1. **Crea una colección inicial llamada logs que contenga registros aleatorios como datos JSON.**

// Crear la colección 'logs' y insertar registros aleatorios

db.createCollection('customers', { capped: false });

db.customers.insert([

    {

        "customer\_name": "John Doe"

    },

    {

        "customer\_name": "Jane Smith"

    },

    {

        "customer\_name": "Alice Johnson"

    },

    {

        "customer\_name": "Bob Brown"

    },

    {

        "customer\_name": "Carol White"

    }

]);

db.createCollection('products', { capped: false });

db.products.insert([

    {

        "product\_name": "Laptop",

        "product\_price": "1200.50"

    },

    {

        "customer\_name": "Smartphone",

        "product\_price": "800.75"

    },

    {

        "customer\_name": "Tablet",

        "product\_price": "300.00"

    },

    {

        "customer\_name": "Monitor",

        "product\_price": "150.99"

    },

    {

        "customer\_name": "Keyboard",

        "product\_price": "50.00"

    }

]);

db.createCollection('orders', { capped: false });

db.orders.insert([

    {

        "customer\_id": "1",

        "product\_id": "1",

        "order\_date": "2024-01-01 10:00:00"

    },

    {

        "customer\_id": "2",

        "product\_id": "2",

        "order\_date": "2024-01-02 11:30:00"

    },

    {

        "customer\_id": "3",

        "product\_id": "3",

        "order\_date": "2024-01-03 15:45:00"

    },

    {

        "customer\_id": "4",

        "product\_id": "4",

        "order\_date": "2024-01-04 09:20:00"

    },

    {

        "customer\_id": "5",

        "product\_id": "5",

        "order\_date": "2024-01-05 14:10:00"

    }

]);

db.createCollection('logs', { capped: false });

db.logs.insert([

    {

        "timestamp": new Date(),

        "level": "INFO",

        "message": "El sistema se ha iniciado correctamente.",

        "userId": Math.floor(Math.random() \* 1000),

        "sessionId": Math.random().toString(36).substring(2, 15)

    },

    {

        "timestamp": new Date(),

        "level": "ERROR",

        "message": "Error al procesar la solicitud de usuario.",

        "userId": Math.floor(Math.random() \* 1000),

        "sessionId": Math.random().toString(36).substring(2, 15)

    },

    {

        "timestamp": new Date(),

        "level": "WARNING",

        "message": "El uso de memoria ha alcanzado el 80%.",

        "userId": Math.floor(Math.random() \* 1000),

        "sessionId": Math.random().toString(36).substring(2, 15)

    },

    {

        "timestamp": new Date(),

        "level": "DEBUG",

        "message": "La función de inicio se está ejecutando.",

        "userId": Math.floor(Math.random() \* 1000),

        "sessionId": Math.random().toString(36).substring(2, 15)

    },

    {

        "timestamp": new Date(),

        "level": "INFO",

        "message": "Usuario ha realizado una compra.",

        "userId": Math.floor(Math.random() \* 1000),

        "sessionId": Math.random().toString(36).substring(2, 15)

    },

    {

        "timestamp": new Date(),

        "level": "ERROR",

        "message": "No se pudo conectar a la base de datos.",

        "userId": Math.floor(Math.random() \* 1000),

        "sessionId": Math.random().toString(36).substring(2, 15)

    },

    {

        "timestamp": new Date(),

        "level": "INFO",

        "message": "El usuario ha cerrado la sesión.",

        "userId": Math.floor(Math.random() \* 1000),

        "sessionId": Math.random().toString(36).substring(2, 15)

    },

    {

        "timestamp": new Date(),

        "level": "DEBUG",

        "message": "Se ha procesado un pedido a través de la API.",

        "userId": Math.floor(Math.random() \* 1000),

        "sessionId": Math.random().toString(36).substring(2, 15)

    },

    {

        "timestamp": new Date(),

        "level": "WARNING",

        "message": "Se ha alcanzado el límite de intentos fallidos de acceso.",

        "userId": Math.floor(Math.random() \* 1000),

        "sessionId": Math.random().toString(36).substring(2, 15)

    },

    {

        "timestamp": new Date(),

        "level": "INFO",

        "message": "La aplicación está en ejecución y esperando solicitudes.",

        "userId": Math.floor(Math.random() \* 1000),

        "sessionId": Math.random().toString(36).substring(2, 15)

    }

]);

// Confirmar que se han insertado los registros

print("Registros insertados en la colección 'logs'.");