```
ironic@iRONiC:~/Desktop/Frontend-Web/MongooseBasics$ node
Welcome to Node.js v18.13.0.
Type ".help" for more information.
> .editor
// Entering editor mode (Ctrl+D to finish, Ctrl+C to cancel)

const mongoose = require("mongoose");
mongoose.connect('mongodb://127.0.0.1:27017/test')


.then(()=> console.log("open"))
.catch((er) => console.log("error ", er))



const moviSchema = new mongoose.Schema({
        title: String,
        year: Number,
        rating: Number,
        isGood: Boolean,
});

const Movie = mongoose.model("Movie", moviSchema);

// const dune = new Movie({ title: "dune", year: 2022, rating: 9, isGood: true})


Movie.insertMany([
        { title: "dune", year: 2022, rating: 9, isGood: true},
        { title: "lakhya", year: 2007, rating: 9, isGood: true},
        { title: "race3", year: 2018, rating: 3, isGood: false},
        { title: "dark", year: 2018, rating: 9.3, isGood: true},
        { title: "ozark", year: 2015, rating: 8, isGood: true}
        ])






.then(data => console.log(data))
Promise {
  <pending>,
  [Symbol(async_id_symbol)]: 167,
  [Symbol(trigger_async_id_symbol)]: 164
}
> open
[
  {
    title: 'dune',
    year: 2022,
    rating: 9,
    isGood: true,
    _id: new ObjectId("644ddf84abb3044f0e3f9cab"),
    __v: 0
  },
  {
    title: 'lakhya',
    year: 2007,
    rating: 9,
    isGood: true,
    _id: new ObjectId("644ddf84abb3044f0e3f9cac"),
    __v: 0
  },
  {
    title: 'race3',
    year: 2018,
    rating: 3,
    isGood: false,
    _id: new ObjectId("644ddf84abb3044f0e3f9cad"),
    __v: 0
  },
  {
    title: 'dark',
    year: 2018,
    rating: 9.3,
    isGood: true,
    _id: new ObjectId("644ddf84abb3044f0e3f9cae"),
    __v: 0
  },
  {
    title: 'ozark',
    year: 2015,
    rating: 8,
    isGood: true,
    _id: new ObjectId("644ddf84abb3044f0e3f9caf"),
    __v: 0
  }
]
> Movie.find()
Query {
  _mongooseOptions: {},
  _transforms: [],
  _hooks: Kareem { _pres: Map(0) {}, _posts: Map(0) {} },
  _executionStack: null,
  mongooseCollection: Collection {
    collection: Collection { s: [Object] },
    Promise: [Function: Promise],
    modelName: 'Movie',
    _closed: false,
    opts: {
      autoIndex: true,
      autoCreate: true,
      schemaUserProvidedOptions: {},
      capped: false,
      Promise: undefined,
      '$wasForceClosed': undefined
    },
    name: 'movies',
    collectionName: 'movies',
    conn: NativeConnection {
      base: [Mongoose],
      collections: [Object],
      models: [Object],
      config: {},
      replica: false,
      options: null,
      otherDbs: [],
      relatedDbs: {},
      states: [Object: null prototype],
      _readyState: 1,
      _closeCalled: undefined,
      _hasOpened: true,
      plugins: [],
      id: 0,
      _queue: [],
      _listening: false,
      _connectionOptions: [Object],
      _connectionString: 'mongodb://127.0.0.1:27017/test',
      client: [MongoClient],
      '$initialConnection': [Promise],
      db: [Db],
      host: '127.0.0.1',
      port: 27017,
      name: 'test'
    },
    queue: [],
    buffer: false,
    emitter: EventEmitter {
      _events: [Object: null prototype] {},
      _eventsCount: 0,
      _maxListeners: undefined,
      [Symbol(kCapture)]: false
    }
  },
  model: Model { Movie },
  schema: Schema {
    obj: {
      title: [Function: String],
      year: [Function: Number],
      rating: [Function: Number],
      isGood: [Function: Boolean]
    },
    paths: {
      title: [SchemaString],
      year: [SchemaNumber],
      rating: [SchemaNumber],
      isGood: [SchemaBoolean],
      _id: [ObjectId],
      __v: [SchemaNumber]
    },
    aliases: {},
    subpaths: {},
    virtuals: { id: [VirtualType] },
    singleNestedPaths: {},
    nested: {},
    inherits: {},
    callQueue: [],
    _indexes: [],
    methods: {},
    methodOptions: {},
    statics: {},
    tree: {
      title: [Function: String],
      year: [Function: Number],
      rating: [Function: Number],
      isGood: [Function: Boolean],
      _id: [Object],
      __v: [Function: Number],
      id: [VirtualType]
    },
    query: {},
    childSchemas: [],
    plugins: [ [Object], [Object], [Object], [Object], [Object] ],
    '$id': 1,
    mapPaths: [],
    s: { hooks: [Kareem] },
    _userProvidedOptions: {},
    options: {
      typeKey: 'type',
      id: true,
      _id: true,
      validateBeforeSave: true,
      read: null,
      shardKey: null,
      discriminatorKey: '__t',
      autoIndex: null,
      minimize: true,
      optimisticConcurrency: false,
      versionKey: '__v',
      capped: false,
      bufferCommands: true,
      strictQuery: false,
      strict: true,
      pluralization: true
    },
    '$globalPluginsApplied': true,
    _requiredpaths: []
  },
  op: 'find',
  options: {},
  _conditions: {},
  _fields: undefined,
  _updateDoc: undefined,
  _path: undefined,
  _distinctDoc: undefined,
  _collection: NodeCollection {
    collection: Collection {
      collection: [Collection],
      Promise: [Function: Promise],
      modelName: 'Movie',
      _closed: false,
      opts: [Object],
      name: 'movies',
      collectionName: 'movies',
      conn: [NativeConnection],
      queue: [],
      buffer: false,
      emitter: [EventEmitter]
    },
    collectionName: 'movies'
  },
  _traceFunction: undefined,
  '$useProjection': true
}
> Movie.find({})
Query {
  _mongooseOptions: {},
  _transforms: [],
  _hooks: Kareem { _pres: Map(0) {}, _posts: Map(0) {} },
  _executionStack: null,
  mongooseCollection: Collection {
    collection: Collection { s: [Object] },
    Promise: [Function: Promise],
    modelName: 'Movie',
    _closed: false,
    opts: {
      autoIndex: true,
      autoCreate: true,
      schemaUserProvidedOptions: {},
      capped: false,
      Promise: undefined,
      '$wasForceClosed': undefined
    },
    name: 'movies',
    collectionName: 'movies',
    conn: NativeConnection {
      base: [Mongoose],
      collections: [Object],
      models: [Object],
      config: {},
      replica: false,
      options: null,
      otherDbs: [],
      relatedDbs: {},
      states: [Object: null prototype],
      _readyState: 1,
      _closeCalled: undefined,
      _hasOpened: true,
      plugins: [],
      id: 0,
      _queue: [],
      _listening: false,
      _connectionOptions: [Object],
      _connectionString: 'mongodb://127.0.0.1:27017/test',
      client: [MongoClient],
      '$initialConnection': [Promise],
      db: [Db],
      host: '127.0.0.1',
      port: 27017,
      name: 'test'
    },
    queue: [],
    buffer: false,
    emitter: EventEmitter {
      _events: [Object: null prototype] {},
      _eventsCount: 0,
      _maxListeners: undefined,
      [Symbol(kCapture)]: false
    }
  },
  model: Model { Movie },
  schema: Schema {
    obj: {
      title: [Function: String],
      year: [Function: Number],
      rating: [Function: Number],
      isGood: [Function: Boolean]
    },
    paths: {
      title: [SchemaString],
      year: [SchemaNumber],
      rating: [SchemaNumber],
      isGood: [SchemaBoolean],
      _id: [ObjectId],
      __v: [SchemaNumber]
    },
    aliases: {},
    subpaths: {},
    virtuals: { id: [VirtualType] },
    singleNestedPaths: {},
    nested: {},
    inherits: {},
    callQueue: [],
    _indexes: [],
    methods: {},
    methodOptions: {},
    statics: {},
    tree: {
      title: [Function: String],
      year: [Function: Number],
      rating: [Function: Number],
      isGood: [Function: Boolean],
      _id: [Object],
      __v: [Function: Number],
      id: [VirtualType]
    },
    query: {},
    childSchemas: [],
    plugins: [ [Object], [Object], [Object], [Object], [Object] ],
    '$id': 1,
    mapPaths: [],
    s: { hooks: [Kareem] },
    _userProvidedOptions: {},
    options: {
      typeKey: 'type',
      id: true,
      _id: true,
      validateBeforeSave: true,
      read: null,
      shardKey: null,
      discriminatorKey: '__t',
      autoIndex: null,
      minimize: true,
      optimisticConcurrency: false,
      versionKey: '__v',
      capped: false,
      bufferCommands: true,
      strictQuery: false,
      strict: true,
      pluralization: true
    },
    '$globalPluginsApplied': true,
    _requiredpaths: []
  },
  op: 'find',
  options: {},
  _conditions: {},
  _fields: undefined,
  _updateDoc: undefined,
  _path: undefined,
  _distinctDoc: undefined,
  _collection: NodeCollection {
    collection: Collection {
      collection: [Collection],
      Promise: [Function: Promise],
      modelName: 'Movie',
      _closed: false,
      opts: [Object],
      name: 'movies',
      collectionName: 'movies',
      conn: [NativeConnection],
      queue: [],
      buffer: false,
      emitter: [EventEmitter]
    },
    collectionName: 'movies'
  },
  _traceFunction: undefined,
  '$useProjection': true
}
> Movie.find({}).then(data => console.log(data))
Promise {
  <pending>,
  [Symbol(async_id_symbol)]: 710,
  [Symbol(trigger_async_id_symbol)]: 706
}
> [
  {
    _id: new ObjectId("644ddf84abb3044f0e3f9cab"),
    title: 'dune',
    year: 2022,
    rating: 9,
    isGood: true,
    __v: 0
  },
  {
    _id: new ObjectId("644ddf84abb3044f0e3f9cac"),
    title: 'lakhya',
    year: 2007,
    rating: 9,
    isGood: true,
    __v: 0
  },
  {
    _id: new ObjectId("644ddf84abb3044f0e3f9cad"),
    title: 'race3',
    year: 2018,
    rating: 3,
    isGood: false,
    __v: 0
  },
  {
    _id: new ObjectId("644ddf84abb3044f0e3f9cae"),
    title: 'dark',
    year: 2018,
    rating: 9.3,
    isGood: true,
    __v: 0
  },
  {
    _id: new ObjectId("644ddf84abb3044f0e3f9caf"),
    title: 'ozark',
    year: 2015,
    rating: 8,
    isGood: true,
    __v: 0
  }
]
> Movie.find({rating:8}).then(data => console.log(data))
Promise {
  <pending>,
  [Symbol(async_id_symbol)]: 1004,
  [Symbol(trigger_async_id_symbol)]: 1000
}
> [
  {
    _id: new ObjectId("644ddf84abb3044f0e3f9caf"),
    title: 'ozark',
    year: 2015,
    rating: 8,
    isGood: true,
    __v: 0
  }
]
> Movie.find({year:{$gt:2018}}).then(data => console.log(data))
Promise {
  <pending>,
  [Symbol(async_id_symbol)]: 1555,
  [Symbol(trigger_async_id_symbol)]: 1551
}
> [
  {
    _id: new ObjectId("644ddf84abb3044f0e3f9cab"),
    title: 'dune',
    year: 2022,
    rating: 9,
    isGood: true,
    __v: 0
  }
]
> Movie.find({year:{$lt:2018}}).then(data => console.log(data))
Promise {
  <pending>,
  [Symbol(async_id_symbol)]: 1788,
  [Symbol(trigger_async_id_symbol)]: 1784
}
> [
  {
    _id: new ObjectId("644ddf84abb3044f0e3f9cac"),
    title: 'lakhya',
    year: 2007,
    rating: 9,
    isGood: true,
    __v: 0
  },
  {
    _id: new ObjectId("644ddf84abb3044f0e3f9caf"),
    title: 'ozark',
    year: 2015,
    rating: 8,
    isGood: true,
    __v: 0
  }
]
> Movie.findOne({}).then(data => console.log(data))
Promise {
  <pending>,
  [Symbol(async_id_symbol)]: 2216,
  [Symbol(trigger_async_id_symbol)]: 2212
}
> {
  _id: new ObjectId("644ddf84abb3044f0e3f9cab"),
  title: 'dune',
  year: 2022,
  rating: 9,
  isGood: true,
  __v: 0
}
> Movie.findOne({year:2018}).then(data => console.log(data))
Promise {
  <pending>,
  [Symbol(async_id_symbol)]: 2423,
  [Symbol(trigger_async_id_symbol)]: 2419
}
> {
  _id: new ObjectId("644ddf84abb3044f0e3f9cad"),
  title: 'race3',
  year: 2018,
  rating: 3,
  isGood: false,
  __v: 0
}
> Movie.findById({_id:"644ddf84abb3044f0e3f9cab"}).then(data => console.log(data))
Promise {
  <pending>,
  [Symbol(async_id_symbol)]: 2857,
  [Symbol(trigger_async_id_symbol)]: 2853
}
> {
  _id: new ObjectId("644ddf84abb3044f0e3f9cab"),
  title: 'dune',
  year: 2022,
  rating: 9,
  isGood: true,
  __v: 0
}
> Movie.findById({_id}).then(data => console.log(data))
Uncaught ReferenceError: _id is not defined
> Movie.findById({"644ddf84abb3044f0e3f9cab"}).then(data => console.log(data))
Movie.findById({"644ddf84abb3044f0e3f9cab"}).then(data => console.log(data))
                ^^^^^^^^^^^^^^^^^^^^^^^^^^


Uncaught SyntaxError: Unexpected string
> Movie.findById("644ddf84abb3044f0e3f9cab").then(data => console.log(data))
Promise {
  <pending>,
  [Symbol(async_id_symbol)]: 3485,
  [Symbol(trigger_async_id_symbol)]: 3481
}
> {
  _id: new ObjectId("644ddf84abb3044f0e3f9cab"),
  title: 'dune',
  year: 2022,
  rating: 9,
  isGood: true,
  __v: 0
}
```