Collection

Name	Description
db.collection.aggregate()	Provides access to the aggregation framework pipeline
db.collection.count()	Wraps count to return a count of the number of documents in a collection or matching a query.
db.collection.createIndex()	Builds an index on a collection. Use db.collection.ensureIndex().
db.collection.getIndexStats()	Renders a human-readable view of the data collected by indexStats which reflects B-tree utilization.
db.collection.dataSize()	Returns the size of the collection. Wraps the size field in the output of the collStats.
db.collection.distinct()	Returns an array of documents that have distinct values for the specified field.
db.collection.drop()	Removes the specified collection from the database.
db.collection.dropIndex()	Removes a specified index on a collection.
db.collection.dropIndexes()	Removes all indexes on a collection.
db.collection.ensureIndex()	Creates an index if it does not currently exist. If the index exists ensureIndex() does nothing.
db.collection.find()	Performs a query on a collection and returns a cursor object.
db.collection.findAndModify()	Atomically modifies and returns a single document.
db.collection.findOne()	Performs a query and returns a single document.
db.collection.getIndexes()	Returns an array of documents that describe the existing indexes on a collection.
db.collection.getShardDistribution()	For collections in sharded clusters, db.collection.getShardDistribution() reports data of chunk distribution.
db.collection.getShardVersion()	Internal diagnostic method for shard cluster.
db.collection.group()	Provides simple data aggregation function. Groups documents in a collection by a key, and processes the results. Use aggregate() for more complex data aggregation.
db.collection.insert()	Creates a new document in a collection.
db.collection.isCapped()	Reports if a collection is a capped collection.
db.collection.mapReduce()	Performs map-reduce style data aggregation.
db.collection.reIndex()	Rebuilds all existing indexes on a collection.
db.collection.remove()	Deletes documents from a collection.
db.collection.renameCollection()	Changes the name of a collection.
db.collection.save()	Provides a wrapper around an insert() and update() to insert new documents.
db.collection.stats()	Reports on the state of a collection. Provides a wrapper around the collStats.
db.collection.storageSize()	Reports the total size used by the collection. Provides a wrapper around the storageSize field of the collStats output.
db.collection.totalSize()	Reports the total size of a collection, including the size of all documents and all indexes on a collection.
db.collection.totalIndexSize()	Reports the total size used by the indexes on a collection. Provides a wrapper around the totalIndexSize field of the collStats output.
db.collection.update()	Modifies a document in a collection.

db.collection.validate()	Performs diagnostic operations on a collection.

Cursor

Name	Description
cursor.addOption()	Adds special wire protocol flags that modify the behavior of the query.’
cursor.batchSize()	Controls the number of documents MongoDB will return to the client in a single network message.
cursor.count()	Returns a count of the documents in a cursor.
cursor.explain()	Reports on the query execution plan , including index use, for a cursor.
cursor.forEach()	Applies a JavaScript function for every document in a cursor.
cursor.hasNext()	Returns true if the cursor has documents and can be iterated.
cursor.hint()	Forces MongoDB to use a specific index for a query.
cursor.limit()	Constrains the size of a cursor’s result set.
cursor.map()	Applies a function to each document in a cursor and collects t he return values in an array.
cursor.max()	Specifies an exclusive upper index bound for a cursor. For use with cursor.hint()
cursor.min()	Specifies an inclusive lower index bound for a cursor. For use with cursor.hint()
cursor.next()	Returns the next document in a cursor.
cursor.objsLeftInBatch()	Returns the number of documents left in the current cursor batch.
cursor.readPref()	Specifies a read preference to a cursor to control how the client directs queries to a replica set.
cursor.showDiskLoc()	Returns a cursor with modified documents that include the on-disk location of the document.
cursor.size()	Returns a count of the documents in the cursor after applying skip() and limit() methods.
cursor.skip()	Returns a cursor that begins returning results only after passing or skipping a number of documents.
cursor.snapshot()	Forces the cursor to use the index on the _id field. Ensures that the cursor returns each document, with regards to the value of the _id field, only once.
cursor.sort()	Returns results ordered according to a sort specification.
cursor.toArray()	Returns an array that contains all documents returned by the cursor.

Database

Name	Description
db.addUser()	Adds a user to a database, and allows administrators to configure the user’s privileges.
db.auth()	Authenticates a user to a database.
db.changeUserPassword()	Changes an existing user’s password.
db.cloneCollection()	Copies data directly between MongoDB instances. Wraps cloneCollection.
db.cloneDatabase()	Copies a database from a remote host to the

	current host. Wraps clone.
db.commandHelp()	Returns help information for a database command.
db.copyDatabase()	Copies a database to another database on the current host. Wraps copydb.
db.createCollection()	Creates a new collection. Commonly used to create a capped collection.
db.currentOp()	Reports the current in-progress operations.
db.dropDatabase()	Removes the current database.
db.eval()	Passes a JavaScript function to the mongod instance for server-side JavaScript evaluation.
db.fsyncLock()	Flushes writes to disk and locks the database to prevent write operations and assist backup operations. Wraps fsync.
db.fsyncUnlock()	Allows writes to continue on a database locked with db.fsyncLock().
db.getCollection()	Returns a collection object. Used to access collections with names that are not valid in the mongo shell.
db.getCollectionNames()	Lists all collections in the current database.
db.getLastError()	Checks and returns the status of the last operation. Wraps getLastError.
db.getLastErrorObj()	Returns the status document for the last operation. Wraps getLastError.
db.getMongo()	Returns the Mongo() connection object for the current connection.
db.getName()	Returns the name of the current database.
db.getPrevError()	Returns a status document containing all errors since the last error reset. Wraps getPrevError.
db.getProfilingLevel()	Returns the current profiling level for database operations.
db.getProfilingStatus()	Returns a document that reflects the current profiling level and the profiling threshold.
db.getReplicationInfo()	Returns a document with replication statistics.
db.getSiblingDB()	Provides access to the specified database.
db.help()	Displays descriptions of common db object methods.
db.hostInfo()	Returns a document with information about the system MongoDB runs on. Wraps hostInfo
db.isMaster()	Returns a document that reports the state of the replica set.
db.killOp()	Terminates a specified operation.
db.listCommands()	Displays a list of common database commands.
db.loadServerScripts()	Loads all scripts in the system.js collection for the current database into the shell session.
db.logout()	Ends an authenticated session.
db.printCollectionStats()	Prints statistics from every collection. Wraps db.collection.stats().
db.printReplicationInfo()	Prints a report of the status of the replica set from the perspective of the primary.
db.printShardingStatus()	Prints a report of the sharding configuration and the chunk ranges.
db.printSlaveReplicationInfo()	Prints a report of the status of the replica set from the perspective of the secondary.
db.removeUser()	Removes a user from a database.
db.repairDatabase()	Runs a repair routine on the current database.
db.resetError()	Resets the error message returned by db.getPrevError() and getPrevError.
db.runCommand()	Runs a database command.
db.serverBuildInfo()	Returns a document that displays the compilation

	parameters for the mongod instance. Wraps buildinfo.
db.serverStatus()	Returns a document that provides an overview of the state of the database process.
db.setProfilingLevel()	Modifies the current level of database profiling.
db.shutdownServer()	Shuts down the current mongod or mongos process cleanly and safely.
db.stats()	Returns a document that reports on the state of the current database.
db.version()	Returns the version of the mongod instance.

Replication

Name	Description
rs.add()	Adds a member to a replica set.
rs.addArb()	Adds an arbiter to a replica set.
rs.conf()	Returns the replica set configuration document.
rs.freeze()	Prevents the current member from seeking election as primary for a period of time.
rs.help()	Returns basic help text for replica set functions.
rs.initiate()	Initializes a new replica set.
rs.reconfig()	Re-configures a replica set by applying a new replica set configuration object.
rs.remove()	Remove a member from a replica set.
rs.slaveOk()	Sets the slaveOk property for the current connection. Deprecated. Use readPref() and Mongo.setReadPref() to set read preference.
rs.status()	Returns a document with information about the state of the replica set.
rs.stepDown()	Causes the current primary to become a secondary which forces an election.
rs.syncFrom()	Sets the member that this replica set member will sync from, overriding the default sync target selection logic.

Sharding

Name	Description
sh._adminCommand	Runs a database command against the admin database, like db.runCommand(), but can confirm that it is issued against a mongos.
sh._checkFullName()	Tests a namespace to determine if its well formed.
sh._checkMongos()	Tests to see if the mongo shell is connected to a mongos instance.
sh._lastMigration()	Reports on the last chunk migration.
sh.addShard()	Adds a shard to a sharded cluster.
sh.addShardTag()	Associates a shard with a tag, to support tag aware sharding.
sh.addTagRange()	Associates range of shard keys with a shard tag, to support tag aware sharding.
sh.disableBalancing()	Disable balancing on a single collection in a sharded database. Does not affect balancing of other collections in a sharded cluster.

sh.enableBalancing()	Activates the sharded collection balancer process if previously disabled using sh.disableBalancing().
sh.enableSharding()	Enables sharding on a specific database.
sh.getBalancerHost()	Returns the name of a mongos that’s responsible for the balancer process.
sh.getBalancerState()	Returns a boolean to report if the balancer is currently enabled.
sh.help()	Returns help text for the sh methods.
sh.isBalancerRunning()	Returns a boolean to report if the balancer process is currently migrating chunks.
sh.moveChunk()	Migrates a chunk in a sharded cluster.
sh.removeShardTag()	Removes the association between a shard and a shard tag shard tag.
sh.setBalancerState()	Enables or disables the balancer which migrates chunks between shards.
sh.shardCollection()	Enables sharding for a collection.
sh.splitAt()	Divides an existing chunk into two chunks using a specific value of the shard key as the dividing point.
sh.splitFind()	Divides an existing chunk that contains a document matching a query into two approximately equal chunks.
sh.startBalancer()	Enables the balancer and waits for balancing to start.
sh.status()	Reports on the status of a sharded cluster, as db.printShardingStatus().
sh.stopBalancer()	Disables the balancer and waits for any in progress balancing rounds to complete.
sh.waitForBalancer()	Internal. Waits for the balancer state to change.
sh.waitForBalancerOff()	Internal. Waits until the balancer stops running.
sh.waitForDLock()	Internal. Waits for a specified distributed sharded cluster lock.
sh.waitForPingChange()	Internal. Waits for a change in ping state from one of the mongos in the sharded cluster.

Connection

Name	Description
Mongo.getDB()	Returns a database object.
Mongo.getReadPrefMode()	Returns the current read preference mode for the MongoDB connection.
Mongo.getReadPrefTagSet()	Returns the read preference tag set for the MongoDB connection.
Mongo.setReadPref()	Sets the read preference for the MongoDB connection.
Mongo.setSlaveOk()	Allows operations on the current connection to read from secondary members.
Mongo()	Creates a new connection object.
connect()	Connects to a MongoDB instance and to a specified database on that instance.

Subprocess

Name	Description
clearRawMongoProgramOutput()	For internal use.
rawMongoProgramOutput()	For internal use.
run()	For internal use.
runMongoProgram()	For internal use.
runProgram()	For internal use.
startMongoProgram()	For internal use.
stopMongoProgram()	For internal use.
stopMongoProgramByPid()	For internal use.
stopMongod()	For internal use.
waitMongoProgramOnPort()	For internal use.
waitProgram()	For internal use.

Native

Name	Description
cat()	Returns the contents of the specified file.
cd()	Changes the current working directory to the specified path.
copyDbpath()	Copies a local dbpath. For internal use.
resetDbpath()	Removes a local dbpath. For internal use.
fuzzFile()	For internal use to support testing.
getHostName()	Returns the hostname of the system running the mongo shell.
getMemInfo()	Returns a document that reports the amount of memory used by the shell.
hostname()	Returns the hostname of the system running the shell.
_isWindows()	Returns true if the shell runs on a Windows system; false if a Unix or Linux system.
listFiles()	Returns an array of documents that give the name and size of each object in the directory.
load()	Loads and runs a JavaScript file in the shell.
ls()	Returns a list of the files in the current directory.
md5sumFile()	The md5 hash of the specified file.
mkdir()	Creates a directory at the specified path.
pwd()	Returns the current directory.
quit()	Exits the current shell session.
_rand()	Returns a random number between 0 and 1.
removeFile()	Removes the specified file from the local file system.
_srand()	For internal use.