Version 2

task 22: baar baar seed hone se bachne ke liye we will deleteMany() pehle.

```
//seed.js

async function seedDB(){
    await Product.deleteMany({}); //added here
    await Product.insertMany(products);
    console.log("data seeded successfully")
}
```

task 23: ab price is in number and humne text bheja hua hai so have to change the type taaki hum usse string mei bhi la paae and decimal values store kr paae**.**

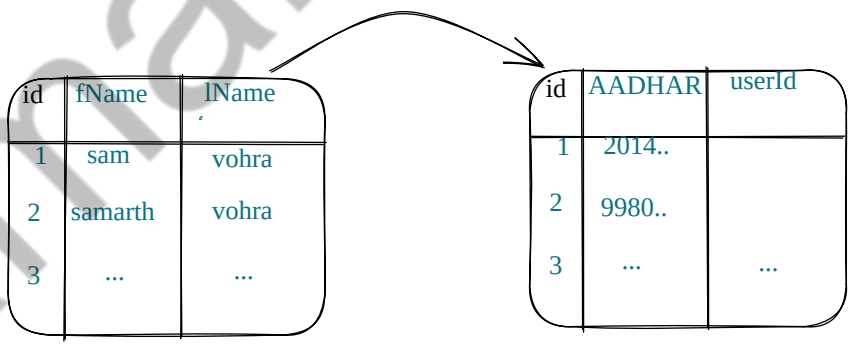
```
//views --> products --> new.ejs
<div class="input-group mb-3">
                    <span class="input-group-text" id="basic-addon1">Rs. </span>
                    <input class="form-control" step="any" type="number" name="price" id="paisa" placeholder="Price of Product">
                </div>
```

aapne aap typecasting hojaaegi need nhi aaegi aapko.

task 24: pehle hum thoda mogoose relationship samajh lete hai and then we will proceed.

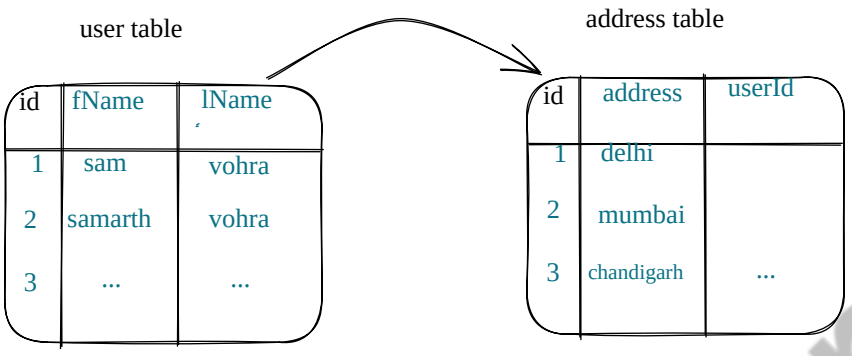
we have different type of relationship:

1. one to one (1:1)



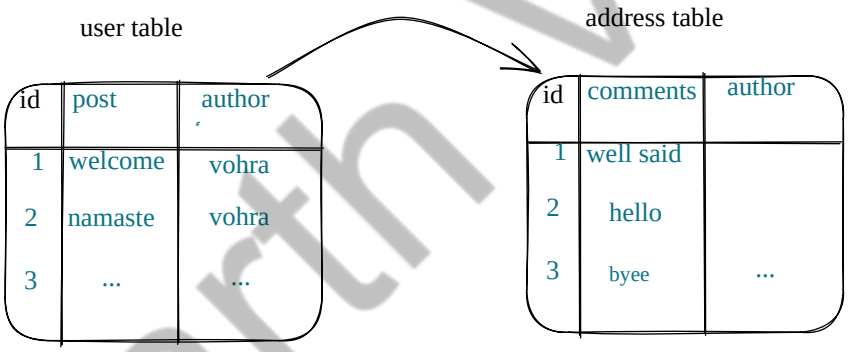
here we cannot have multiple addhar for one person
so it simply signifies one-one relationship

2. one to few (here few means less i.e 3-4 se zyada address nhi hote, so one to few)



here one user can have many address so it is one to few realtionship in this case.

3. one to many (1:N)



here one post can have many comments on it so it is one to many realtionship in this case.

4. m many to one
5. many to many

(majorly you will get one to few or one to many relationship)

task 25: (ignore krdo baccho)

adding user's address as array by creating schema.

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://127.0.0.1:27017/relationDB')
.then(()=>{console.log("Db connected")})
.catch((err)=>{console.log(err)})
```

```
const userSchema = new mongoose.Schema({
    name:String ,
    age:Number ,
    addresses:[
        {
            _id:(id:false),
            lane: String ,
            city: String ,
            state: String ,
            country : String
        }
    ]
});

const User = mongoose.model('User' , userSchema );

const makeUser = async()=>{
   const user = new User({
        name:"samarth Vohra" ,
        age:27
    })
    await user.save();
    console.log("user successfully saved")
    console.log(user);
}

// makeUser();

const addAdress = async(id)=>{
    const user = await User.findById(id);
    user.addresses.push({
        lane: "G-157" ,
        city: "new delhi" ,
        state: "delhi" ,
        country : 'india'
    })

    await user.save();
    console.log(user)

}

addAdress('640e330a2e7f40db7d70c77e');
```

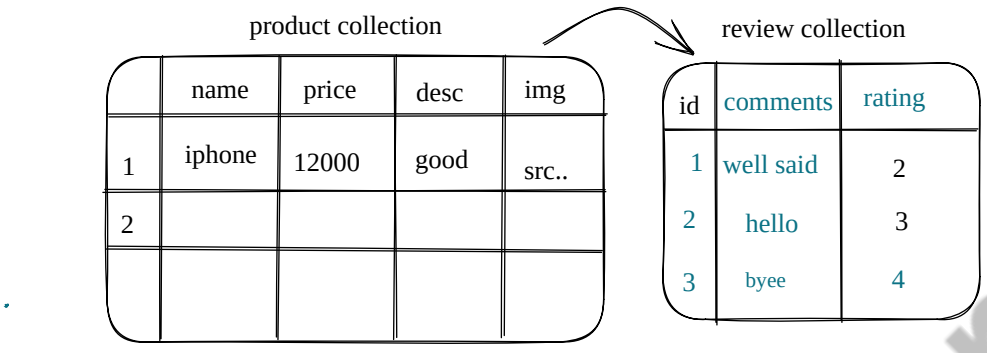
task 26:

Ab hume review add krna hai and review is also our 1:many relationship.

1. **make review.js**. file in model ka folder.

ab agar mai product ke model ke andar junga and vahan add krunga to that will be a case of one to few relationship lekin ab i want ki reviews to 1:many relationship hai to we will make reviews ka model alag se.

ab 2 table banenege



Ab hume in 2 collections ko apas mei kuch relation se jodhna hai taaki pta chal jaae kis product par kaunsa review aaya hai.

```
//review.js

const mongoose =  require('mongoose');

const reviewSchema = new mongoose.Schema({
    rating:{
        type:Number,
        min:0,
        max:5

    },
    comment:{
        type:String,
        trim:true
    }
})
let Review = mongoose.model('Review' , reviewSchema);
module.exports = Review;
```

Task 27: ab mai kya krunga mai product ke schema ke andar ek **reviews ki array** banaunga jiska kaam hoga ki us particular product ke upar jo jo review kia gya hai us review ko mai review ki id ki madad se inside the array store krdunga taaki jo jo review us product par hai vo hume using id mil jaae.

```
//product.js

desc:{
        type:String,
        trim:true
     },
reviews:[   //adding this
    {
        type:mongoose.Schema.Types.ObjectId,
        ref:'Review'
    }
]
```

task 28:

ab hume routes banane hai for the review to hume **review.js** krke file banani hai inside routes folder.

```
//routes --> review.js

const express =  require('express');
const Product = require('../models/product');
const router = express.Router();

router.post('/products/:id/review' , (req,res)=>{
    console.log(req.body);
    res.send("review route")
})


module.exports = router;
```

task 29: hume show.ejs mei bhi to input dene hai to get the values of rating.

```
//view --> product --> show.ejs

//adding here
<div class="col-lg-4 mt-5">
        <h2 class="display-5">Leave your review</h2>
        <form action="/products/<%= foundProduct._id %>/review" method="POST">
            <div class="mb-3">
                <label class="form-label" for="stars">Rating: </label>
                <input class="form-control" type="range" min="0" max="5" id="stars" name="rating">
            </div>
            <div class="mb-3">
                <label class="form-label" for="comment">Comment: </label>
                <textarea class="form-control" name="comment" id="comment" rows="3"></textarea>
            </div>
            <button class="btn btn-sm btn-success">Submit</button>
        </form>
    </div>
```

task 30: ab hume review route ko bhi app.js ke andar as a review hi to dekhna hai.

```
//app.js

const methodOverride  = require('method-override');
const reviewRoutes = require("./routes/review"); //added


// Routes
app.use(productRoutes);
app.use(reviewRoutes); //added
```

ab simply **review model** ko bhi require kro and start working on it.

ab jo id nikaal rha hu i.e product ki id and review ki id us product ki id mei store honi chahiye.

```
//review.js

const express =  require('express');
const Product = require('../models/product');
const Review = require('../models/review');

const router = express.Router();

router.post('/products/:id/review' , async(req,res)=>{
    let {productId} = req.params;
    let {rating , comment} = req.body;
```

```
    const product = await Product.findById(productId);
    // creating a new review
    let review  = new Review({rating , comment}) // let review  = new Review({...req.body})

    // adding review id to product array
    product.reviews.push(review); //mongodb internally isme se id nikaal kr usse push krdega.

    await review.save();
    await product.save();
})


module.exports = router;
```

—> **Now see ki review push hua ya nhi hua in the array**

```
//terminal
yahan par dono colletions check kro you will be able to see
```

task 31:

ab hume simply show krna hai humare rendered reviews iske liye hum populate ka use krenge.

hum simply jahan par ek particular product display hora tha **(show.ejs)** ussi jagah par we need to populate this review.

> 💡 **populate** kya krega simply aapne jo object id store kari hai usko populate krega and us id ki jagah saara ka ssara comment and rating show krdega in place of that objectId.

```
//productRoutes.js
//go to show waala route


// route for shwoing the deatails of thre products
router.get('/products/:id' , async(req,res)=>{
    let {id} = req.params;
    // let foundProduct = await Product.findById(id);
    let foundProduct = await Product.findById(id).populate('reviews'); //changed
    console.log(foundProduct); //added
    res.render('products/show' , {foundProduct});

})
```

above way se we can join these 2 collections.

task 32:

Ab mere paas **foundProduct** to hai hi, to us **foundProduct** ke andar se reviews ke array use krke simply **show.ejs** ke andar jaakr mai simply show kar sakta hu.

```
//show.ejs

<button class="btn btn-sm btn-success">Submit</button>
        </form>

// added form here for showing reviews
        <div class="my-3">
```

```
           <% for(let review of foundProduct.reviews){ %>

            <div class="card mb-3">
                <div class="card-header">Rating: <%=review.rating%></div>
                <div class="card-body">
                  <!-- <h5 class="card-title">Special title treatment</h5> -->
                  <p class="card-text">Comment: <%=review.comment%></p>
                  <button class="btn btn-danger">Delete</button>
                </div>
              </div>

           <% } %>
        </div>
```

ab hume redirect bhi krna hai to see the show.ejs page.

```
//review.js (routes)

router.post('/products/:productId/review' , async(req,res)=>{

        let {productId} = req.params;
        let {rating , comment} = req.body;
        const product = await Product.findById(productId);
        console.log(product);
        // creating a new review
        const review  = new Review({rating , comment});

        // adding review id to product array
        product.reviews.push(review);
        await review.save();
        await product.save();
        res.redirect(`/products/${productId}`) //added here

})
```

task 33:

Ab hum idhar star rating add krenge using a **libary starability.css**

```
//show.ejs

<div class="col-lg-4 mt-5">
        <h2 class="display-5">Leave your review</h2>
        <form action="/products/<%= foundProduct._id %>/review" method="POST">
            <div class="mb-3">
                <label class="form-label" for="stars">Rating: </label>
                <fieldset class="starability-basic">
                    <!-- <legend>First rating:</legend> -->below adding from library
                    <input type="radio" id="no-rate" class="input-no-rate" name="rating" value="0" checked aria-label="No rating." />
                    <input type="radio" id="first-rate1" name="rating" value="1" />
                    <label for="first-rate1" title="Terrible">1 star</label>
                    <input type="radio" id="first-rate2" name="rating" value="2" />
                    <label for="first-rate2" title="Not good">2 stars</label>
                    <input type="radio" id="first-rate3" name="rating" value="3" />
                    <label for="first-rate3" title="Average">3 stars</label>
                    <input type="radio" id="first-rate4" name="rating" value="4" />
                    <label for="first-rate4" title="Very good">4 stars</label>
                    <input type="radio" id="first-rate5" name="rating" value="5" />
                    <label for="first-rate5" title="Amazing">5 stars</label>
                  </fieldset>
                <!-- <input class="form-control" type="range" min="0" max="5" id="stars" name="rating"> -->
            </div>
            <div class="mb-3">
                <label class="form-label" for="comment">Comment: </label>
                <textarea class="form-control" name="comment" id="comment" rows="3"></textarea>
            </div>
            <button class="btn btn-sm btn-success">Submit</button>
        </form>

        <div class="my-3">
           <% for(let review of foundProduct.reviews){ %>

            <div class="card mb-3">
```

```
                <div class="card-header">Rating: <%=review.rating%></div>
                <div class="card-body">
                  <!-- <h5 class="card-title">Special title treatment</h5> -->
                  <p class="card-text">Comment: <%=review.comment%></p>
                  <button class="btn btn-danger">Delete</button>
                </div>
              </div>

            <% } %>
          </div>


    </div>
```

task 34: ab hume css bhi add krni hai to we can do one thing css folder ke andar **star.css** file banalo

```
//css --> star.css

.starability-result {
  position: relative;
  width: 150px;
  height: 30px;
  background-image: url("
  font-size: 0.1em;
  color: transparent;
}

.starability-result:after {
  content: ' ';
  position: absolute;
  left: 0;
  height: 30px;
  background-image: url("
  background-position: 0 -30px;
}

.starability-result[data-rating="5"]::after {
  width: 150px;
}

.starability-result[data-rating="4"]::after {
  width: 120px;
}

.starability-result[data-rating="3"]::after {
  width: 90px;
}

.starability-result[data-rating="2"]::after {
  width: 60px;
}

.starability-result[data-rating="1"]::after {
  width: 30px;
}

@media screen and (-webkit-min-device-pixel-ratio: 2), screen and (min-resolution: 192dpi) {
  .starability-result {
    background-image: url("
    background-size: 30px auto;
  }
  .starability-result:after {
    background-image: url("
    background-size: 30px auto;
  }
}

.starability-basic {
  display: block;
  position: relative;
  width: 150px;
  min-height: 60px;
  padding: 0;
  border: none;
}
```

```
.starability-basic > input {
  position: absolute;
  margin-right: -100%;
  opacity: 0;
}

.starability-basic > input:checked ~ label,
.starability-basic > input:focus ~ label {
  background-position: 0 0;
}

.starability-basic > input:checked + label,
.starability-basic > input:focus + label {
  background-position: 0 -30px;
}

.starability-basic > input[disabled]:hover + label {
  cursor: default;
}

.starability-basic > input:not([disabled]):hover ~ label {
  background-position: 0 0;
}

.starability-basic > input:not([disabled]):hover + label {
  background-position: 0 -30px;
}

.starability-basic > input:not([disabled]):hover + label::before {
  opacity: 1;
}

.starability-basic > input:focus + label {
  outline: 1px dotted #999;
}

.starability-basic .starability-focus-ring {
  position: absolute;
  left: 0;
  width: 100%;
  height: 30px;
  outline: 2px dotted #999;
  pointer-events: none;
  opacity: 0;
}

.starability-basic > .input-no-rate:focus ~ .starability-focus-ring {
  opacity: 1;
}

.starability-basic > label {
  position: relative;
  display: inline-block;
  float: left;
  width: 30px;
  height: 30px;
  font-size: 0.1em;
  color: transparent;
  cursor: pointer;
  background-image: url("
  background-repeat: no-repeat;
  background-position: 0 -30px;
}

.starability-basic > label::before {
  content: '';
  position: absolute;
  display: block;
  height: 30px;
  background-image: url("
  background-position: 0 30px;
  pointer-events: none;
  opacity: 0;
}

.starability-basic > label:nth-of-type(5)::before {
  width: 120px;
  left: -120px;
}

.starability-basic > label:nth-of-type(4)::before {
```

```
  width: 90px;
  left: -90px;
}

.starability-basic > label:nth-of-type(3)::before {
  width: 60px;
  left: -60px;
}

.starability-basic > label:nth-of-type(2)::before {
  width: 30px;
  left: -30px;
}

.starability-basic > label:nth-of-type(1)::before {
  width: 0px;
  left: 0px;
}

@media screen and (-webkit-min-device-pixel-ratio: 2), screen and (min-resolution: 192dpi) {
  .starability-basic > label {
    background-image: url("
    background-size: 30px auto;
  }
}

@media screen and (-ms-high-contrast: active) {
  .starability-basic {
    width: auto;
  }
  .starability-basic > input {
    position: static;
    margin-right: 0;
    opacity: 1;
  }
  .starability-basic .input-no-rate {
    display: none;
  }
  .starability-basic > label {
    display: inline;
    float: none;
    width: auto;
    height: auto;
    font-size: 1em;
    color: inherit;
    background: none;
  }
  .starability-basic > label::before, .starability-basic > label::after {
    display: none;
  }
}
```

ab boiler plate par bhi to add kro is css ka link.

```
//layouts --> boilerplate.ejs

<link rel="stylesheet" href="/css/app.css">
<link rel="stylesheet" href="/css/star.css"> //added
```

task 35:

ab task mei hume star hi to show krne hai rather than showing rating as number so simply use the same library anc scroll down. (showing the static result)

```
//show.ejs
<div class="my-3">
          <% for(let review of foundProduct.reviews){ %>

           <div class="card mb-3"> //changed below
               <!-- <div class="card-header">Rating: <%=review.rating%></div> -->
               <div class="card-body">
```

```
                    <p class="starability-result" data-rating="<%=review.rating%>">
                        Rated: <%=review.rating%> stars //changed here
                     </p>
                  <!-- <h5 class="card-title">Special title treatment</h5> -->
                  <p class="card-text">Comment: <%=review.comment%></p>
                  <button class="btn btn-danger">Delete</button>
                </div>
              </div>

            <% } %>
 </div>
```

task36: ab hume time of comment bhi add krna hai to hum simply **review ke model** ke andar ek object bhejdenge **timestamps** ka jiski madad se we can conclude the values **updatedAt** adn **createdAt.**

```
//review.js (schema)

comment:{
        type:String,
        trim:true
    }
} , {timestamps:true}) //adding here
```

now show at the databse that are they showing the value

ab merko is timestamp ko months and years mei krna hai change so uske liye ill make use of our stackoverflow. **(timestamp to date js) —>** toDateString();

```
//show.ejs
<div class="card mb-3">
                <!-- <div class="card-header">Rating: <%=review.rating%></div> -->
                <div class="card-body">
                    <p class="starability-result" data-rating="<%=review.rating%>">
                        Rated: <%=review.rating%> stars
                     </p>
                  <!-- <h5 class="card-title">Special title treatment</h5> -->

                  //adding code below
                  <p class="card-text">Comment: <%=review.comment%></p>
                  <%if(review.createdAt){%>
                    <p> <%=review.createdAt.toDateString()%></p>
                  <%}%>
                  <button class="btn btn-danger">Delete</button>
                </div>
</div>
```