PRACTICAL 7

Implementing data validation rules using MongoDB's schema validation feature

In mongosh:

```
> use Yesha
< already on db   Yesha
> db.createCollection("notification", {
    validator: {
      $jsonSchema: {
        required: ["name", "price"],
        properties: {
          name: {
            bsonType: "string",
            description: "must be a string"
          },
          price: {
            bsonType: "double",
            description: "must be a number"
          }
        }
      }
    },
    validationAction: "warn"
  });
< { ok: 1 }
```

```
> // This works because 'name' is a string and 'price' is a number.
  db.notification.insertOne({
    name: "Summer Sale Announcement",
    price: 29.99
  });
< {
    acknowledged: true,
    insertedId: ObjectId('68ef1551543d822520f4a126')
  }
> // This also works. Extra fields are allowed as long as the required ones are present.
  db.notification.insertOne({
    name: "New Product Launch",
    price: 15.0,
    inStock: true,
    category: "Electronics",
    acknowledged: true,
    insertedId: ObjectId("68ed3787e1e59f2bdbcebea5")
  });
< {
    acknowledged: true,
    insertedId: ObjectId('68ef1558543d822520f4a127')
  }
```

```
> // FAIL: The 'price' field is missing, which is required by the schema.
  db.notification.insertOne({
    name: "Flash Alert"
  });

  // FAIL: The 'price' field is a string ("59.00") instead of a number.
  db.notification.insertOne({
    name: "Daily Deal",
    price: "59.00"
  });

  // FAIL: The required 'name' field is missing.
  db.notification.insertOne({
    price: 99.99,
    message: "This won't work"
  });
✖ ▸ MongoServerError: Document failed validation
```

```
db.runCommand({
  collMod: "notification",
  validator: {
    $jsonSchema: {
      required: ["name", "price", "author"],
      properties: {
        name: {
          bsonType: "string",
          description: "must be a string"
        },
        price: {
          bsonType: "number",
          description: "must be a number"
        },
        author: {
          bsonType: "array",
          description: "must be an array",
          items: {
            bsonType: "object",
            required: ["name", "email"],
            properties: {
              name: {
                bsonType: "string",
                description: "must be a string"
              },
              email: {
                bsonType: "string",
                description: "must be a string"
              }
            }
```

```
          }
        }
      }
    }
  },
  validationAction: "error"
});
```

```
< { ok: 1 } >
  Yesha>
```

```
> // This works because 'author' is an array of objects,
  // and each object has a string 'name' and 'email'.
  db.notification.insertOne({
    name: "API Documentation Update",
    price: 29.99,
    author: [
      {
        name: "The Dev Team",
        email: "devs@example.com"
      }
    ]
  });
< {
    acknowledged: true,
    insertedId: ObjectId('68ef1869543d822520f4a131')
  }
Yesha| >
```