MongoDB Practise Sheet

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

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 users (`
`  id MEDIUMINT NOT NULL`
`  AUTO_INCREMENT,`
`  user_id varchar(30),`
`  age Number,`
`  status char(1),`
`  PRIMARY KEY (id)`
`)`
	Implicitly created on first `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")`

`ALTER TABLE users` `ADD join_date DATETIME`	Collections do not describe or enforce the structure of the constituent documents. See Collections do not describe or enforce the structure of the constituent documents
`ALTER TABLE users` `DROP COLUMN join_date`	Collections do not describe or enforce the structure of the constituent documents
`CREATE INDEX idx_user_id_asc` `ON users(user_id)`	`db.users.ensureIndex( { user_id: 1 } )`

SQL Schema Statements	MongoDB Schema Statements
`CREATE INDEX idx_user_id_asc_age_desc ON users(user_id, age DESC)`	`db.users.ensureIndex( { user_id: 1, age: -1 } )`
`DROP TABLE users`	`db.users.drop()`

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
`INSERT INTO users(user_id, age, status) VALUES ("bcd001", 45, "A")`	`db.users.insert( { user_id: "bcd001", age: 45, status: "A" } )`

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
`SELECT * FROM users`	`db.users.find()`
`SELECT id, user_id, status FROM users`	`db.users.find( { }, { user_id: 1, status: 1 } )`

SQL SELECT Statements	MongoDB find() Statements

`SELECT user_id, status FROM users`	`db.users.find(`
	`{ },`
	`{ user_id: 1, status: 1, _id: 0 }`
	`)`

`SELECT * FROM users WHERE status = "A"`	`db.users.find(`
	`{ status: "A" }`
	`)`

`SELECT user_id, status FROM users`
`WHERE status = "A"`
	`db.users.find(`
	`{ status: "A" },`
	`{ user_id: 1, status: 1, _id: 0 }`
	`)`

`SELECT * FROM users`
`WHERE status != "A"`
	`db.users.find(`
	`{ status: { $ne: "A" } }`
	`)`

`SELECT * FROM users`
`WHERE status = "A" AND age = 50`
	`db.users.find(`
	`{ status: "A",`
	`age: 50 }`
	`)`

`SELECT * FROM users`
`WHERE status = "A" OR age = 50`
	`db.users.find(`
	`{ $or: [ { status: "A" } ,`
	`{ age: 50 } ] }`
	`)`

SQL SELECT Statements	MongoDB find() Statements
SELECT * FROM users WHERE age > 25	db.users.find({ age: { $gt: 25 } })
SELECT * FROM users WHERE age < 25	db.users.find({ age: { $lt: 25 } })
SELECT * FROM users WHERE age > 25 AND age <= 50	db.users.find({ age: { $gt: 25, $lte: 50 } })
SELECT * FROM users WHERE user_id like "%bc%"	db.users.find({ user_id: /bc/ })
SELECT * FROM users WHERE user_id like "bc%"	db.users.find({ user_id: /^bc/ })
SELECT * FROM users WHERE status = "A" ORDER BY user_id ASC	db.users.find({ status: "A" }).sort({ user_id: 1 })
SELECT * FROM users WHERE status = "A" ORDER BY user_id DESC	db.users.find({ status: "A" }).sort({ user_id: -1 })

SQL SELECT Statements	MongoDB find() Statements
SELECT COUNT(*) FROM users	db.users.count() *or* db.users.find().count()
SELECT COUNT(user_id) FROM users	db.users.count({ user_id: { $exists: true } }) *or* db.users.find({ user_id: { $exists: true } }).count()
SELECT COUNT(*) FROM users WHERE age > 30	db.users.count({ age: { $gt: 30 } }) *or* db.users.find({ age: { $gt: 30 } }).count()
SELECT DISTINCT(status) FROM users	db.users.distinct("status")
SELECT *FROM sers LIMIT 1	db.users.findOne() *or* db.users.find().limit(1)
SELECT * FROM users LIMIT 5 SKIP 10	db.users.find().limit(5).skip(10)
EXPLAIN SELECT * FROM users WHERE status = "A"	db.users.find({ status: "A" }).explain()

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

`UPDATE users`
`SET status = "C"`
`WHERE age > 25`
	`db.users.update(`
	`{ age: { $gt: 25 } },`
	`{ $set: { status: "C" } },`
	`{ multi: true }`
	`)`

`UPDATE users`
`SET age = age + 3`
`WHERE status = "A"`
	`db.users.update(`
	`{ status: "A" } ,`
	`{ $inc: { age: 3 } },`
	`{ multi: true }`
	`)`

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
`DELETE FROM users` `WHERE status = "D"`	`db.users.remove( { status: "D" } )`
`DELETE FROM users`	`db.users.remove( )`