**User Management Queries**

1. db.createUser({ user: "admin", pwd: "securePass", roles: ["root"] })
2. db.createUser({ user: "developer", pwd: "devPass", roles: [{ role: "readWrite", db: "devDB" }] })
3. db.createUser({ user: "analyst", pwd: "analystPass", roles: [{ role: "read", db: "analyticsDB" }] })
4. db.dropUser("developer")
5. db.dropAllUsers()
6. db.getUsers()
7. db.getUser("admin")
8. db.updateUser("developer", { roles: [{ role: "dbAdmin", db: "devDB" }] })
9. db.changeUserPassword("analyst", "newSecurePass")
10. db.createUser({ user: "backupUser", pwd: "backup123", roles: [{ role: "backup", db: "admin" }] })

**Role Management Queries**

1. db.createRole({ role: "customRole", privileges: [{ resource: { db: "testDB", collection: "" }, actions: ["find", "insert"] }], roles: [] })
2. db.getRoles({ showPrivileges: true })
3. db.getRole("customRole", { showPrivileges: true })
4. db.dropRole("customRole")
5. db.updateRole("customRole", { privileges: [{ resource: { db: "testDB", collection: "users" }, actions: ["find"] }] })
6. db.grantRolesToUser("analyst", [{ role: "readWrite", db: "analyticsDB" }])
7. db.revokeRolesFromUser("analyst", [{ role: "readWrite", db: "analyticsDB" }])
8. db.grantRolesToRole("customRole", [{ role: "read", db: "testDB" }])
9. db.revokeRolesFromRole("customRole", [{ role: "read", db: "testDB" }])
10. db.dropAllRoles()

**Authorization Queries**

1. db.auth("admin", "securePass")
2. db.runCommand({ connectionStatus: 1 })
3. db.runCommand({ usersInfo: "admin" })
4. db.runCommand({ rolesInfo: "read" })
5. db.runCommand({ grantRolesToUser: "developer", roles: [{ role: "dbOwner", db: "devDB" }] })
6. db.runCommand({ revokeRolesFromUser: "developer", roles: [{ role: "dbOwner", db: "devDB" }] })
7. db.runCommand({ grantRolesToRole: "customRole", roles: [{ role: "readWrite", db: "testDB" }] })
8. db.runCommand({ revokeRolesFromRole: "customRole", roles: [{ role: "readWrite", db: "testDB" }] })
9. db.runCommand({ invalidateUserCache: 1 })
10. db.runCommand({ invalidateRoleCache: 1 })

**Database and Collection-Level Access Control**

1. db.runCommand({ createUser: "readonlyUser", pwd: "readonlyPass", roles: [{ role: "read", db: "reportingDB" }] })
2. db.runCommand({ updateUser: "readonlyUser", roles: [{ role: "readWrite", db: "reportingDB" }] })
3. db.runCommand({ dropUser: "readonlyUser" })
4. db.runCommand({ createRole: "customDBAdmin", privileges: [{ resource: { db: "salesDB", collection: "" }, actions: ["find", "insert", "update"] }], roles: [] })
5. db.runCommand({ dropRole: "customDBAdmin" })
6. db.runCommand({ grantPrivilegesToRole: "customDBAdmin", privileges: [{ resource: { db: "salesDB", collection: "orders" }, actions: ["find"] }] })
7. db.runCommand({ revokePrivilegesFromRole: "customDBAdmin", privileges: [{ resource: { db: "salesDB", collection: "orders" }, actions: ["find"] }] })
8. db.runCommand({ usersInfo: 1 })
9. db.runCommand({ rolesInfo: 1 })
10. db.runCommand({ listDatabases: 1 })

**Authentication Mechanism Queries**

1. db.adminCommand({ setParameter: 1, authenticationMechanisms: "SCRAM-SHA-256" })
2. db.adminCommand({ setParameter: 1, authenticationMechanisms: "MONGODB-X509" })
3. db.adminCommand({ logLevel: 2 })
4. db.adminCommand({ getParameter: 1, authenticationMechanisms: 1 })
5. db.adminCommand({ logApplicationMessage: "Security Audit Log Enabled" })
6. db.runCommand({ logLevel: 0 })
7. db.runCommand({ usersInfo: { forAllDBs: true } })
8. db.runCommand({ rolesInfo: { showPrivileges: true, showBuiltinRoles: true } })
9. db.getSiblingDB("admin").system.users.find().pretty()
10. db.getSiblingDB("admin").system.roles.find().pretty()