In MongoDB, you can enforce a schema on a collection using the $jsonSchema operator in a validator expression. When you create a collection with a validator expression that includes $jsonSchema, MongoDB will enforce the specified schema on all documents inserted into the collection.

Here is an example of how to create a collection with a schema using $jsonSchema:

db.createCollection("products", {

validator: {

$jsonSchema: {

bsonType: "object",

required: [ "product\_name", "category", "price", "stock\_quantity" ],

properties: {

product\_name: {

bsonType: "string",

description: "must be a string and is required"

},

category: {

bsonType: "string",

description: "must be a string and is required"

},

price: {

bsonType: "double",

minimum: 0,

description: "must be a double and is required"

},

stock\_quantity: {

bsonType: "int",

minimum: 0,

description: "must be an integer and is required"

}

}

}

}

})

In this example, we are creating a collection called "products" with a validator expression that includes a $jsonSchema object. The $jsonSchema object specifies the required properties for each document in the collection, as well as the data type and constraints for each property.

For example, the product\_name property must be a string and is required, while the price property must be a double and greater than or equal to 0. If a document is inserted into the "products" collection that does not conform to this schema, MongoDB will reject the document and return an error message.\

Overall, using $jsonSchema in a validator expression when creating a collection in MongoDB allows you to enforce a schema on the collection and ensure that all documents inserted into the collection adhere to that schema.