SQL to MongoDB Mapping Chart

**On this page**

* [Terminology and Concepts](https://docs.mongodb.com/manual/reference/sql-comparison/#terminology-and-concepts)
* [Executables](https://docs.mongodb.com/manual/reference/sql-comparison/#executables)
* [Examples](https://docs.mongodb.com/manual/reference/sql-comparison/#examples)
* [Additional Resources](https://docs.mongodb.com/manual/reference/sql-comparison/#additional-resources)

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

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://docs.mongodb.com/manual/reference/glossary/#term-database) |
| table | [collection](https://docs.mongodb.com/manual/reference/glossary/#term-collection) |
| row | [document](https://docs.mongodb.com/manual/reference/glossary/#term-document) or [BSON](https://docs.mongodb.com/manual/reference/glossary/#term-bson) document |
| column | [field](https://docs.mongodb.com/manual/reference/glossary/#term-field) |
| index | [index](https://docs.mongodb.com/manual/reference/glossary/#term-index) |
| table joins | [$lookup](https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/#pipe._S_lookup), embedded documents |
| primary key  Specify any unique column or column combination as primary key. | [primary key](https://docs.mongodb.com/manual/reference/glossary/#term-primary-key)  In MongoDB, the primary key is automatically set to the [\_id](https://docs.mongodb.com/manual/reference/glossary/#term-id)field. |
| aggregation (e.g. group by) | aggregation pipeline  See the [SQL to Aggregation Mapping Chart](https://docs.mongodb.com/manual/reference/sql-aggregation-comparison/). |

Executables

The following table presents some database executables and the corresponding MongoDB executables. This table is *not* meant to be exhaustive.

|  | **MongoDB** | **MySQL** | **Oracle** | **Informix** | **DB2** |
| --- | --- | --- | --- | --- | --- |
| Database Server | [mongod](https://docs.mongodb.com/manual/reference/program/mongod/#bin.mongod) | mysqld | oracle | IDS | DB2 Server |
| Database Client | [mongo](https://docs.mongodb.com/manual/reference/program/mongo/#bin.mongo) | mysql | sqlplus | DB-Access | DB2 Client |

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

**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** |
| --- | --- |
| **CREATE** **TABLE** people (  id MEDIUMINT **NOT** **NULL**  AUTO\_INCREMENT,  user\_id Varchar(30),  age Number,  status char(1),  **PRIMARY** **KEY** (id)  ) | Implicitly created on first [insertOne()](https://docs.mongodb.com/manual/reference/method/db.collection.insertOne/#db.collection.insertOne) or [insertMany()](https://docs.mongodb.com/manual/reference/method/db.collection.insertMany/#db.collection.insertMany)operation. The primary key \_id is automatically added if \_id field is not specified.  db.people.insertOne( {  user\_id: "abc123",  age: 55,  status: "A"  } )  However, you can also explicitly create a collection:  db.createCollection("people") |
| **ALTER** **TABLE** people  **ADD** join\_date DATETIME | Collections do not describe or enforce the structure of its documents; i.e. there is no structural alteration at the collection level.  However, at the document level, [updateMany()](https://docs.mongodb.com/manual/reference/method/db.collection.updateMany/#db.collection.updateMany) operations can add fields to existing documents using the [$set](https://docs.mongodb.com/manual/reference/operator/update/set/#up._S_set) operator.  db.people.updateMany(  { },  { $set: { join\_date: **new** Date() } }  ) |
| **ALTER** **TABLE** people  **DROP** **COLUMN** join\_date | Collections do not describe or enforce the structure of its documents; i.e. there is no structural alteration at the collection level.  However, at the document level, [updateMany()](https://docs.mongodb.com/manual/reference/method/db.collection.updateMany/#db.collection.updateMany) operations can remove fields from documents using the [$unset](https://docs.mongodb.com/manual/reference/operator/update/unset/#up._S_unset) operator.  db.people.updateMany(  { },  { $unset: { "join\_date": "" } }  ) |
| **CREATE** **INDEX** idx\_user\_id\_asc  **ON** people(user\_id) | db.people.createIndex( { user\_id: 1 } ) |
| **CREATE** **INDEX**  idx\_user\_id\_asc\_age\_desc  **ON** people(user\_id, age **DESC**) | db.people.createIndex( { user\_id: 1, age: -1 } ) |
| **DROP** **TABLE** people | db.people.drop() |

For more information, see:

* [db.collection.insertOne()](https://docs.mongodb.com/manual/reference/method/db.collection.insertOne/#db.collection.insertOne)
* [db.collection.insertMany()](https://docs.mongodb.com/manual/reference/method/db.collection.insertMany/#db.collection.insertMany)
* [db.createCollection()](https://docs.mongodb.com/manual/reference/method/db.createCollection/#db.createCollection)
* [db.collection.updateMany()](https://docs.mongodb.com/manual/reference/method/db.collection.updateMany/#db.collection.updateMany)
* [$set](https://docs.mongodb.com/manual/reference/operator/update/set/#up._S_set)
* [$unset](https://docs.mongodb.com/manual/reference/operator/update/unset/#up._S_unset)
* [db.collection.createIndex()](https://docs.mongodb.com/manual/reference/method/db.collection.createIndex/#db.collection.createIndex)
* [Indexes](https://docs.mongodb.com/manual/indexes/)
* [db.collection.drop()](https://docs.mongodb.com/manual/reference/method/db.collection.drop/#db.collection.drop)
* [Data Modeling Concepts](https://docs.mongodb.com/manual/core/data-models/).

**Insert**

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

| **SQL INSERT Statements** | **MongoDB insertOne() Statements** |
| --- | --- |
| **INSERT** **INTO** people(user\_id,  age,  status)  **VALUES** ("bcd001",  45,  "A") | db.people.insertOne(  { user\_id: "bcd001", age: 45, status: "A" }  ) |

For more information, see [db.collection.insertOne()](https://docs.mongodb.com/manual/reference/method/db.collection.insertOne/#db.collection.insertOne).

**Select**

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

**NOTE**

The [find()](https://docs.mongodb.com/manual/reference/method/db.collection.find/#db.collection.find) method always includes the \_id field in the returned documents unless specifically excluded through [projection](https://docs.mongodb.com/manual/tutorial/project-fields-from-query-results/#projection). Some of the SQL queries below may include an \_id field to reflect this, even if the field is not included in the corresponding [find()](https://docs.mongodb.com/manual/reference/method/db.collection.find/#db.collection.find) query.

| **SQL SELECT Statements** | **MongoDB find() Statements** |
| --- | --- |
| **SELECT** \*  **FROM** people | db.people.find() |
| **SELECT** id,  user\_id,  status  **FROM** people | db.people.find(  { },  { user\_id: 1, status: 1 }  ) |
| **SELECT** user\_id, status  **FROM** people | db.people.find(  { },  { user\_id: 1, status: 1, \_id: 0 }  ) |
| **SELECT** \*  **FROM** people  **WHERE** status = "A" | db.people.find(  { status: "A" }  ) |
| **SELECT** user\_id, status  **FROM** people  **WHERE** status = "A" | db.people.find(  { status: "A" },  { user\_id: 1, status: 1, \_id: 0 }  ) |
| **SELECT** \*  **FROM** people  **WHERE** status != "A" | db.people.find(  { status: { $ne: "A" } }  ) |
| **SELECT** \*  **FROM** people  **WHERE** status = "A"  **AND** age = 50 | db.people.find(  { status: "A",  age: 50 }  ) |
| **SELECT** \*  **FROM** people  **WHERE** status = "A"  **OR** age = 50 | db.people.find(  { $or: [ { status: "A" } ,  { age: 50 } ] }  ) |
| **SELECT** \*  **FROM** people  **WHERE** age > 25 | db.people.find(  { age: { $gt: 25 } }  ) |
| **SELECT** \*  **FROM** people  **WHERE** age < 25 | db.people.find(  { age: { $lt: 25 } }  ) |
| **SELECT** \*  **FROM** people  **WHERE** age > 25  **AND** age <= 50 | db.people.find(  { age: { $gt: 25, $lte: 50 } }  ) |
| **SELECT** \*  **FROM** people  **WHERE** user\_id **like** "%bc%" | db.people.find( { user\_id: /bc/ } )  -or-  db.people.find( { user\_id: { $regex: /bc/ } } ) |
| **SELECT** \*  **FROM** people  **WHERE** user\_id **like** "bc%" | db.people.find( { user\_id: /^bc/ } )  -or-  db.people.find( { user\_id: { $regex: /^bc/ } } ) |
| **SELECT** \*  **FROM** people  **WHERE** status = "A"  **ORDER** **BY** user\_id **ASC** | db.people.find( { status: "A" } ).sort( { user\_id: 1 } ) |
| **SELECT** \*  **FROM** people  **WHERE** status = "A"  **ORDER** **BY** user\_id **DESC** | db.people.find( { status: "A" } ).sort( { user\_id: -1 } ) |
| **SELECT** **COUNT**(\*)  **FROM** people | db.people.count()  *or*  db.people.find().count() |
| **SELECT** **COUNT**(user\_id)  **FROM** people | db.people.count( { user\_id: { $exists: **true** } } )  *or*  db.people.find( { user\_id: { $exists: **true** } } ).count() |
| **SELECT** **COUNT**(\*)  **FROM** people  **WHERE** age > 30 | db.people.count( { age: { $gt: 30 } } )  *or*  db.people.find( { age: { $gt: 30 } } ).count() |
| **SELECT** **DISTINCT**(status)  **FROM** people | db.people.distinct( "status" ) |
| **SELECT** \*  **FROM** people  **LIMIT** 1 | db.people.findOne()  *or*  db.people.find().limit(1) |
| **SELECT** \*  **FROM** people  **LIMIT** 5  SKIP 10 | db.people.find().limit(5).skip(10) |
| **EXPLAIN** **SELECT** \*  **FROM** people  **WHERE** status = "A" | db.people.find( { status: "A" } ).explain() |

For more information, see:

* [db.collection.find()](https://docs.mongodb.com/manual/reference/method/db.collection.find/#db.collection.find)
* [db.collection.distinct()](https://docs.mongodb.com/manual/reference/method/db.collection.distinct/#db.collection.distinct)
* [db.collection.findOne()](https://docs.mongodb.com/manual/reference/method/db.collection.findOne/#db.collection.findOne)
* Query operators: [$ne](https://docs.mongodb.com/manual/reference/operator/query/ne/#op._S_ne), [$and](https://docs.mongodb.com/manual/reference/operator/query/and/#op._S_and), [$or](https://docs.mongodb.com/manual/reference/operator/query/or/#op._S_or), [$gt](https://docs.mongodb.com/manual/reference/operator/query/gt/#op._S_gt), [$lt](https://docs.mongodb.com/manual/reference/operator/query/lt/#op._S_lt), [$exists](https://docs.mongodb.com/manual/reference/operator/query/exists/#op._S_exists), [$lte](https://docs.mongodb.com/manual/reference/operator/query/lte/#op._S_lte), and [$regex](https://docs.mongodb.com/manual/reference/operator/query/regex/#op._S_regex).
* [limit()](https://docs.mongodb.com/manual/reference/method/cursor.limit/#cursor.limit)
* [skip()](https://docs.mongodb.com/manual/reference/method/cursor.skip/#cursor.skip)
* [explain()](https://docs.mongodb.com/manual/reference/method/cursor.explain/#cursor.explain)
* [sort()](https://docs.mongodb.com/manual/reference/method/cursor.sort/#cursor.sort)
* [count()](https://docs.mongodb.com/manual/reference/method/cursor.count/#cursor.count)

**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 updateMany() Statements** |
| --- | --- |
| **UPDATE** people  **SET** status = "C"  **WHERE** age > 25 | db.people.updateMany(  { age: { $gt: 25 } },  { $set: { status: "C" } }  ) |
| **UPDATE** people  **SET** age = age + 3  **WHERE** status = "A" | db.people.updateMany(  { status: "A" } ,  { $inc: { age: 3 } }  ) |

For more information, see [db.collection.updateMany()](https://docs.mongodb.com/manual/reference/method/db.collection.updateMany/#db.collection.updateMany), [$set](https://docs.mongodb.com/manual/reference/operator/update/set/#up._S_set), [$inc](https://docs.mongodb.com/manual/reference/operator/update/inc/#up._S_inc), and [$gt](https://docs.mongodb.com/manual/reference/operator/query/gt/#op._S_gt).

**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 deleteMany() Statements** |
| --- | --- |
| **DELETE** **FROM** people  **WHERE** status = "D" | db.people.deleteMany( { status: "D" } ) |
| **DELETE** **FROM** people | db.people.deleteMany({}) |