**SQL to MongoDB Mapping Chart**[**¶**](https://gist.github.com/aponxi/4380516#sql-to-mongodb-mapping-chart)

In addition to the charts that follow, you might want to consider the [*Frequently Asked Questions*](https://gist.github.com/faq/) section for a selection of common questions about MongoDB.

**Executables**[**¶**](https://gist.github.com/aponxi/4380516#executables)

The following table presents the MySQL/Oracle executables and the corresponding MongoDB executables.

|  | **MySQL/Oracle** | **MongoDB** |
| --- | --- | --- |
| Database Server | mysqld/oracle | [*mongod*](https://gist.github.com/mongod/) |
| Database Client | mysql/sqlplus | [*mongo*](https://gist.github.com/mongo/) |

**Terminology and Concepts**[**¶**](https://gist.github.com/aponxi/4380516#terminology-and-concepts)

The following table presents the various SQL terminology and concepts and the corresponding MongoDB terminology and concepts.

| **SQL Terms/Concepts** | **MongoDB Terms/Concepts** |
| --- | --- |
| database | [*database*](https://gist.github.com/glossary/#term-database) |
| table | [*collection*](https://gist.github.com/glossary/#term-collection) |
| row | [*document*](https://gist.github.com/glossary/#term-document) or [*BSON*](https://gist.github.com/glossary/#term-bson) document |
| column | [*field*](https://gist.github.com/glossary/#term-field) |
| index | [*index*](https://gist.github.com/glossary/#term-index) |
| table joins | embedded documents and linking |
| primary key  Specify any unique column or column combination as primary key. | [*primary key*](https://gist.github.com/glossary/#term-primary-key)  In MongoDB, the primary key is automatically set to the[*\_id*](https://gist.github.com/glossary/#term-id) field. |
| aggregation (e.g. group by) | aggregation framework  See the [*SQL to Aggregation Framework Mapping Chart*](https://gist.github.com/sql-aggregation-comparison/). |

**Examples**[**¶**](https://gist.github.com/aponxi/4380516#examples)

The following table presents the various SQL statements and the corresponding MongoDB statements. The examples in the table assume the following conditions:

* The SQL examples assume a table named users.
* The MongoDB examples assume a collection named users that contain documents of the following prototype:
* {
* \_id: ObjectID("509a8fb2f3f4948bd2f983a0"),
* user\_id: "abc123",
* age: 55,
* status: 'A'
* }

**Create and Alter**[**¶**](https://gist.github.com/aponxi/4380516#create-and-alter)

The following table presents the various SQL statements related to table-level actions and the corresponding MongoDB statements.

| **SQL Schema Statements** | **MongoDB Schema Statements** | **Reference** |
| --- | --- | --- |
| CREATE TABLE users (  id MEDIUMINT NOT NULL  AUTO\_INCREMENT,  user\_id Varchar(30),  age Number,  status char(1),  PRIMARY KEY (id)  ) | Implicitly created on first [insert](https://gist.github.com/method/db.collection.insert/#db.collection.insert) operation. The primary key \_id is automatically added if \_id field is not specified.  db.users.insert( {  user\_id: "abc123",  age: 55,  status: "A"  } )  However, you can also explicitly create a collection:  db.createCollection("users") | See [insert()](https://gist.github.com/method/db.collection.insert/#db.collection.insert) and[createCollection()](https://gist.github.com/method/db.createCollection/#db.createCollection)for more information. |
| ALTER TABLE users  ADD join\_date DATETIME | Collections do not describe or enforce the structure of the constituent documents. See the [Schema Design](http://www.mongodb.org/display/DOCS/Schema+Design)wiki page for more information. | See [update()](https://gist.github.com/method/db.collection.update/#db.collection.update) and[$set](https://gist.github.com/operators/#_S_set) for more information on changing the structure of documents in a collection. |
| ALTER TABLE users  DROP COLUMN join\_date | Collections do not describe or enforce the structure of the constituent documents. See the [Schema Design](http://www.mongodb.org/display/DOCS/Schema+Design)wiki page for more information. | See [update()](https://gist.github.com/method/db.collection.update/#db.collection.update) and[$set](https://gist.github.com/operators/#_S_set) for more information on changing the structure of documents in a collection. |
| CREATE INDEX idx\_user\_id\_asc  ON users(user\_id) | db.users.ensureIndex( { user\_id: 1 } ) | See [ensureIndex()](https://gist.github.com/method/db.collection.ensureIndex/" \l "db.collection.ensureIndex" \o "db.collection.ensureIndex)and [*indexes*](https://gist.github.com/core/indexes/) for more information. |
| CREATE INDEX  idx\_user\_id\_asc\_age\_desc  ON users(user\_id, age DESC) | db.users.ensureIndex( { user\_id: 1, age: -1 } ) | See [ensureIndex()](https://gist.github.com/method/db.collection.ensureIndex/" \l "db.collection.ensureIndex" \o "db.collection.ensureIndex)and [*indexes*](https://gist.github.com/core/indexes/) for more information. |
| DROP TABLE users | db.users.drop() | See [drop()](https://gist.github.com/method/db.collection.drop/#db.collection.drop) for more information. |

**Insert**[**¶**](https://gist.github.com/aponxi/4380516#insert)

The following table presents the various SQL statements related to inserting records into tables and the corresponding MongoDB statements.

| **SQL INSERT Statements** | **MongoDB insert() Statements** | **Reference** |
| --- | --- | --- |
| INSERT INTO users(user\_id,  age,  status)  VALUES ("bcd001",  45,  "A") | db.users.insert( {  user\_id: "bcd001",  age: 45,  status: "A"  } ) | See [insert()](https://gist.github.com/method/db.collection.insert/#db.collection.insert) for more information. |

**Select**[**¶**](https://gist.github.com/aponxi/4380516#select)

The following table presents the various SQL statements related to reading records from tables and the corresponding MongoDB statements.

| **SQL SELECT Statements** | **MongoDB find() Statements** | **Reference** |
| --- | --- | --- |
| SELECT \*  FROM users | db.users.find() | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find) for more information. |
| SELECT id, user\_id, status  FROM users | db.users.find(  { },  { user\_id: 1, status: 1 }  ) | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find) for more information. |
| SELECT user\_id, status  FROM users | db.users.find(  { },  { user\_id: 1, status: 1, \_id: 0 }  ) | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find) for more information. |
| SELECT \*  FROM users  WHERE status = "A" | db.users.find(  { status: "A" }  ) | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find) for more information. |
| SELECT user\_id, status  FROM users  WHERE status = "A" | db.users.find(  { status: "A" },  { user\_id: 1, status: 1, \_id: 0 }  ) | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find) for more information. |
| SELECT \*  FROM users  WHERE status != "A" | db.users.find(  { status: { $ne: "A" } }  ) | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find) and [$ne](https://gist.github.com/operators/#_S_ne) for more information. |
| SELECT \*  FROM users  WHERE status = "A"  AND age = 50 | db.users.find(  { status: "A",  age: 50 }  ) | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find) and [$and](https://gist.github.com/operators/#_S_and) for more information. |
| SELECT \*  FROM users  WHERE status = "A"  OR age = 50 | db.users.find(  { $or: [ { status: "A" } ,  { age: 50 } ] }  ) | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find) and [$or](https://gist.github.com/operators/#_S_or) for more information. |
| SELECT \*  FROM users  WHERE age > 25 | db.users.find(  { age: { $gt: 25 } }  ) | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find) and [$gt](https://gist.github.com/operators/#_S_gt) for more information. |
| SELECT \*  FROM users  WHERE age < 25 | db.users.find(  { age: { $lt: 25 } }  ) | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find) and [$lt](https://gist.github.com/operators/#_S_lt) for more information. |
| SELECT \*  FROM users  WHERE age > 25  AND age <= 50 | db.users.find(  { age: { $gt: 25, $lte: 50 } }  ) | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find),[$gt](https://gist.github.com/operators/#_S_gt), and [$lte](https://gist.github.com/operators/#_S_lte)for more information. |
| SELECT \*  FROM users  WHERE user\_id like "%bc%" | db.users.find(  { user\_id: /bc/ }  ) | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find) and [$regex](https://gist.github.com/operators/#_S_regex) for more information. |
| SELECT \*  FROM users  WHERE user\_id like "bc%" | db.users.find(  { user\_id: /^bc/ }  ) | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find) and [$regex](https://gist.github.com/operators/#_S_regex) for more information. |
| SELECT \*  FROM users  WHERE status = "A"  ORDER BY user\_id ASC | db.users.find( { status: "A" } ).sort( { user\_id: 1 } ) | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find) and [sort()](https://gist.github.com/method/cursor.sort/#cursor.sort) for more information. |
| SELECT \*  FROM users  WHERE status = "A"  ORDER BY user\_id DESC | db.users.find( { status: "A" } ).sort( { user\_id: -1 } ) | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find) and [sort()](https://gist.github.com/method/cursor.sort/#cursor.sort) for more information. |
| SELECT COUNT(\*)  FROM users | db.users.count()  *or*  db.users.find().count() | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find) and [count()](https://gist.github.com/method/cursor.count/#cursor.count) for more information. |
| SELECT COUNT(user\_id)  FROM users | db.users.count( { user\_id: { $exists: true } } )  *or*  db.users.find( { user\_id: { $exists: true } } ).count() | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find),[count()](https://gist.github.com/method/cursor.count/#cursor.count), and[$exists](https://gist.github.com/operators/#_S_exists) for more information. |
| SELECT COUNT(\*)  FROM users  WHERE age > 30 | db.users.count( { age: { $gt: 30 } } )  *or*  db.users.find( { age: { $gt: 30 } } ).count() | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find),[count()](https://gist.github.com/method/cursor.count/#cursor.count), and[$gt](https://gist.github.com/operators/#_S_gt) for more information. |
| SELECT DISTINCT(status)  FROM users | db.users.distinct( "status" ) | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find) and [distinct()](https://gist.github.com/method/db.collection.distinct/#db.collection.distinct) for more information. |
| SELECT \*  FROM users  LIMIT 1 | db.users.findOne()  *or*  db.users.find().limit(1) | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find),[findOne()](https://gist.github.com/method/db.collection.findOne/" \l "db.collection.findOne" \o "db.collection.findOne), and [limit()](https://gist.github.com/method/cursor.limit/#cursor.limit) for more information. |
| SELECT \*  FROM users  LIMIT 5  SKIP 10 | db.users.find().limit(5).skip(10) | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find),[limit()](https://gist.github.com/method/cursor.limit/#cursor.limit), and[skip()](https://gist.github.com/method/cursor.skip/#cursor.skip) for more information. |
| EXPLAIN SELECT \*  FROM users  WHERE status = "A" | db.users.find( { status: "A" } ).explain() | See [find()](https://gist.github.com/method/db.collection.find/#db.collection.find) and [explain()](https://gist.github.com/method/cursor.explain/#cursor.explain) for more information. |

**Update Records**[**¶**](https://gist.github.com/aponxi/4380516#update-records)

The following table presents the various SQL statements related to updating existing records in tables and the corresponding MongoDB statements.

| **SQL Update Statements** | **MongoDB update() Statements** | **Reference** |
| --- | --- | --- |
| UPDATE users  SET status = "C"  WHERE age > 25 | db.users.update(  { age: { $gt: 25 } },  { $set: { status: "C" } },  { multi: true }  ) | See [update()](https://gist.github.com/method/db.collection.update/#db.collection.update), [$gt](https://gist.github.com/operators/#_S_gt), and [$set](https://gist.github.com/operators/#_S_set) for more information. |
| UPDATE users  SET age = age + 3  WHERE status = "A" | db.users.update(  { status: "A" } ,  { $inc: { age: 3 } },  { multi: true }  ) | See [update()](https://gist.github.com/method/db.collection.update/#db.collection.update), [$inc](https://gist.github.com/operators/#_S_inc), and [$set](https://gist.github.com/operators/#_S_set) for more information. |

**Delete Records**[**¶**](https://gist.github.com/aponxi/4380516#delete-records)

The following table presents the various SQL statements related to deleting records from tables and the corresponding MongoDB statements.

| **SQL Delete Statements** | **MongoDB remove() Statements** | **Reference** |
| --- | --- | --- |
| DELETE FROM users  WHERE status = "D" | db.users.remove( { status: "D" } ) | See [remove()](https://gist.github.com/method/db.collection.remove/#db.collection.remove) for more information. |
| DELETE FROM users | db.users.remove( ) |  |