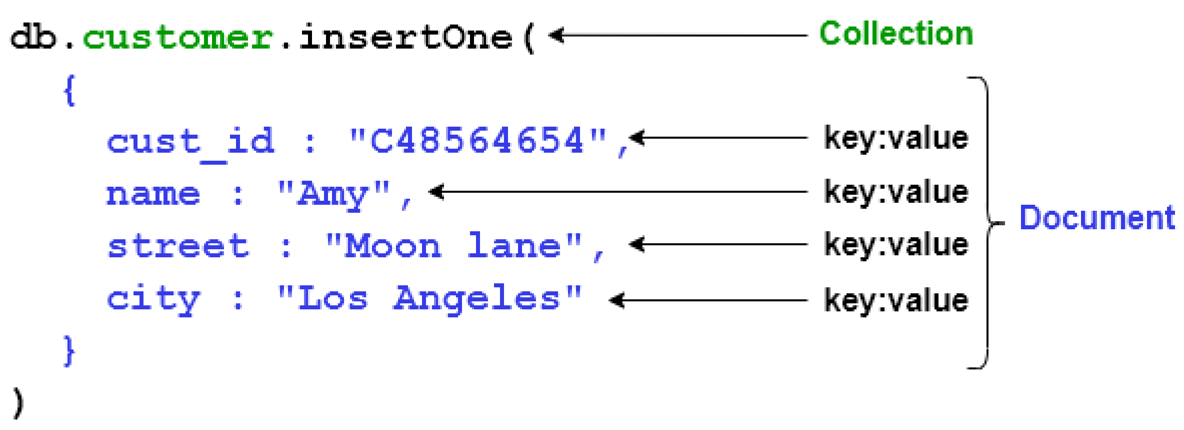
MongoDB Cheat Sheet

Important Note
MongoDB functions has the following structure:
`<database_name>.<collection_name>.function_name()`

Basic Shell Commands

Description	Command
Show all current databases managed by MongoDB	`show databases` or `show dbs`
Switch databases, also "creates" the database	`use <database_name>`
Show the current collections in the database	`show collections`
Show help (general), database related functions and collection related functions respectively.	`help` or `db.help()` or `db.collection.help()`

Inserting Documents

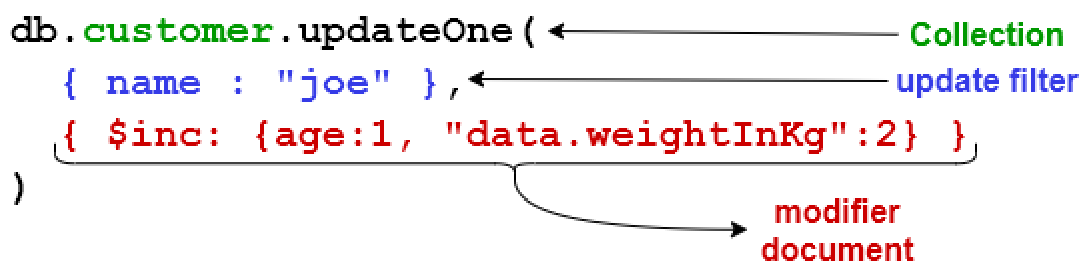


```
db.customer.insertOne(
   {
     cust_id : "C48564654",
     name : "Amy",
     street : "Moon lane",
     city : "Los Angeles"
   }
)
```

Figure 1: Insert command.

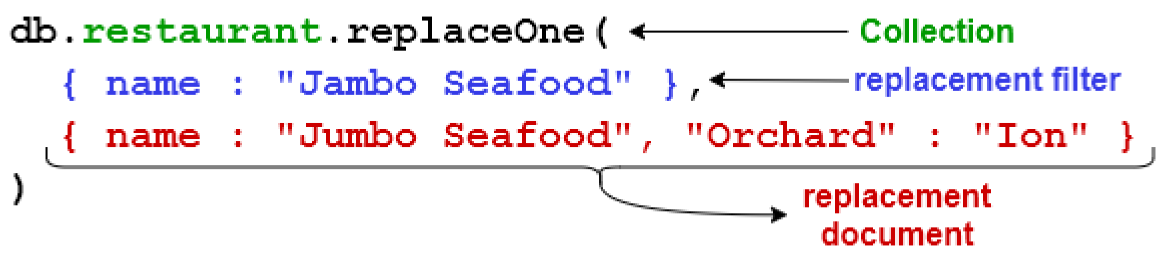
Command Description	Command
Insert **only** 1 document [doc]	`db.collection.insertOne()`
Insert multiple documents [doc]	`db.collection.insertMany()`
Inserts a document or documents into a collection. [doc]	`db.collection.insert()`

Updating Documents



```
db.customer.updateOne(
   { name : "joe" },
  { $inc: {age:1, "data.weightInKg":2} }
)
```

Figure 2: Update command.



```
db.restaurant.replaceOne(
   { name : "Jambo Seafood" },
  { name : "Jumbo Seafood", "Orchard" : "Ion" }
)
```

Figure 3: Replace command.

Command Description	Command
Update **only** 1 document [doc]	`db.collection.updateOne(query, update, options)`
Updates all documents that match the specified filter for a collection. [doc]	`db.collection.updateMany(query, update, options)`
Update an existing document or documents in a collection [doc]	`db.collection.update(query, update, options)`
Replaces a single document within the collection based on the filter. [doc]	`db.collection.replaceOne(query, replacement, options)`

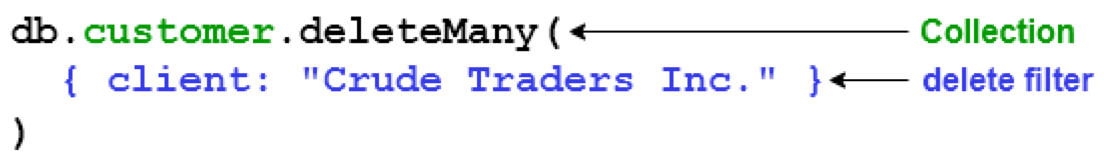
where:

- `query` - the criteria for the update. Uses the same format as the `find` query criteria
- `update` - specifies how the document should be updated (aka the *modifier document*)
- `replacement` - the replacement document
- `options` - refers to the additional options that are specific to the command

There are several operators available to the `update` function, some are listed below:

Expressions Description	Expressions
Increments the value of the field by the specified amount.	`$inc`
Replaces a value of the field.	`$set`
Removes the specified field from a document.	`$unset`
Adds an item to an array.	`$push`
Removes the first or last item of an array.	`$pop`
Removes all array elements that match a specified query.	`$pull`
Acts as a placeholder to update all elements that match the `arrayFilters` condition for the documents that match the query condition.	`$[<identifier>]`

Removing Documents



```
db.customer.deleteMany(
   { client: "Crude Traders Inc." }
)
```

Figure 4: Delete command.

Command Description	Command
Removes a single document from a collection. [doc]	`db.collection.deleteOne(filter, options)`
Removes all documents that match the `filter` from a collection. [doc]	`db.collection.deleteMany(filter, options)`
Removes a collection from the database. [doc]	`db.collection.drop(options)`
Removes documents from a collection. [doc]	`db.collection.remove(filter, options)`

where:

- `filter` - specifies the deletion criteria. To delete all documents in a collection, pass in an empty document (`{ }`).
- `options` - refers to the additional options that are specific to the command
- the difference between the `remove` & `delete` function is in their return values.

Finding Documents

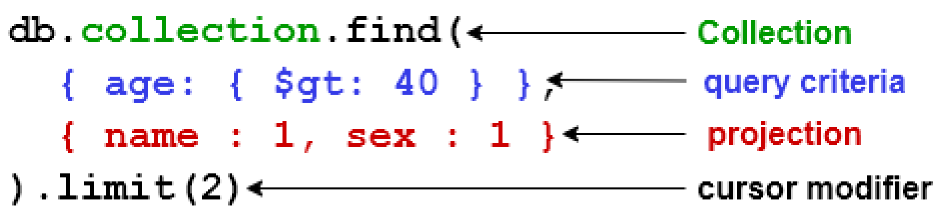

```
db.collection.find(
   { age: { $gt: 40 } },
   { name : 1, sex : 1 }
).limit(2)
```

Figure 5: Find command.

Command Description	Command
Selects documents in a collection and returns a **cursor** to the selected documents. [doc]	`db.collection.find(query, projection)`
Returns one document that satisfies the specified query criteria on the collection. If multiple documents satisfy the query, this method returns the first document. [doc]	`db.collection.findOne(query, projection)`

where:

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

The `projection` parameter takes a document of the form:
`{ <field1>: <value>, <field2>: <value> ... }`

The `value` has several options:

- `1` or `true` - specifies the inclusion of a field.
- `0` or `false` - specifies the exclusion of a field.
- combination of aggregation expressions
- array projection operators

The `projection` parameter **cannot** have a mix of inclusion and exclusion values. The rule of thumb is to **only declare** all *field-value* pairs that are to be shown (inclusion), save for the `_id` field which can be independently excluded.

Aggregation Pipeline Stages

MongoDB uses an Aggregation Framework to perform analytics on documents in one or more collections. The aggregate syntax begins with the `aggregate` command following the format: `db.collection.aggregate(pipeline, options)`.

The pipeline consists of many stages and some are listed below

Stages	Description
`$project`	Passes along the documents with the requested fields to the next stage in the pipeline. The specified fields can be existing fields from the input documents or newly computed fields.
`$unwind`	Deconstructs an array field from the input documents to output a document for each element. Each output document is the same input document but with the value of the array field replaced by the element.
`$match`	Filters the documents to pass only the documents that match the specified condition(s) to the next pipeline stage.
`$group`	Grouping values from multiple documents and perform some type of aggregation operation on them then returning a document.
`$sort`	Sorts all input documents and returns them to the pipeline in sorted order.
`$limit`	Limits the number of documents passed to the next stage in the pipeline.
`$lookup`	Performs a **left outer join** to an unsharded collection (aka the collection is not distributed) in the same database to filter in documents from the "joined" collection for processing.
`$merge`	Writes the results of the aggregation pipeline to a specified collection. This operator **must be the last stage** in the pipeline.
`$sample`	Randomly selects the specified number of documents from its input.
`$skip`	Skips over the specified number of documents that pass into the stage and passes the remaining documents to the next stage in the pipeline.

Important Note
Aggregation pipeline stages uses their own aggregation pipeline operators! These operators are **different** from the operators used for the query, projection and update functions.

Comparison with SQL's Aggregation

From the MongoDB's SQL to Aggregation Mapping Chart

SQL Terms, Functions, and Concepts	MongoDB Aggregation Operators
WHERE	`$match`
GROUP BY	`$group`
HAVING	`$match`
SELECT	`$project`
ORDER BY	`$sort`
LIMIT	`$limit`
SUM()	`$sum`
COUNT()	`$sum` or `$sortByCount`
JOIN	`$lookup`
SELECT INTO NEW TABLE	`$out`
MERGE INTO TABLE	`$merge` MongoDB 4.2 onwards
UNION ALL	`$unionWith` MongoDB 4.4 onwards

But if you would like to read further on how the concepts map over, it can be found in SQL to MongoDB Mapping Chart.