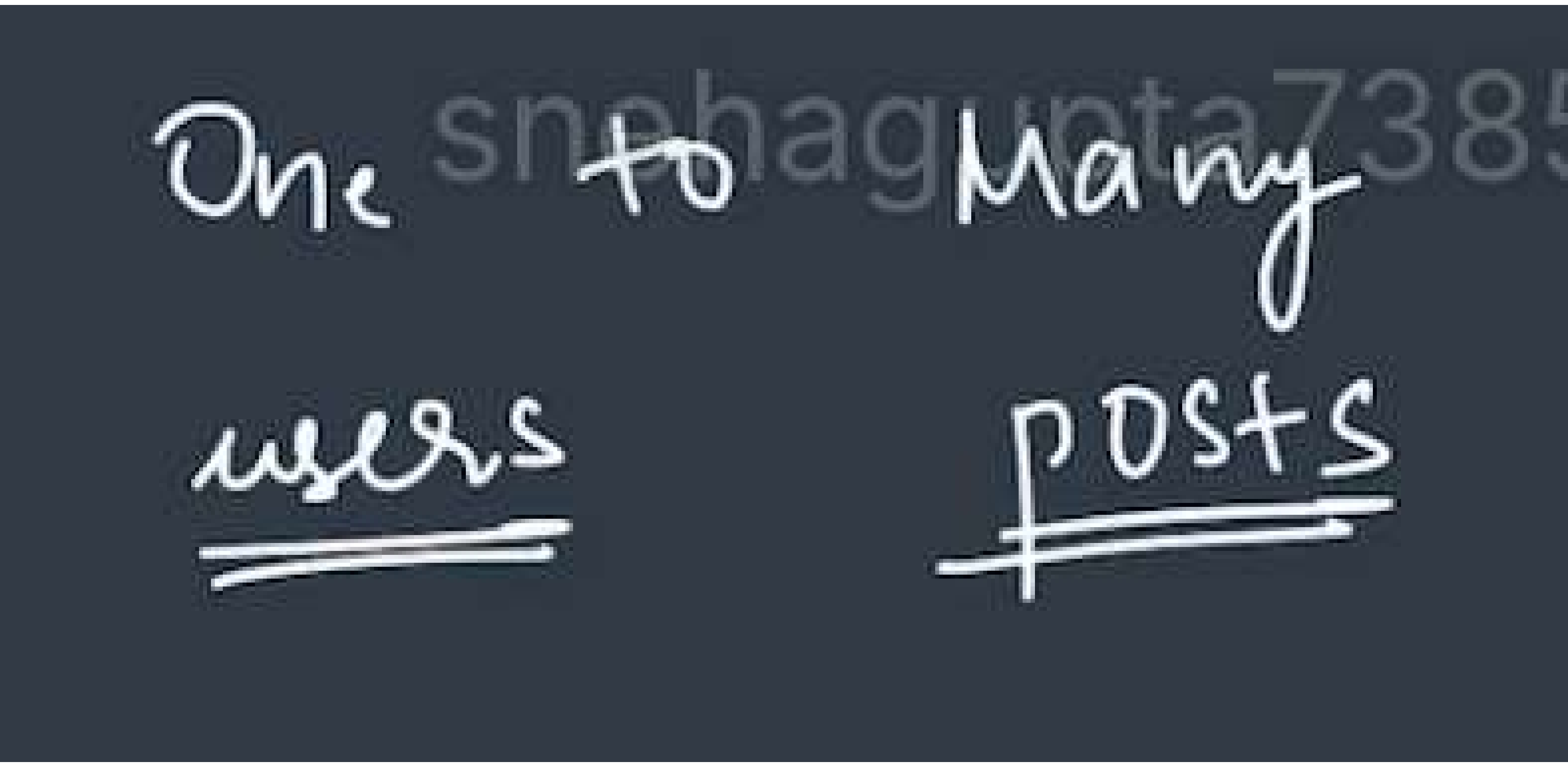
Handling Deletion

using **Mongoose Middleware**

We can use 2 middlewares :

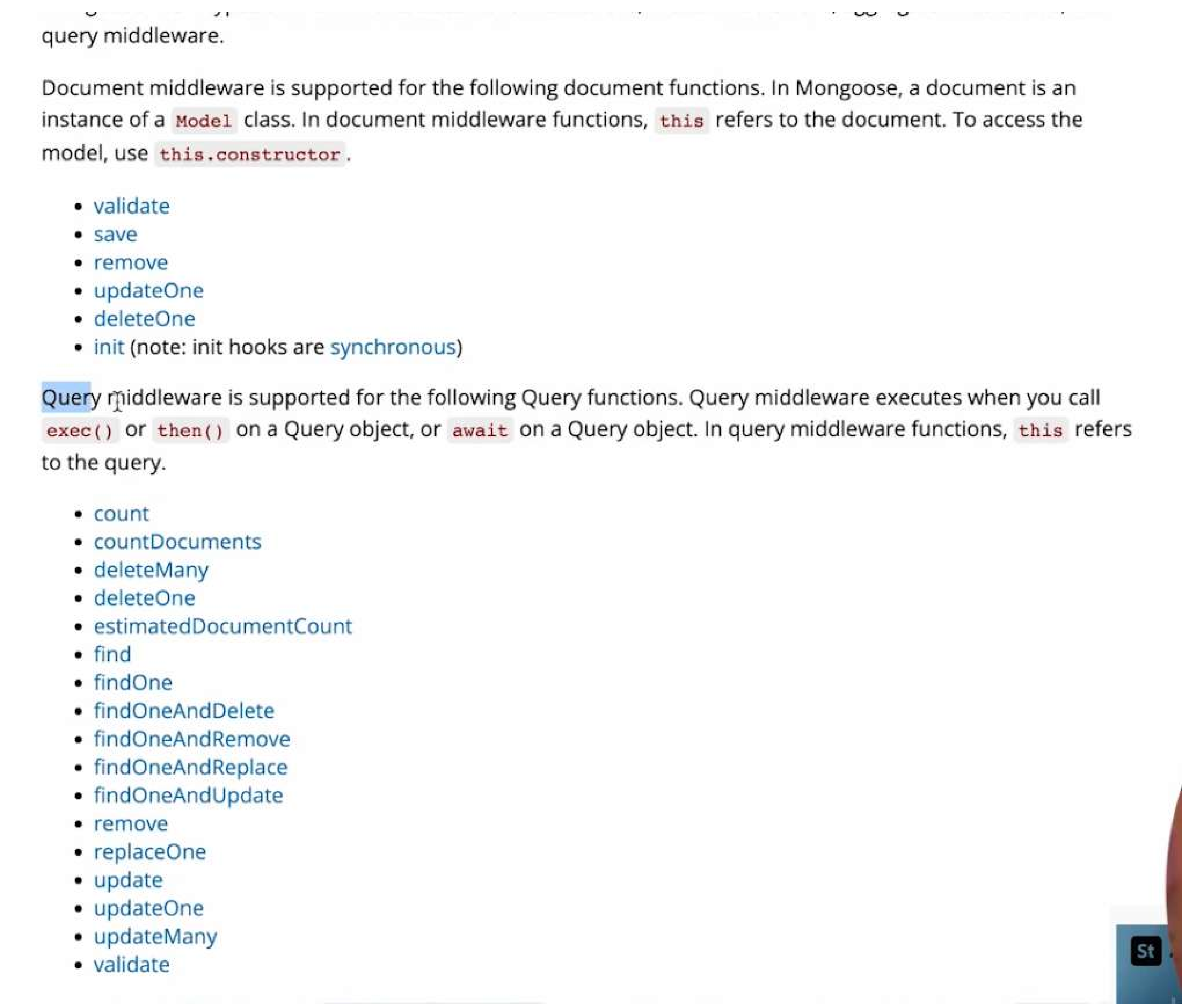
- **Pre - run before the query is executed**
- **Post - run after the query is executed**

```
customerSchema.post("findOneAndDelete", async (customer) => {
  if (customer.orders.length) {
    let res = await Order.deleteMany({ _id: { $in: customer.orders } });
    console.log(res);
  }
});
```



One to Many

users posts



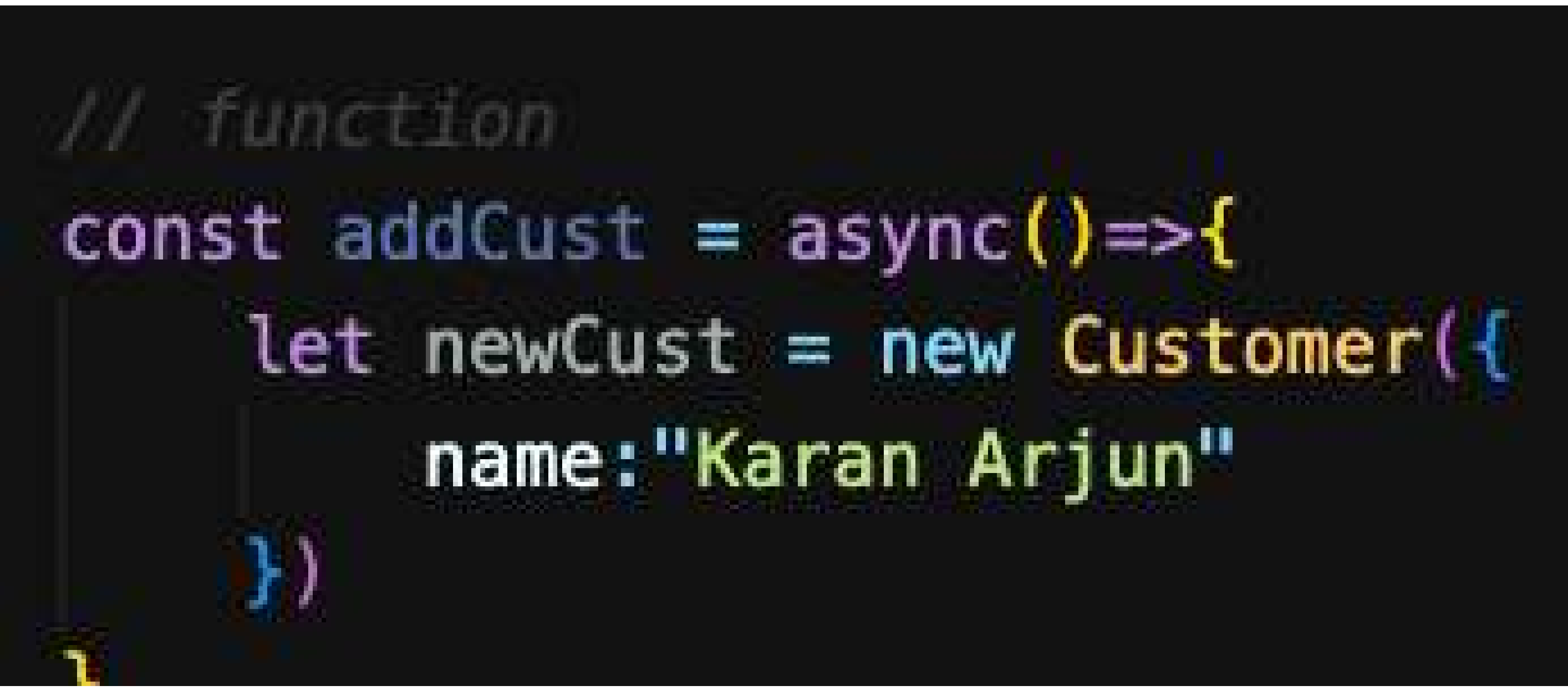
query middleware.

Document middleware is supported for the following document functions. In Mongoose, a document is an instance of a `Model` class. In document middleware functions, `this` refers to the document. To access the model, use `this.constructor`.

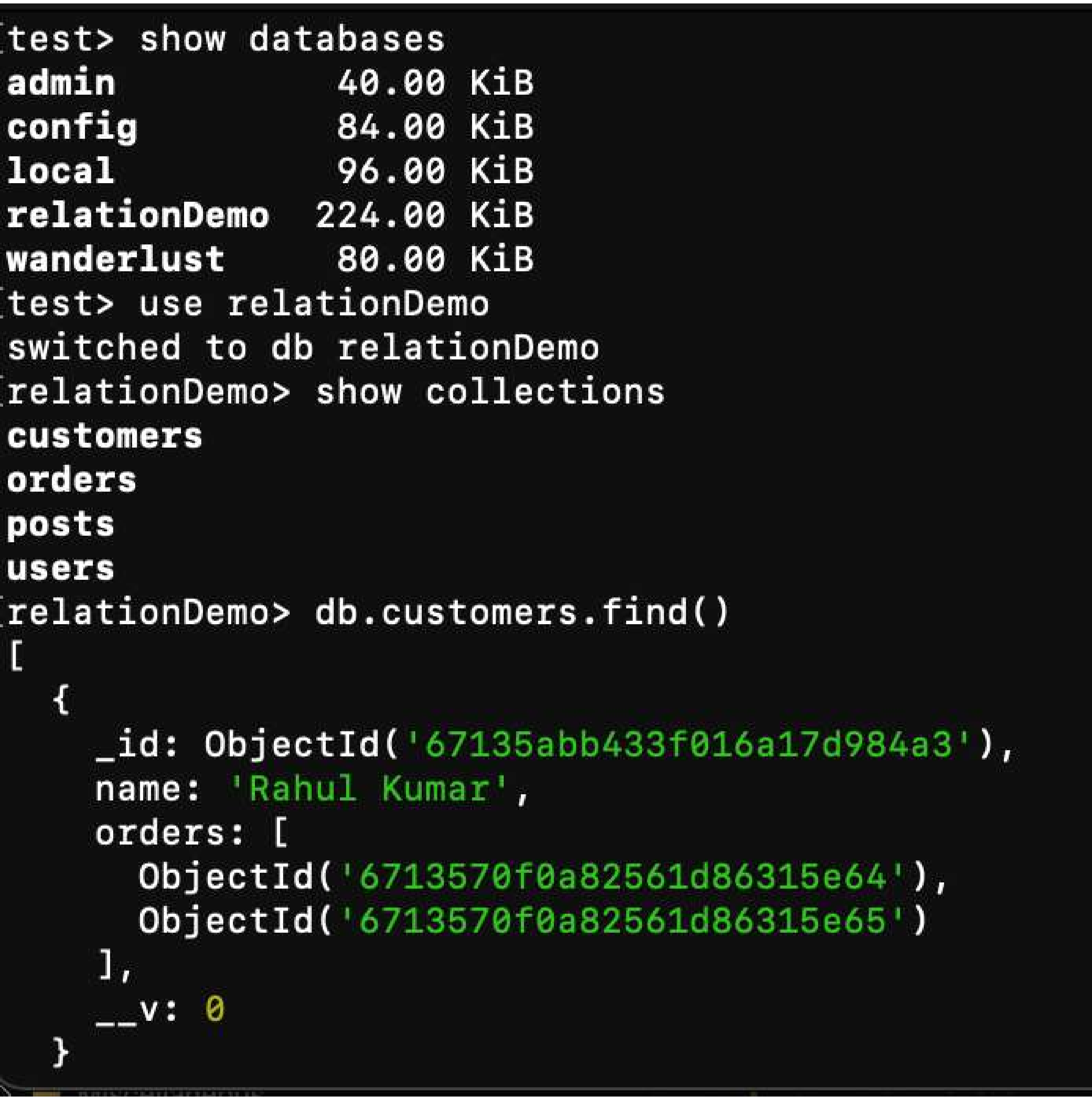
- validate
- save
- remove
- updateOne
- deleteOne
- init (note: init hooks are synchronous)

Query middleware is supported for the following Query functions. Query middleware executes when you call `exec()` or `then()` on a Query object, or `await` on a Query object. In query middleware functions, `this` refers to the query.

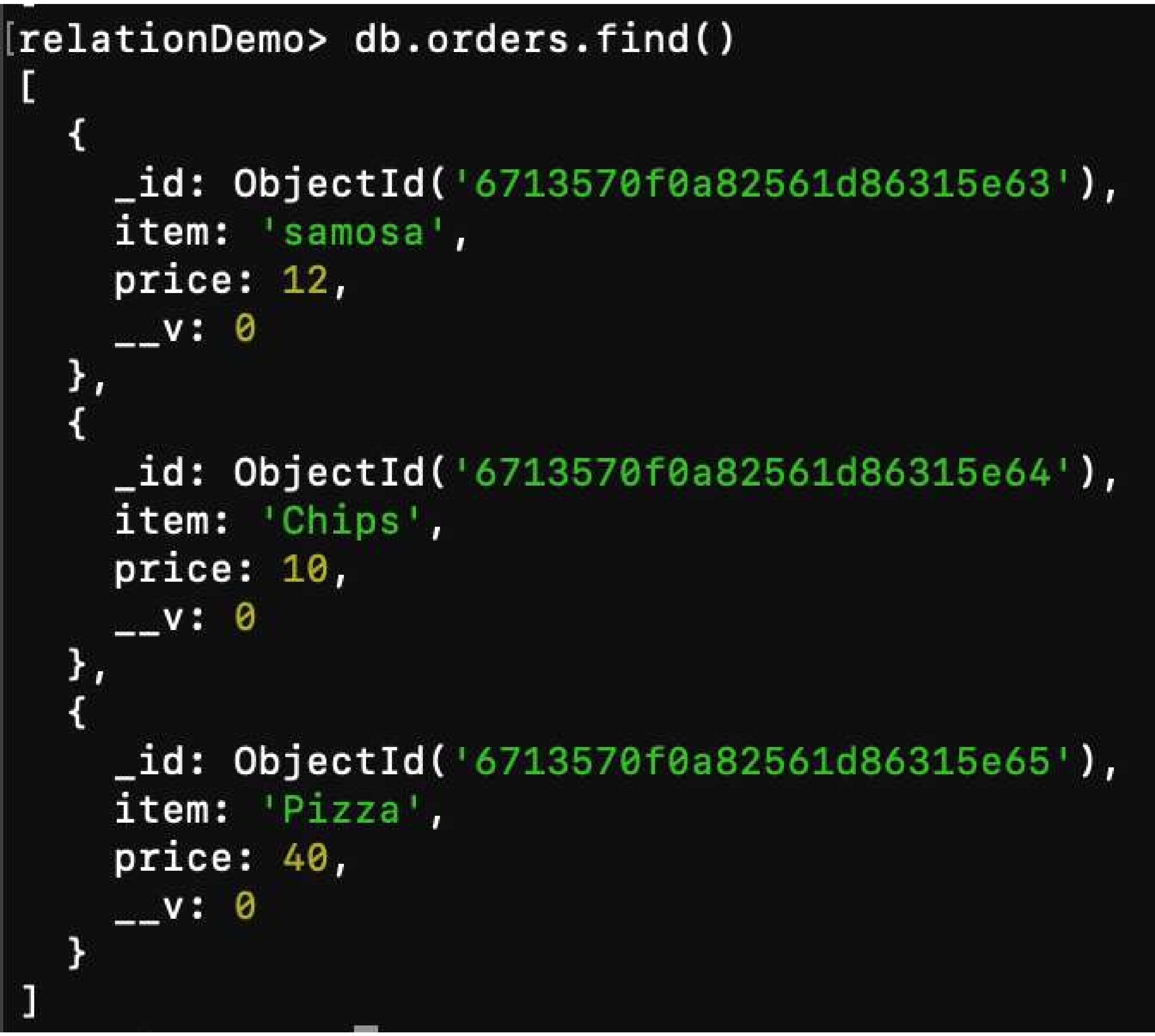
- count
- countDocuments
- deleteMany
- deleteOne
- estimatedDocumentCount
- find
- findOne
- findOneAndDelete
- findOneAndRemove
- findOneAndReplace
- findOneAndUpdate
- remove
- replaceOne
- update
- updateOne
- updateMany
- validate



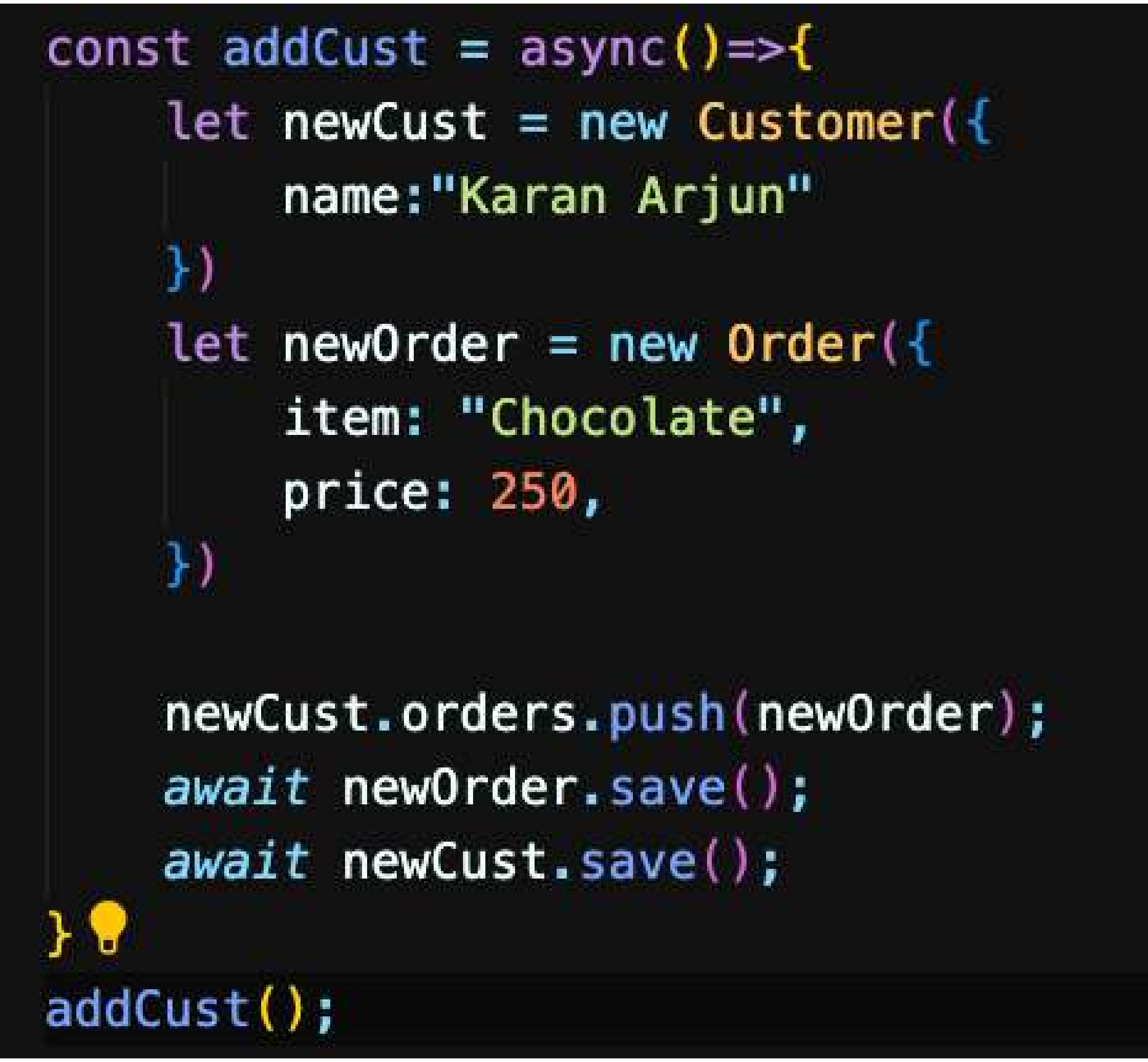
```
// function
const addCust = async()=>{
    let newCust = new Customer({
        name:"Karan Arjun"
    })
}
```



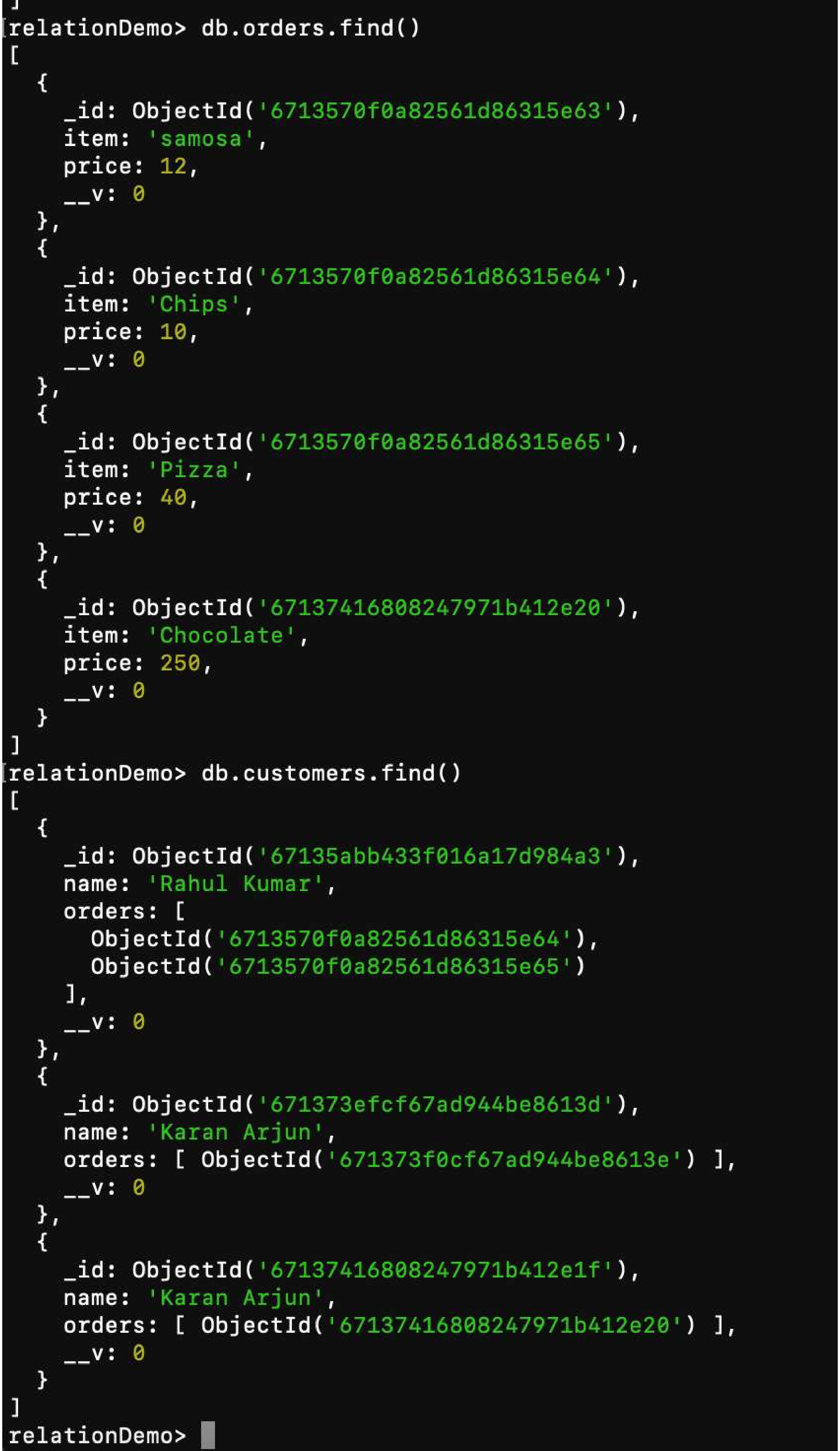
```
test> show databases
admin          40.00 KiB
config         84.00 KiB
local          96.00 KiB
relationDemo  224.00 KiB
wanderlust     80.00 KiB
test> use relationDemo
switched to db relationDemo
relationDemo> show collections
customers
orders
posts
users
relationDemo> db.customers.find()
[
  {
    _id: ObjectId('67135abb433f016a17d984a3'),
    name: 'Rahul Kumar',
    orders: [
      ObjectId('6713570f0a82561d86315e64'),
      ObjectId('6713570f0a82561d86315e65')
    ],
    __v: 0
  }
```



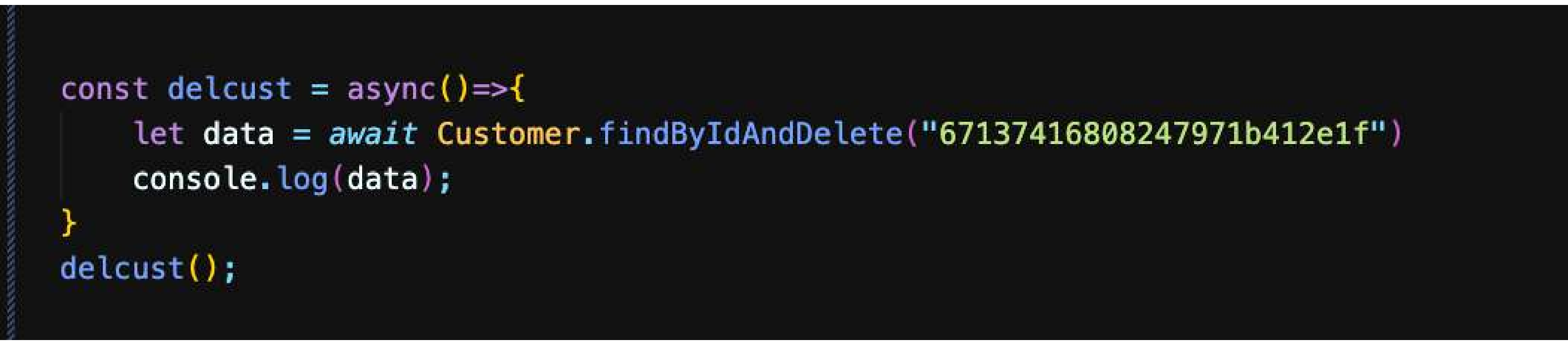
```
relationDemo> db.orders.find()
[
  {
    _id: ObjectId('6713570f0a82561d86315e63'),
    item: 'samosa',
    price: 12,
    __v: 0
  },
  {
    _id: ObjectId('6713570f0a82561d86315e64'),
    item: 'Chips',
    price: 10,
    __v: 0
  },
  {
    _id: ObjectId('6713570f0a82561d86315e65'),
    item: 'Pizza',
    price: 40,
    __v: 0
  }
]
```



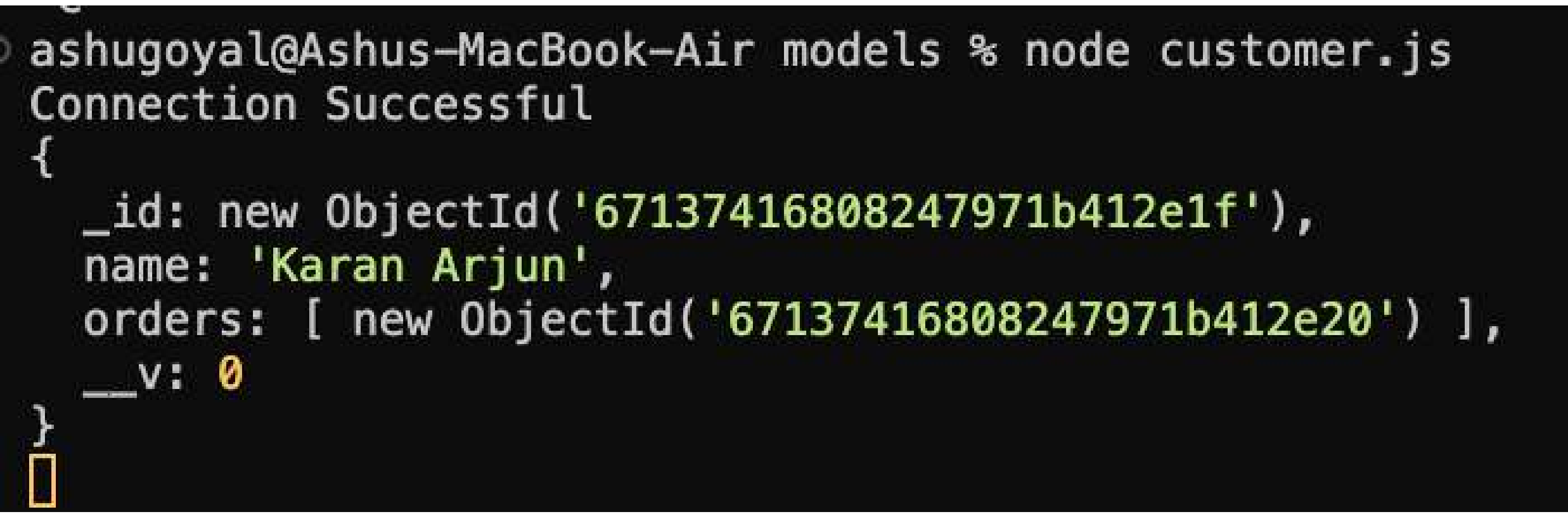
```
const addCust = async()=>{
    let newCust = new Customer({
        name:"Karan Arjun"
    })
    let newOrder = new Order({
        item: "Chocolate",
        price: 250,
    })

    newCust.orders.push(newOrder);
    await newOrder.save();
    await newCust.save();
}
addCust();
```



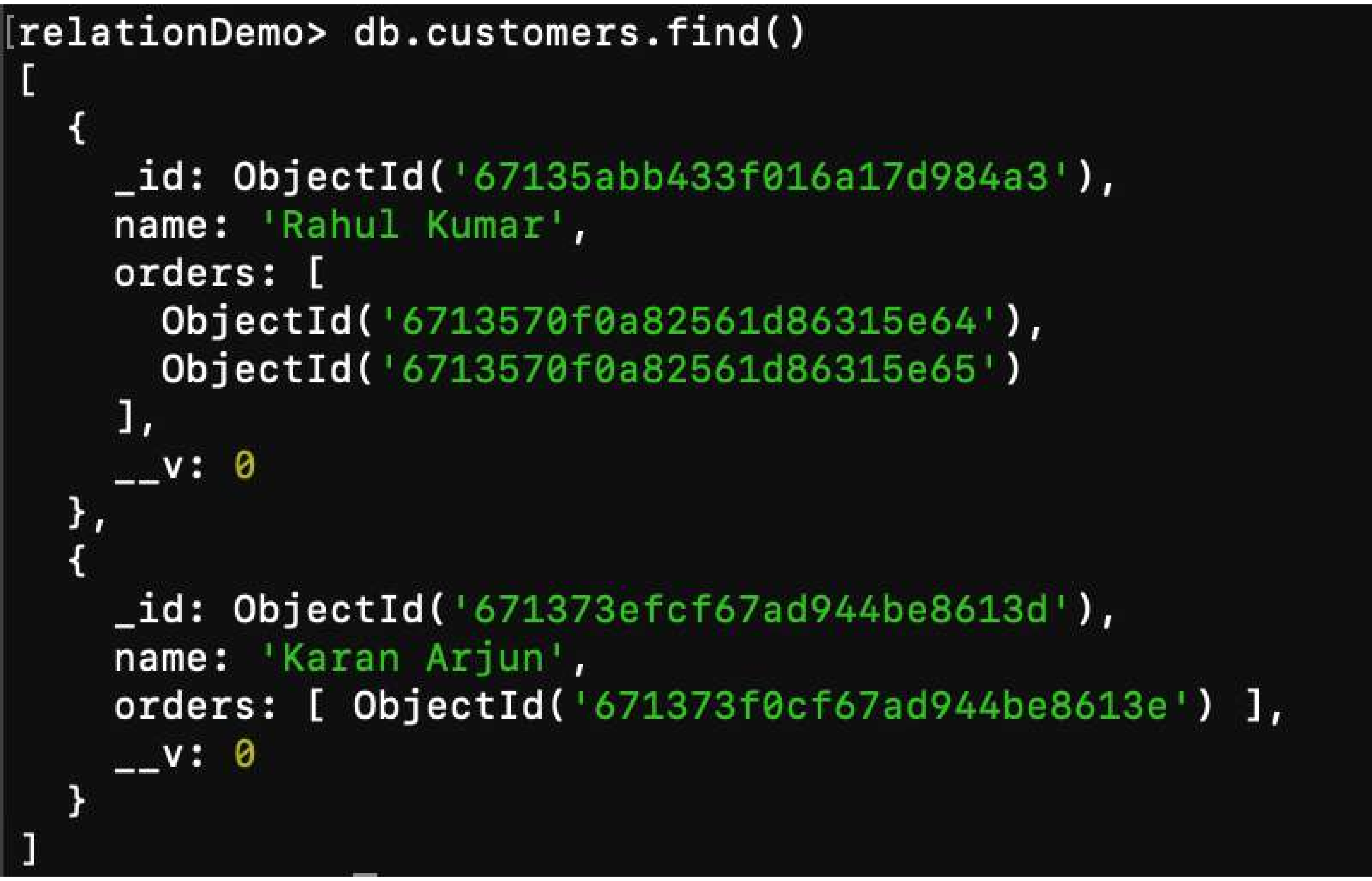
```
]
relationDemo> db.orders.find()
[
  {
    _id: ObjectId('6713570f0a82561d86315e63'),
    item: 'samosa',
    price: 12,
    __v: 0
  },
  {
    _id: ObjectId('6713570f0a82561d86315e64'),
    item: 'Chips',
    price: 10,
    __v: 0
  },
  {
    _id: ObjectId('6713570f0a82561d86315e65'),
    item: 'Pizza',
    price: 40,
    __v: 0
  },
  {
    _id: ObjectId('67137416808247971b412e20'),
    item: 'Chocolate',
    price: 250,
    __v: 0
  }
]
relationDemo> db.customers.find()
[
  {
    _id: ObjectId('67135abb433f016a17d984a3'),
    name: 'Rahul Kumar',
    orders: [
      ObjectId('6713570f0a82561d86315e64'),
      ObjectId('6713570f0a82561d86315e65')
    ],
    __v: 0
  },
  {
    _id: ObjectId('671373efcf67ad944be8613d'),
    name: 'Karan Arjun',
    orders: [ ObjectId('671373f0cf67ad944be8613e') ],
    __v: 0
  },
  {
    _id: ObjectId('67137416808247971b412e1f'),
    name: 'Karan Arjun',
    orders: [ ObjectId('67137416808247971b412e20') ],
    __v: 0
  }
]
relationDemo> 
```



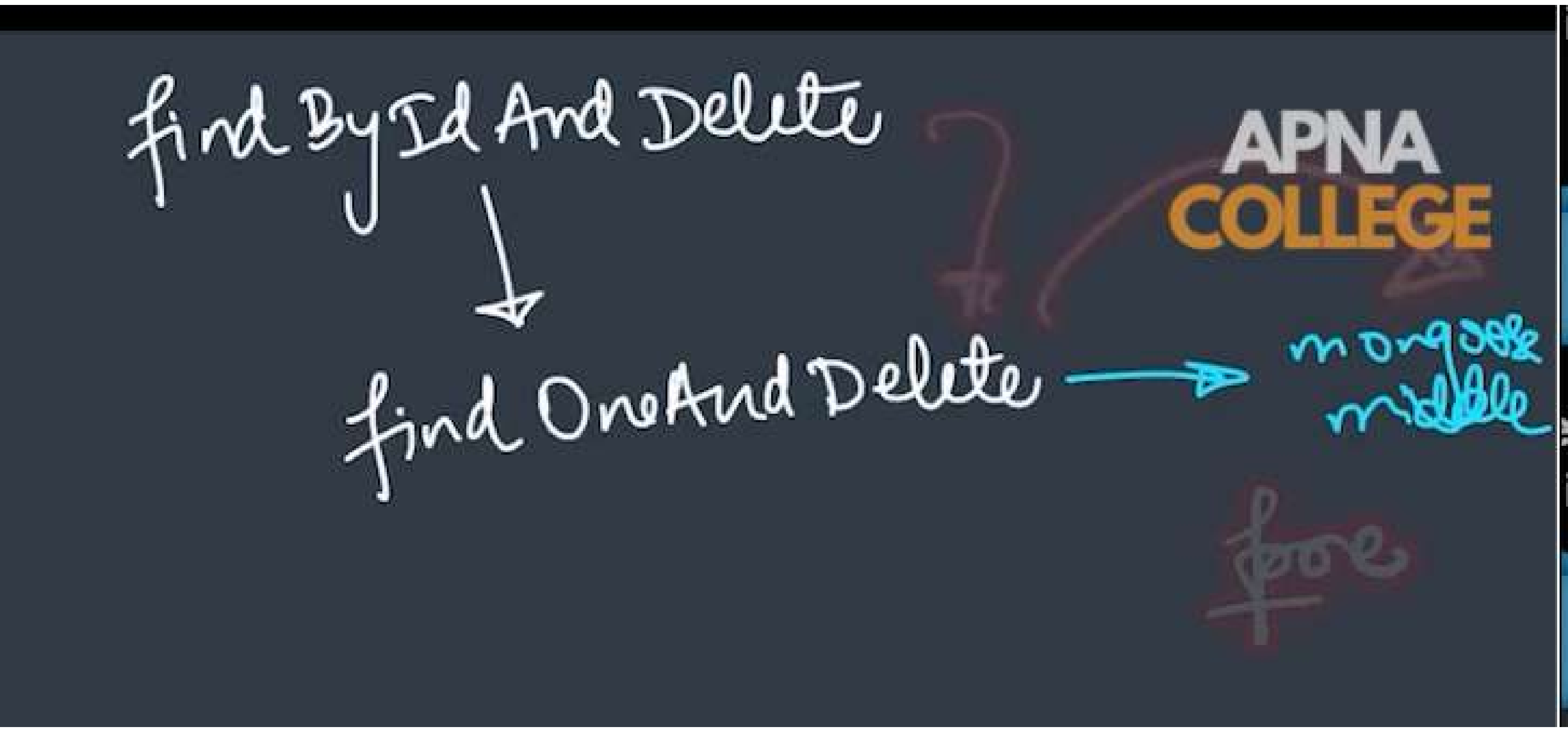
```
const delcust = async()=>{
    let data = await Customer.findByIdAndDelete("67137416808247971b412e1f")
    console.log(data);
}
delcust();
```



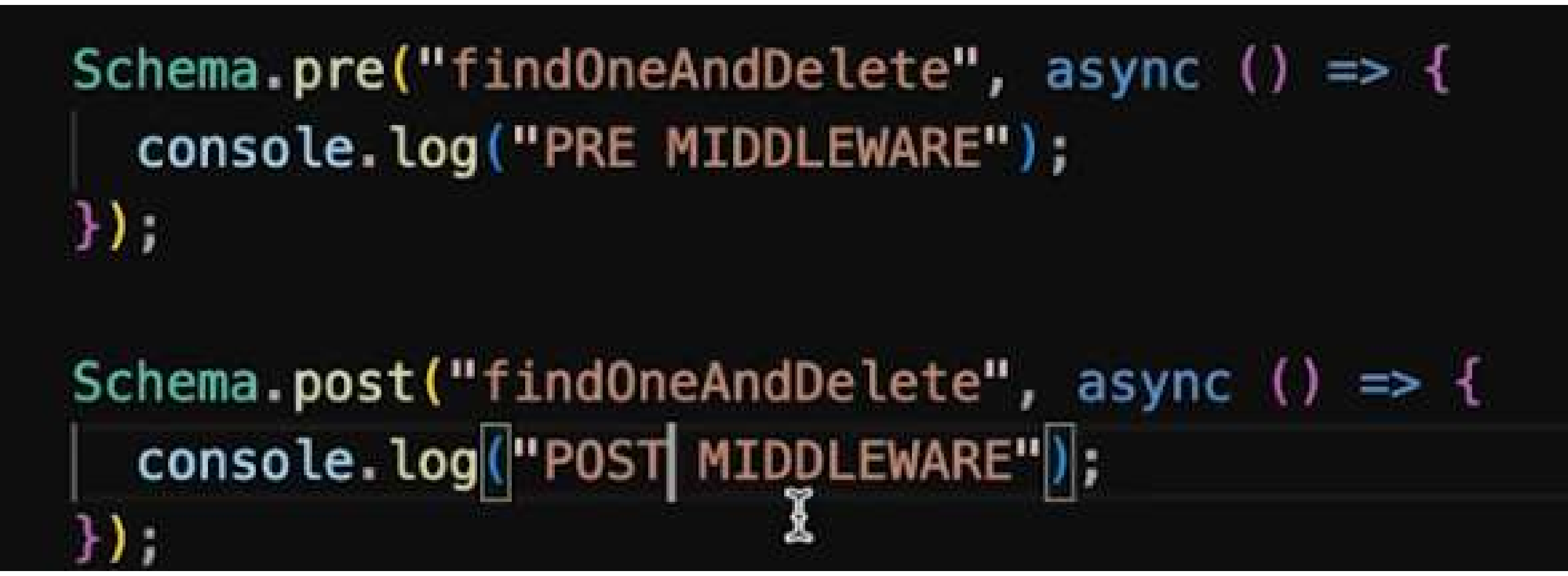
```
ashugoyal@Ashus-MacBook-Air models % node customer.js
Connection Successful
{
  _id: new ObjectId('671374168082479 71b412e1f'),
  name: 'Karan Arjun',
  orders: [ new ObjectId('67137416808247971b412e20') ],
  __v: 0
}
```



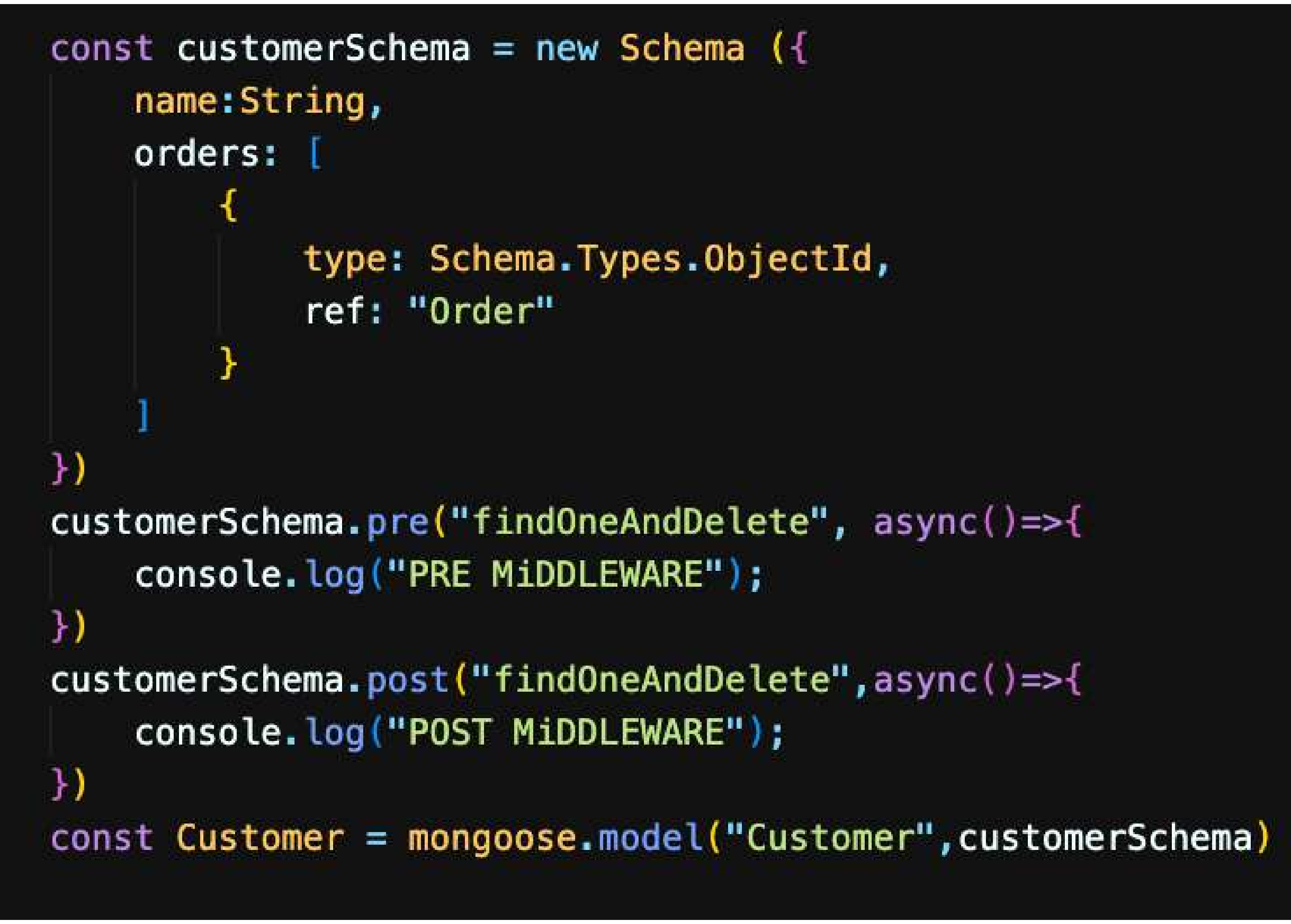
```
relationDemo> db.customers.find()
[
  {
    _id: ObjectId('67135abb433f016a17d984a3'),
    name: 'Rahul Kumar',
    orders: [
      ObjectId('6713570f0a82561d86315e64'),
      ObjectId('6713570f0a82561d86315e65')
    ],
    __v: 0
  },
  {
    _id: ObjectId('671373efcf67ad944be8613d'),
    name: 'Karan Arjun',
    orders: [ ObjectId('671373f0cf67ad944be8613e') ],
    __v: 0
  }
]
```



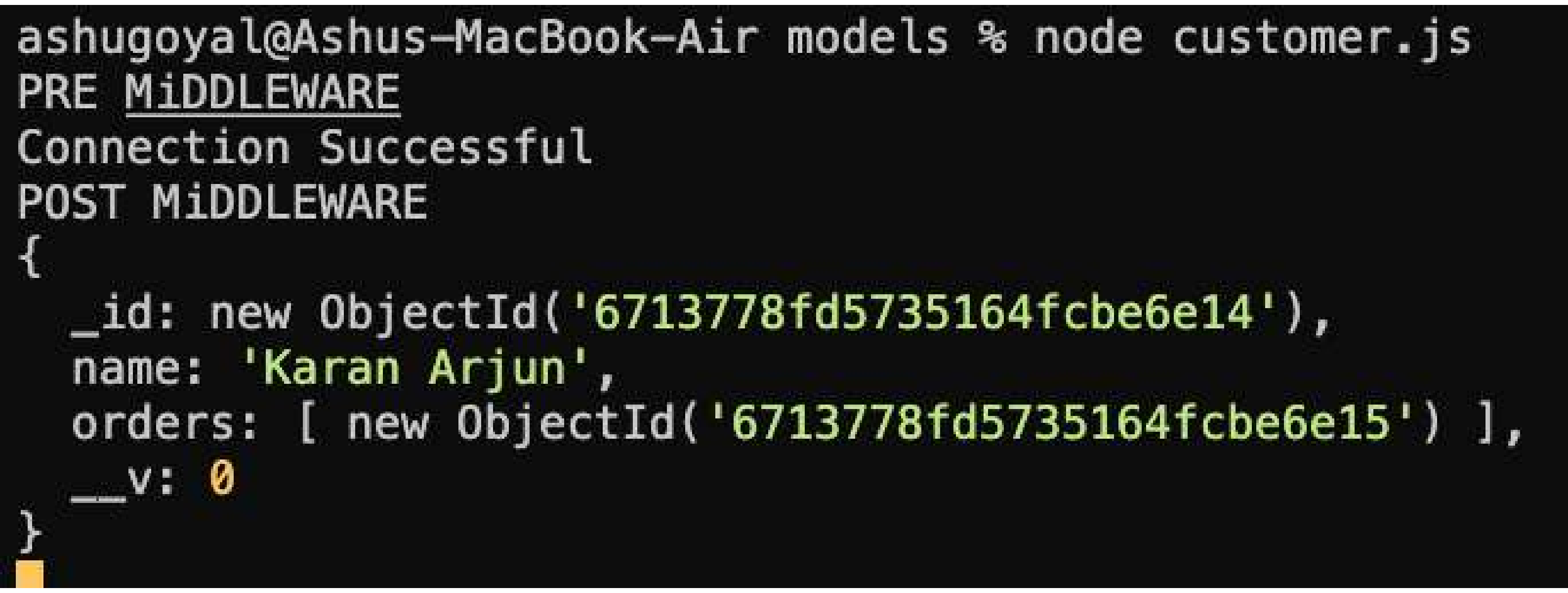
$$\begin{tikzcd}
\text{findByIdAndDelete} \arrow[d] & & \\
\text{findOneAndDelete} \arrow[rr] & & \text{mongoose middleware}
\end{tikzcd}$$



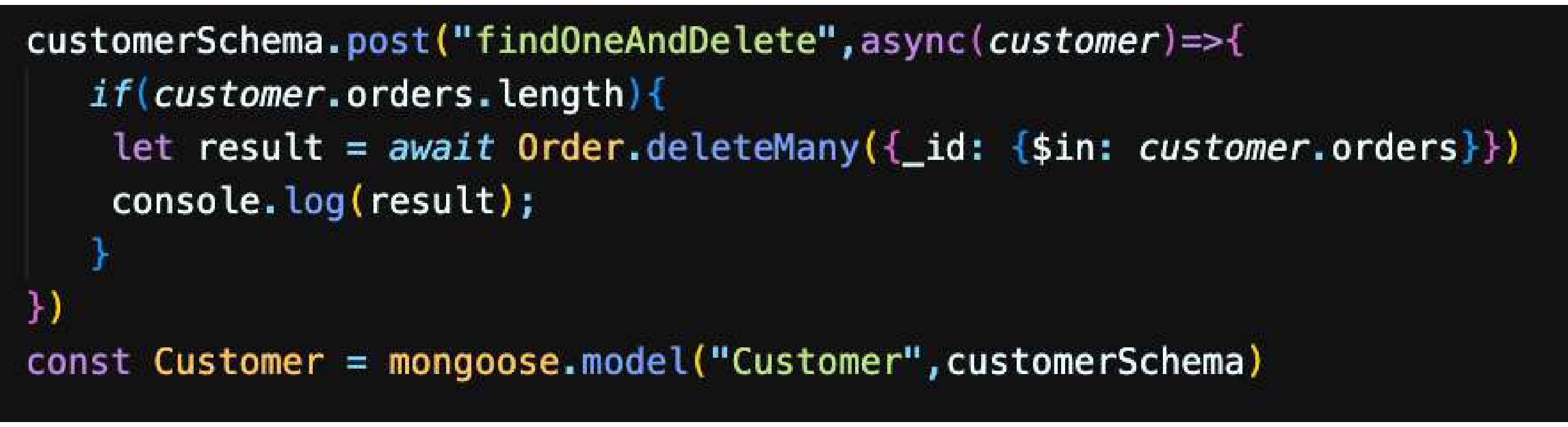
```
Schema.pre("findOneAndDelete", async () => {
  console.log("PRE MIDDLEWARE");
});

Schema.post("findOneAndDelete", async () => {
  console.log("POST MIDDLEWARE");
});
```



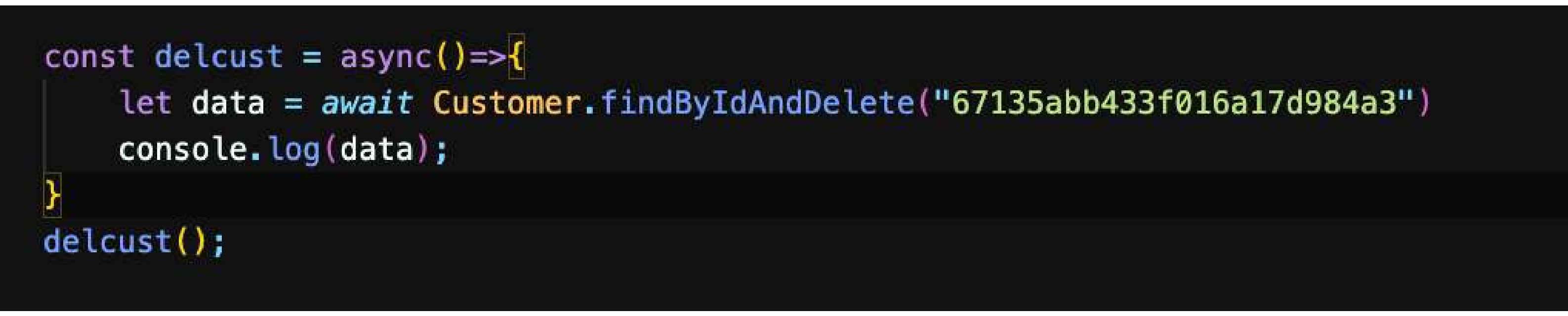
```
const customerSchema = new Schema ({
    name:String,
    orders: [
        {
            type: Schema.Types.ObjectId,
            ref: "Order"
        }
    ]
})
customerSchema.pre("findOneAndDelete", async()=>{
    console.log("PRE MiDDLEWARE");
})
customerSchema.post("findOneAndDelete",async()=>{
    console.log("POST MiDDLEWARE");
})
const Customer = mongoose.model("Customer",customerSchema)
```



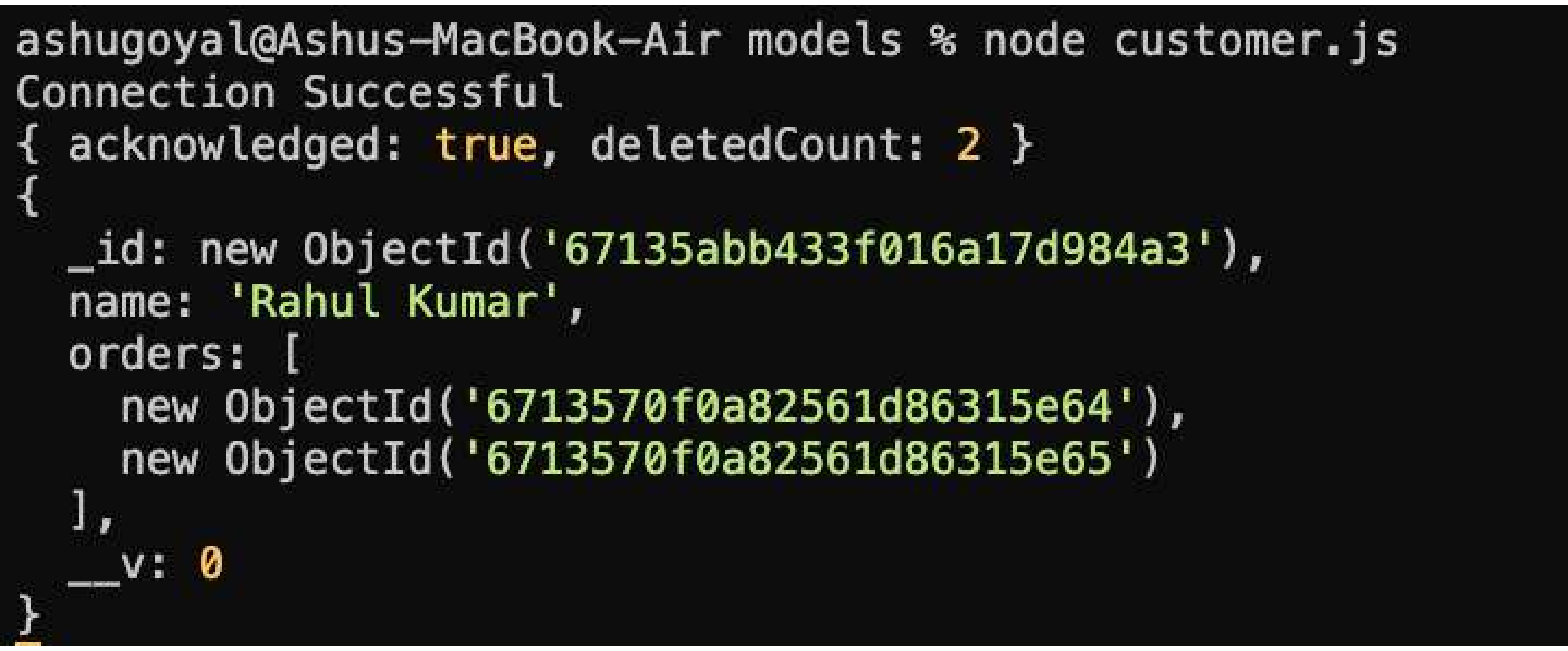
```
ashugoyal@Ashus-MacBook-Air models % node customer.js
PRE MiDDLEWARE
Connection Successful
POST MiDDLEWARE
{
  _id: new ObjectId('6713778fd5735164fcbe6e14'),
  name: 'Karan Arjun',
  orders: [ new ObjectId('6713778fd5735164fcbe6e15') ],
  __v: 0
}
```



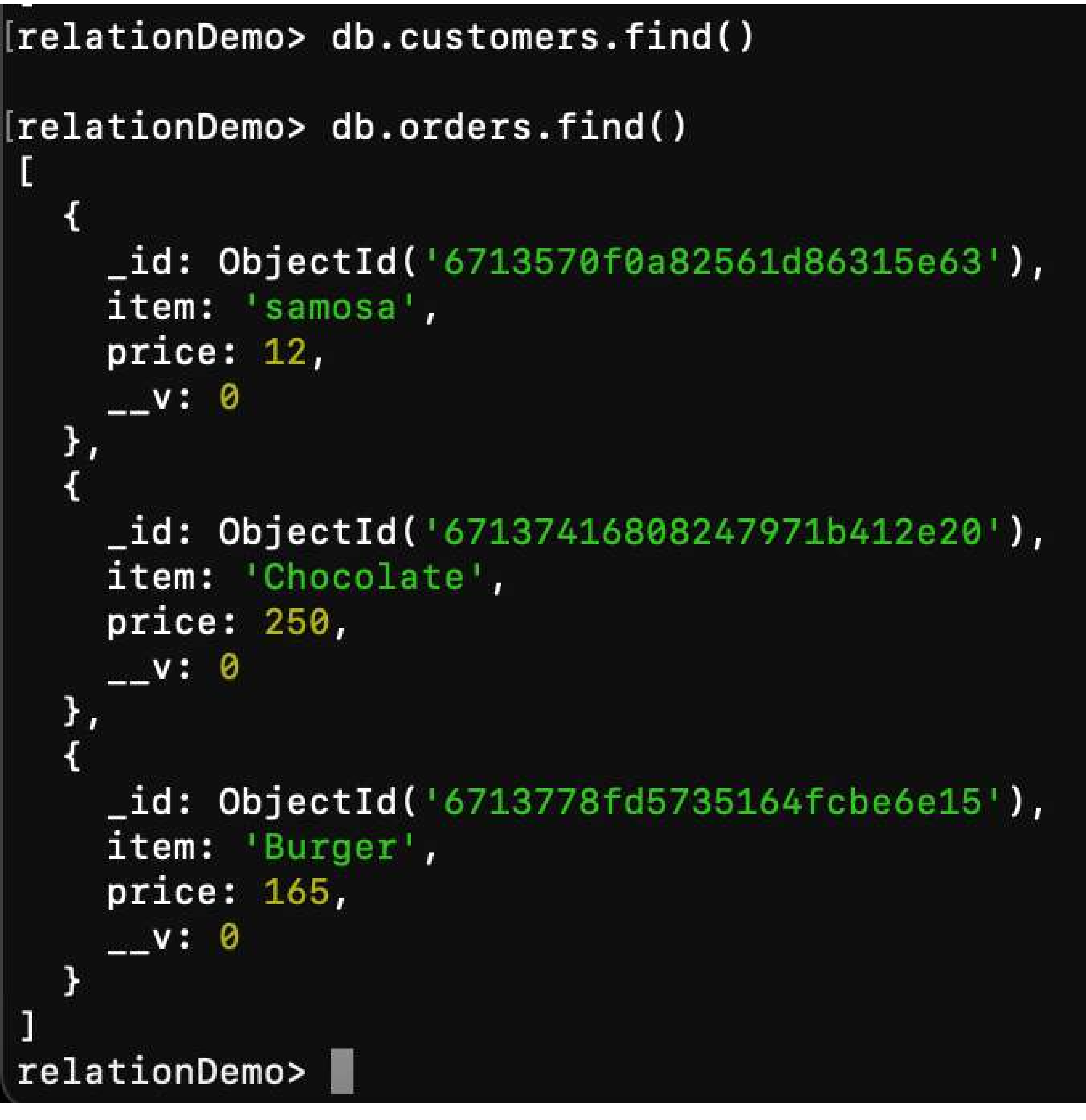
```
customerSchema.post("findOneAndDelete",async(customer)=>{
    if(customer.orders.length){
     let result = await Order.deleteMany({_id: {$in: customer.orders}})
     console.log(result);
    }
})
const Customer = mongoose.model("Customer",customerSchema)
```



```
const delcust = async()=>{
    let data = await Customer.findByIdAndDelete("67135abb433f016a17d984a3")
    console.log(data);
}
delcust();
```



```
ashugoyal@Ashus-MacBook-Air models % node customer.js
Connection Successful
{ acknowledged: true, deletedCount: 2 }
{
  _id: new ObjectId('67135abb433f016a17d984a3'),
  name: 'Rahul Kumar',
  orders: [
    new ObjectId('6713570f0a82561d86315e64'),
    new ObjectId('6713570f0a82561d86315e65')
  ],
  __v: 0
}
```



```
relationDemo> db.customers.find()

relationDemo> db.orders.find()
[
  {
    _id: ObjectId('6713570f0a82561d86315e63'),
    item: 'samosa',
    price: 12,
    __v: 0
  },
  {
    _id: ObjectId('671374168082479 71b412e20'),
    item: 'Chocolate',
    price: 250,
    __v: 0
  },
  {
    _id: ObjectId('6713778fd5735164fcbe6e15'),
    item: 'Burger',
    price: 165,
    __v: 0
  }
]
relationDemo> 
```