MongoDB Script for Inserting Sample Data and Setting Collection Validation

3. Insert Sample Data:

To insert two sample documents into the **products** collection with different **productName** and **price**, use the following MongoDB script:

```
// Insert two sample products into the "products" collecti
db.products.insertMany([
    { productName: "Smartwatch", price: 250 },
    { productName: "Headphones", price: 150 }
]);
```

This script inserts two documents into the **products** collection, each containing the following fields:

- **productName** (string)
- **price** (integer)

---

4. Set Collection Validation for "users" Collection:

To enforce validation rules on the **users** collection, where the **username** field must be a string and the **email** field must be unique, use the following MongoDB script:

```
// Enable validation rules for the "users" collection
db.createCollection("users", {
    validator: {
        $jsonSchema: {
            bsonType: "object",
            required: ["username", "email"],
            properties: {
                username: {
                    bsonType: "string",
                    description: "Username must be a string
                },
                email: {
                    bsonType: "string",
                    unique: true,
                    description: "Email must be a string an
                }
            }
        }
    }
});
```

Explanation:

1. **`db.createCollection("users", {...})`:** Creates the **users** collection in the **ecommerce** database with validation rules.
2. **`$jsonSchema`:** Defines the schema validation for the collection.
   - **`required`** : Specifies that both **username** and **email** are required fields.
   - **`username`** : Ensures that the **username** field is a string.
   - **`email`** : Ensures that the **email** field is a string and enforces uniqueness (this requires a separate index to be created on the **email** field).
3. **Note**: To enforce the **unique** constraint on the **email** field, you need to create a unique index on the **email** field separately:

```
// Create a unique index on the "email" field in the "users
db.users.createIndex({ email: 1 }, { unique: true });
```

Example Usage After Script Execution:

- **Inserting a document into the "users" collection (should pass validation):**

```
db.users.insertOne({
    username: "john_doe",
    email: "john.doe@example.com"
});
```

- **Inserting a document with a duplicate email (should fail validation):**

```
db.users.insertOne({
    username: "jane_doe",
    email: "john.doe@example.com"  // This email already
});
```

The above script ensures that:

- The **username** is always a string.
- The **email** is a string and must be unique across the **users** collection.