MongoDB Tutorial: Using Regex Operator for Pattern Matching

MongoDB **comparison query operators** such as **$eq** and **$ne** are useful for finding values that are equal to or not equal to a specific exact comparison value. When it's necessary to find values based on similarity instead, a pattern match is useful. To perform a pattern match in MongoDB, use **{ *field* : { $regex : "*pattern*" } }**, where field is the field you want to test and "*pattern*" is a regular expression pattern string that describes the general form of values that you want to match.

Patterns used with the $regex **evaluation query operator** can contain several special characters, or "metacharacters", that stand for something other than themselves.

While most characters in patterns match themselves, the following *special characters* are excaptions:

* asterisk (\*),
* plus sign (+),
* question mark (?),
* backslash (\),
* period (.),
* caret (^),
* square brackets ([ and ]),
* dollar sign ($),
* ampersand (&).
* or sign (|).

To match a special character, precede it with a backslash, like this \\*.

* The lack of special characters in a pattern matches any string that contains those characters, regardless of any characters comes before or after the pattern. For example, the pattern "a" matches any string that contains 'a' within the string.
* The '**^**' (caret) character matches strings that contain the pattern at the beginning of the string.
* The '**$**' (dollar sign) character matches strings that contain the pattern at the end of the string.
* The '**.**' (dot) character matches any single charater. "d.g" matches strings such as "big dog", "dig dug", and "good@guy". Because '**.**' matches any single character, it matches itself and the pattern "d.g" also matches the string "ed.gov".

The following table lists MySQL statements and the corresponding MongoDB statements.

| MySQL | MongoDB | Method |
| --- | --- | --- |
| [Create Connection](https://www.w3schools.com/nodejs/nodejs_mysql.asp) | [Create Client](https://www.w3schools.com/nodejs/nodejs_mongodb.asp) | [require('mongodb').MongoClient](https://mongodb.github.io/node-mongodb-native/3.0/api/MongoClient.html) |
| [Create Database](https://www.w3schools.com/nodejs/nodejs_mysql_create_db.asp) | [Create Connection & Database](https://www.w3schools.com/nodejs/nodejs_mongodb_create_db.asp) | [connect()](https://mongodb.github.io/node-mongodb-native/3.0/api/MongoClient.html#.connect) |
| [Create Table](https://www.w3schools.com/nodejs/nodejs_mysql_create_table.asp) | [Create Collection](https://www.w3schools.com/nodejs/nodejs_mongodb_createcollection.asp) | [createCollection()](https://mongodb.github.io/node-mongodb-native/3.0/api/Db.html#createCollection) |
| [Insert Into](https://www.w3schools.com/nodejs/nodejs_mysql_insert.asp) | [Insert](https://www.w3schools.com/nodejs/nodejs_mongodb_insert.asp) | [insertOne()](https://mongodb.github.io/node-mongodb-native/3.0/api/Collection.html#insertOne) |
| [Select From](https://www.w3schools.com/nodejs/nodejs_mysql_select.asp) | [Find](https://www.w3schools.com/nodejs/nodejs_mongodb_find.asp) | [findOne()](https://mongodb.github.io/node-mongodb-native/3.0/api/Collection.html#findOne) |
| [Where](https://www.w3schools.com/nodejs/nodejs_mysql_where.asp) | [Query](https://www.w3schools.com/nodejs/nodejs_mongodb_query.asp) | [find()](https://mongodb.github.io/node-mongodb-native/3.0/api/Collection.html#find) |
| Comparison Query Operators  = != > < IN NOT IN | Comparison Query Operators  [$eq](https://docs.mongodb.com/manual/reference/operator/query/eq/#op._S_eq) [$ne](https://docs.mongodb.com/manual/reference/operator/query/ne/#op._S_ne) [$gt](https://docs.mongodb.com/manual/reference/operator/query/gt/#op._S_gt) [$lt](https://docs.mongodb.com/manual/reference/operator/query/lt/#op._S_lt) [$in](https://docs.mongodb.com/manual/reference/operator/query/in/#op._S_in) [$nin](https://docs.mongodb.com/manual/reference/operator/query/nin/#op._S_nin) | |  |  | | --- | --- | | MySQL | MongoDB | | SELECT \* FROM Inventory WHERE qty = 20; | db.inventory.find( { qty: { $eq: 20 } } ) | | SELECT \* FROM Inventory WHERE qty != 20; | db.inventory.find( { qty: { $ne: 20 } } ) | | UPDATE inventory SET price = 9.99 WHERE carrier.fee > 2 | db.inventory.update( { "carrier.fee": { $gt: 2 } }, { $set: { price: 9.99 } } ) | | SELECT \* FROM inventory WHERE qty < 20; | db.inventory.find( { qty: { $lt: 20 } } ) | | SELECT \* FROM inventory WHERE qty IN (5, 15); | db.inventory.find( { qty: { $in: [ 5, 15 ] } } ) | | SELECT \* FROM inventory WHERE qty NOT IN (5, 15); | db.inventory.find( { qty: { $nin: [ 5, 15 ] } } ) | |
| Logical Query Operators  AND OR NOT | Logical Query Operators  [$and](https://docs.mongodb.com/manual/reference/operator/query/and/#op._S_and) [$or](https://docs.mongodb.com/manual/reference/operator/query/or/#op._S_or) [$not](https://docs.mongodb.com/manual/reference/operator/query/not/#op._S_not) [$nor](https://docs.mongodb.com/manual/reference/operator/query/nor/#op._S_nor) | |  |  | | --- | --- | | MySQL | MongoDB | | SELECT \* FROM inventory WHERE price > 19.99 AND category IN ('liquor', 'liqueur'); | db.inventory.find( { $and: [ { price: { $gt: 19.99 } }, { category: { $in: ['liquor', 'liqueur'] } } ] } ) | | SELECT \* FROM inventory WHERE qty < 20 OR price = 10; | db.inventory.find( { $or: [ { qty: { $lt: 20 } }, { price: 10 } ] } ) | | SELECT \* FROM inventory WHERE NOT (price > 1.00); | db.inventory.find( { price: { $not: { $gt: 1.99 } } } ) | | SELECT \* FROM inventory WHERE price != 1.99 AND qty >= 20 AND sale = FALSE; | db.inventory.find( { $nor: [ { price: 1.99 }, { qty: { $lt: 20 } }, { sale: true } ] } ) | |
| LIKE | [$regex](https://docs.mongodb.com/manual/reference/operator/query/regex/) | |  |  | | --- | --- | | MySQL | MongoDB | | LIKE '%a%' | { "name": { "$regex": "a" } | | LIKE 'A%' | { "name": { "$regex": "^A" } | | LIKE '%e' | { "name": { "$regex": "e$" } | | LIKE '%d\_g%' | { "name": { "$regex": "d.g" } | |
| [Order By](https://www.w3schools.com/nodejs/nodejs_mysql_orderby.asp) | [Sort](https://www.w3schools.com/nodejs/nodejs_mongodb_sort.asp) | [sort()](https://mongodb.github.io/node-mongodb-native/3.0/api/Cursor.html#sort) |
| [Delete](https://www.w3schools.com/nodejs/nodejs_mysql_delete.asp) | [Delete](https://www.w3schools.com/nodejs/nodejs_mongodb_delete.asp) | [deleteOne()](https://mongodb.github.io/node-mongodb-native/3.0/api/Collection.html#deleteOne) |
| [Drop Table](https://www.w3schools.com/nodejs/nodejs_mysql_drop_table.asp) | [Drop Collection](https://www.w3schools.com/nodejs/nodejs_mongodb_drop.asp) | [drop()](https://mongodb.github.io/node-mongodb-native/3.0/api/Collection.html#drop) |
| [Update](https://www.w3schools.com/nodejs/nodejs_mysql_update.asp) | [Update](https://www.w3schools.com/nodejs/nodejs_mongodb_update.asp) | [updateOne()](https://mongodb.github.io/node-mongodb-native/2.0/api/Collection.html#updateOne) |
| [Limit](https://www.w3schools.com/nodejs/nodejs_mysql_limit.asp) | [Limit](https://www.w3schools.com/nodejs/nodejs_mongodb_limit.asp) | [limit()](https://mongodb.github.io/node-mongodb-native/api-generated/cursor.html#limit) |
| [Join](https://www.w3schools.com/nodejs/nodejs_mysql_join.asp) | [Join](https://www.w3schools.com/nodejs/nodejs_mongodb_join.asp) | [aggregate([{ $lookup: }])](https://mongodb.github.io/node-mongodb-native/api-generated/collection.html?#aggregate) |