {
...     "$sort":{"average":-1}
... },
... {
...     "$limit":2
... }
... ])

test>
```