

# MongoDB Practice Tasks & Test Cases

## Test Case 1: MongoDB Service Down

- **Practice Task:** Attempt to connect while the database engine is stopped.
- **Input:** mongosh or application start command.
- **Steps:**
  1. Ensure the MongoDB service is stopped (systemctl stop mongod or stop via Task Manager).
  2. Attempt to connect via shell or application.
- **Expected Output:** Error Connection refused or Is the server running on host "localhost" (127.0.0.1)

## Test Case 2: Wrong URI

- **Practice Task:** Attempt to connect using an incorrectly formatted string.
- **Input:** mongodb://localhost:9999
- **Steps:**
  1. Open MongoDB Compass or Shell.
  2. Enter a URI with an incorrect port or typo.
- **Expected Output:** Connection error: ECONNREFUSED or Server selection timeout.

## Test Case 3: Authentication Failure

- **Practice Task:** Access a secured database with the wrong credentials.
- **Input:** db.auth("admin", "wrong\_password")
- **Steps:**
  1. Connect to a MongoDB instance with Access Control enabled.
  2. Attempt login with an invalid password.
- **Expected Output:** Error: AuthenticationFailed (Error Code 18).

## Test Case 4: DB Not Exists

- **Practice Task:** Query a database that hasn't been created yet.
- **Input:** use non\_existent\_db -> db.collection.find()
- **Steps:**
  1. Switch to a new database name.
  2. Attempt to read data.

- **Expected Output:** No error; returns an empty result set (Lazy creation).

### Test Case 5: Collection Missing

- **Practice Task:** Attempt to drop a collection that does not exist.
- **Input:** `db.ghost_collection.drop()`
- **Steps:**
  1. Access an existing database.
  2. Run a drop command on a non-existent collection name.
- **Expected Output:** Returns false or ns not found.

### Test case 6: Empty Document Insert

- **Practice Task:** Attempt to insert a null object.
- **Input:** `db.collection.insertOne(null)`
- **Steps:**
  1. Pass null or undefined into the insert function via a driver (Node/Python).
- **Expected Output:** Driver Error: Document must be an object.

### Test Case 7. Duplicate \_id

- **Practice Task:** Insert two documents with the same primary key.
- **Input:** `db.users.insertOne({ _id: 1 })` twice.
- **Steps:**
  1. Insert a document with a manual `_id`.
  2. Attempt to insert another document with the same `_id`.
- **Expected Output:** E11000 duplicate key error.

### Test Case 8. Invalid BSON

- **Practice Task:** Save a data type not supported by BSON.
- **Input:** `{ "field": undefined }`
- **Steps:**

1. Create an object with an undefined value.
  2. Attempt to save via a driver.
- **Expected Output:** Serialization error or field exclusion depending on driver settings.

### Test Case 9: Wrong Query Field

- **Practice Task:** Search for a document using a typo in the field name.
- **Input:** `db.users.find({ user_namme: "Alice" })`
- **Steps:**
  1. Insert `{ username: "Alice" }`.
  2. Query using `user_namme`.
- **Expected Output:** No error; returns 0 results (Logical failure).

### Test Case 10: Update without \$set

- **Practice Task:** Observe behavior of a replacement update.
- **Input:** `db.collection.updateOne({ name: "Alice" }, { age: 30 })`
- **Steps:**
  1. Insert document with multiple fields.
  2. Update without `$set`.
- **Expected Output:** Document is replaced; original fields (except `_id`) are deleted.

### Test Case 11: Delete all Documents

- **Practice Task:** Perform a destructive operation without a filter.
- **Input:** `db.collection.deleteMany({})`
- **Steps:**
  1. Ensure data exists in the collection.
  2. Run `deleteMany` with an empty filter.
- **Expected Output:** `deletedCount` matches total collection size; collection becomes empty.

### Test Case 12: Missing Index

- **Practice Task:** Analyze a query performance on a large dataset.
- **Input:** `db.col.find({ email: "x" }).explain("executionStats")`
- **Steps:**
  1. Populate 10k documents.
  2. Query an unindexed field with `.explain()`.
- **Expected Output:** `winningPlan` shows COLLSCAN (high latency).

### Test Case 13: Network Timeout

- **Practice Task:** Perform a database operation when MongoDB is unreachable
- **Input:** `tasks_collection.find()`
- **Steps:**
  - Stop MongoDB service
  - Run the Flask application
  - Trigger a database read request
- **Expected Output:** PyMongo raises `ServerSelectionTimeoutError`. Handle gracefully.

### Test Case 14: Schema Mismatch

- **Practice Task:** Insert document with unexpected fields
- **Input:** `insert_one({"name":"Task","extra":123})`
- **Steps:**
  - Insert document with extra field
- **Expected Output:** MongoDB allows by default. Enable schema validation.

### Test Case 15: Wrong Datatype

- **Practice Task:** Insert wrong datatype
- **Input:** `insert_one({"priority":"high"})`
- **Steps:**
  - Insert string instead of integer
- **Expected Output:** MongoDB stores wrong type. Validate in application.

### Test Case 16: Aggregation Error

- **Practice Task:** Run invalid aggregation
- **Input:** `aggregate([{"$invalid":{}}])`
- **Steps:**
  - Execute incorrect pipeline
- **Expected Output:** MongoDB throws `OperationFailure`.

### Test Case 17: Cursor Not Closed

- **Practice Task:** Fetch large data without closing cursor
- **Input:** `collection.find()`
- **Steps:**
  - Execute large query
- **Expected Output:** Cursor auto-closes later, but bad practice.

### Test Case 18: Bulk Insert Failure

- **Practice Task:** Bulk insert with duplicate key
- **Input:** `insert_many([...])`
- **Steps:**
  - Include duplicate document
- **Expected Output:** `BulkWriteError` raised.

### Test Case 19: ObjectId Error

- **Practice Task:** Use invalid `ObjectId`
- **Input:** `ObjectId("123")`
- **Steps:**
  - Query with invalid id
- **Expected Output:** Invalid Id error thrown.

### Test Case 20: Replica Set Down

- **Practice Task:** Write when primary down
- **Input:** `insert_one({...})`
- **Steps:**
  - Stop primary node
- **Expected Output:** Write fails until failover.

### Test Case 21: Read Preference Error

- **Practice Task:** Read from unavailable secondary
- **Input:** readPreference=secondary
- **Steps:**
  - Shut secondary node
- **Expected Output:** Read preference error.

### Test Case 22: Write Concern Failure

- **Practice Task:** Write with majority concern
- **Input:** w=majority
- **Steps:**
  - Replica unavailable
- **Expected Output:** WriteConcernError occurs.

### Test Case 23: Disk Full

- **Practice Task:** Write when disk full
- **Input:** insert\_one({...})
- **Steps:**
  - Exhaust disk space
- **Expected Output:** MongoDB rejects write.

### Test Case 24: Unauthorized Access

- **Practice Task:** Delete without permission
- **Input:** delete\_one()
- **Steps:**
  - Use limited role
- **Expected Output:** Unauthorized error.

### Test Case 25: Version Mismatch

- **Practice Task:** Client-server version mismatch
- **Input:** Unsupported feature call
- **Steps:**
  - Use old PyMongo
- **Expected Output:** Feature fails or errors.