

Scenario

3. Imagine you work for amazon, what is the Meta data information you will store for an item in your Database. For E.g. the item is a shirt, once you have stored the Meta data how will use the information?

Metadata to Store for an Item (e.g., Shirt)

When storing a product like a shirt in Amazon’s database, you would typically store the following **metadata attributes**:

Attribute	Description
item_id	Unique identifier for the product
title	Name of the shirt (e.g., "Men's Cotton Casual Shirt")
brand	Brand name (e.g., "Levi's", "Arrow")
category	Product category (e.g., "Clothing > Men > Shirts")
description	Detailed product description
material	Material used (e.g., "100% Cotton")
color	Available colors (e.g., "Blue", "Black")
size	Available sizes (e.g., "S, M, L, XL")
gender	Target gender (e.g., "Men", "Women", "Unisex")
price	Current price
discount	Discount percentage or amount
stock_quantity	Number of items available in stock
rating	Average user rating (e.g., 4.5 stars)
reviews_count	Number of reviews
image_urls	Links to product images
shipping_details	Estimated delivery time, shipping cost
return_policy	Return eligibility and time frame
created_at / updated_at	Timestamps for record keeping

How the Metadata is Used

1. Search & Discovery

- Users searching for "blue shirt" or "cotton shirt" will be shown results using fields like title, color, and material.

2. Filtering and Sorting

- On category pages, filters for **brand**, **price range**, **size**, **color**, etc., depend on this metadata.
- Sorting by **price**, **rating**, or **popularity** uses fields like `price`, `rating`, and `reviews_count`.

3. Product Recommendations

- Based on `category`, `brand`, and `user behavior`, Amazon suggests related or frequently bought together items.

4. Inventory Management

- `stock_quantity` helps track available stock and triggers restocking or hiding out-of-stock items.

5. Pricing and Promotions

- Metadata like `price`, `discount`, and `brand` are used to manage deals, flash sales, and promotions.

6. Analytics and Reporting

- Metadata helps in understanding buying patterns, regional demand, or performance of specific brands/categories.

7. Personalization

- Metadata (like `gender`, `category`, `brand`) helps tailor product listings to individual users based on past behavior.

8. Order Fulfillment & Logistics

- `shipping_details` and `stock_quantity` help in determining the fastest warehouse to fulfill an order from.

9. Customer Experience

- Rich `description`, `images`, `ratings`, and `reviews` help customers make informed decisions.

Here's a complete **Node.js + MongoDB** setup to store item metadata (e.g., a shirt on Amazon) with:

1. Mongoose schema
2. CRUD APIs
3. 10 dummy items inserted

```
amazon-items-api/
├── server.js
├── models/
│   └── Item.js
├── routes/
│   └── itemRoutes.js
└── data/
    └── seed.js
```

models/[Item.js](#)

```
const mongoose = require('mongoose');
```

```
const itemSchema = new mongoose.Schema({
  title: String,
  brand: String,
  category: String,
  description: String,
  material: String,
  color: [String],
  size: [String],
  gender: String,
  price: Number,
  discount: Number,
  stock_quantity: Number,
  rating: Number,
  image_urls: [String],
  created_at: { type: Date, default: Date.now }
});
```

```
module.exports = mongoose.model('Item', itemSchema);
```

----->

[routes/itemRoutes.js](#)

```
const express = require('express');
const router = express.Router();
const Item = require('../models/Item');
```

```
// Create new item
```

```
router.post('/', async (req, res) => {
  const item = new Item(req.body);
  await item.save();
  res.status(201).json(item);
});
```

```
// Get all items
```

```
router.get('/', async (req, res) => {
  const items = await Item.find();
  res.json(items);
});
```

```
// Get single item
```

```
router.get('/:id', async (req, res) => {
  const item = await Item.findById(req.params.id);
  res.json(item);
});
```

```
// Update item
```

```
router.put('/:id', async (req, res) => {
  const item = await Item.findByIdAndUpdate(req.params.id, req.body, { new: true });
  res.json(item);
});
```

```
// Delete item
```

```
router.delete('/:id', async (req, res) => {
  await Item.findByIdAndDelete(req.params.id);
});
```

```
res.json({ message: 'Item deleted' });
});
```

```
module.exports = router;
```

----->

server.js

```
const express = require('express');
const mongoose = require('mongoose');
const itemRoutes = require('./routes/itemRoutes');
```

```
const app = express();
app.use(express.json());
```

```
mongoose.connect('mongodb://localhost:27017/amazonmeta', {
  useNewUrlParser: true,
  useUnifiedTopology: true
}).then(() => console.log('MongoDB connected'));
```

```
app.use('/api/items', itemRoutes);
```

```
const PORT = 3000;
app.listen(PORT, () => console.log(`Server running on http://localhost:\${PORT}`));
```

----->

data/[seed.js](#)

```
const mongoose = require('mongoose');
const Item = require('./models/Item');
```

```
mongoose.connect('mongodb://localhost:27017/amazonmeta');
```

```
const items = [
  {
    title: 'Men Cotton Shirt',
    brand: 'Levi's',
    category: 'Clothing > Men > Shirts',
    description: 'Comfortable cotton shirt for men',
    material: 'Cotton',
    color: ['Blue', 'White'],
    size: ['M', 'L'],
    gender: 'Men',
    price: 999,
    discount: 10,
    stock_quantity: 50,
    rating: 4.2,
    image_urls: ['https://example.com/img1.jpg']
  },
  {
    title: 'Women Silk Kurti',
```

```
brand: 'Biba',
category: 'Clothing > Women > Kurtis',
description: 'Elegant silk kurti for women',
material: 'Silk',
color: ['Red'],
size: ['S', 'M', 'L'],
gender: 'Women',
price: 1299,
discount: 15,
stock_quantity: 30,
rating: 4.5,
image_urls: ['https://example.com/img2.jpg']
},
{
  title: 'Kids T-Shirt',
  brand: 'Babyhug',
  category: 'Clothing > Kids > T-Shirts',
  description: 'Colorful cotton t-shirt for kids',
  material: 'Cotton',
  color: ['Green', 'Yellow'],
  size: ['XS', 'S'],
  gender: 'Unisex',
  price: 499,
  discount: 5,
  stock_quantity: 100,
  rating: 4.0,
  image_urls: ['https://example.com/img3.jpg']
},
{
  title: 'Formal Men’s Shirt',
  brand: 'Arrow',
  category: 'Clothing > Men > Shirts',
  description: 'Perfect for office wear',
  material: 'Polyester Cotton',
  color: ['White'],
  size: ['M', 'L', 'XL'],
  gender: 'Men',
  price: 1199,
  discount: 20,
  stock_quantity: 40,
  rating: 4.3,
  image_urls: ['https://example.com/img4.jpg']
},
{
  title: 'Women’s Sports T-shirt',
  brand: 'Puma',
  category: 'Clothing > Women > Sportswear',
  description: 'Breathable sportswear for women',
  material: 'Polyester',
  color: ['Pink'],
  size: ['S', 'M'],
  gender: 'Women',
  price: 899,
  discount: 10,
  stock_quantity: 25,
  rating: 4.6,
  image_urls: ['https://example.com/img5.jpg']
}
```

```
},
{
  title: 'Men Track Pants',
  brand: 'Nike',
  category: 'Clothing > Men > Sportswear',
  description: 'Stretchable and comfy track pants',
  material: 'Nylon',
  color: ['Black'],
  size: ['L', 'XL'],
  gender: 'Men',
  price: 1499,
  discount: 12,
  stock_quantity: 20,
  rating: 4.4,
  image_urls: ['https://example.com/img6.jpg']
},
{
  title: 'Casual Shirt',
  brand: 'Roadster',
  category: 'Clothing > Men > Casual Wear',
  description: 'Trendy casual shirt',
  material: 'Cotton',
  color: ['Olive'],
  size: ['S', 'M'],
  gender: 'Men',
  price: 799,
  discount: 8,
  stock_quantity: 60,
  rating: 4.1,
  image_urls: ['https://example.com/img7.jpg']
},
{
  title: 'Printed Women Top',
  brand: 'ONLY',
  category: 'Clothing > Women > Tops',
  description: 'Stylish printed top',
  material: 'Rayon',
  color: ['White', 'Blue'],
  size: ['S', 'M', 'L'],
  gender: 'Women',
  price: 699,
  discount: 5,
  stock_quantity: 45,
  rating: 4.0,
  image_urls: ['https://example.com/img8.jpg']
},
{
  title: 'Boy's Jeans',
  brand: 'Pepe Jeans',
  category: 'Clothing > Kids > Jeans',
  description: 'Stretch jeans for boys',
  material: 'Denim',
  color: ['Blue'],
  size: ['XS', 'S', 'M'],
  gender: 'Boys',
  price: 1099,
  discount: 18,
```

```

    stock_quantity: 70,
    rating: 4.3,
    image_urls: ['https://example.com/img9.jpg']
  },
  {
    title: 'Women's Leggings',
    brand: 'Zivame',
    category: 'Clothing > Women > Leggings',
    description: 'Soft and stretchable leggings',
    material: 'Lycra',
    color: ['Black', 'Maroon'],
    size: ['M', 'L'],
    gender: 'Women',
    price: 599,
    discount: 7,
    stock_quantity: 90,
    rating: 4.2,
    image_urls: ['https://example.com/img10.jpg']
  }
];

```

```

Item.insertMany(items).then(() => {
  console.log('Dummy items inserted');
  mongoose.connection.close();
});

```

Run MongoDB locally

```
npm init -y
```

```
npm install express mongoose
```

Start the server:

```
node server.js
```

1. Find all men's clothing under ₹1000

```

db.items.find({
  gender: "Men",
  price: { $lt: 1000 }
})

```

✅ 2. Find all products with stock less than 30

```
db.items.find({
  stock_quantity: { $lt: 30 }
})
```

✅ 3. Get items with rating above 4.3 and sorted by rating (descending)

```
db.items.find({
  rating: { $gt: 4.3 }
}).sort({ rating: -1 })
```

✅ 4. Find all items available in the color "Blue"

```
db.items.find({
  color: "Blue"
})
```

MongoDB matches arrays, so "Blue" in ["White", "Blue"] works.

✅ 5. Count how many items belong to the brand "Puma"

```
db.items.countDocuments({
  brand: "Puma"
})
```