Mongoose Schema & Validation Cheatsheet

1. Import & Connect

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/mydb', { useNewUrlParser: true, useUnifiedTopology: true });
```

2. Define Schema with Validation

```
const postSchema = new mongoose.Schema({
  title: { type: String, required: true, trim: true },
  body: { type: String, required: true },
  category: { type: String, enum: ['News', 'Sports', 'Technology'] },
  likes: { type: Number, default: 0, min: 0 },
  tags: [String],
  createdAt: { type: Date, default: Date.now },
  email: {
    type: String,
    required: true,
    validate: {
      validator: v => /^\S+@\S+\.\S+$/.test(v),
      message: props => `${props.value} is not a valid email!`
    }
  }
});
```

3. Model

```
const Post = mongoose.model('Post', postSchema);
```

4. Create Document

```
const post = new Post({
  title: 'My First Post',
  body: 'Hello World!',
  category: 'News',
  likes: 5,
  tags: ['news'],
  email: 'user@example.com'
});

post.save()
  .then(doc => console.log(doc))
  .catch(err => console.log(err.message));
```

5. Update with Validation

```
Post.updateOne({ title: 'My First Post' }, { likes: 10 }, { runValidators: true });
```

6. Custom Async Validator Example

```
const userSchema = new mongoose.Schema({
  username: { type: String, required: true },
  email: {
    type: String,
    required: true,
    validate: {
      validator: async function(v) {
        const existing = await this.constructor.findOne({ email: v });
        return !existing;
      },
      message: 'Email already exists!'
    }
  }
});
```

7. Virtual Field Example

```
postSchema.virtual('summary').get(function() {
  return `${this.title} - ${this.category}`;
});
```

8. Schema Method Example

```
postSchema.methods.printTitle = function() {
  console.log('Title:', this.title);
};
```

9. Export Model

```
module.exports = mongoose.model('Post', postSchema);
```