Regular Expressions

3️⃣ Basic Regex Syntax (VERY IMPORTANT)

Regex is written between **slashes**:

/pattern/

Example:

/A/

➡ matches any string containing **A**

4️⃣ Using Regex in MongoDB

Basic syntax

```
{ <field>: { $regex: /pattern/, $options: '<options>' } }
OR
{ <field>: { $regex: 'pattern', $options: '<options>' } }
OR
{ <field>: { $regex: /pattern/<options> } }
OR
{ <field>: /pattern/<options> }
```

The $options clause takes the following as parameters:

i – Performs a case insensitive match

m – Performs pattern match that consists of anchors i.e. ^ for the beginning, $ for the end

x – Ignores all white space characters in the pattern

s – Allows dot character (.) to match all characters

Ex: Search for the products whose *prodname* starts with iphone:

```
db.product_catalog.find(
        { prodname:{ $regex:/^iphone/i } }
);
```

To retrive the *prodnames* that does not start with big

1. db.product_catalog.find(
2. { prodname: { $not: { $regex:/^big/i } } }

);

9️⃣ Predefined Character Classes (VERY USEFUL)

Symbol	Meaning
\d	digit (0–9)
\D	not digit
\w	word (a-z, A-Z, 0-9, _)
\W	not word
\s	space
\S	not space

Example:

/\d\d\d/

Matches:

- 123
- 456

MongoDB:

db.prod.find({ phone: /\d{10}/ })

🔟 Quantifiers { }

Exact count

/\d{4}/

➡ exactly 4 digits

Range

/\d{2,4}/

➡ 2 to 4 digits

MongoDB:

db.prod.find({ pincode: /^\d{6}$/ })

1️⃣1️⃣ $regex (Official MongoDB Way)

Instead of / /

```
db.prod.find({
  name: { $regex: "^App", $options: "i" }
})
```

Same as:

```
db.prod.find({ name: /^App/i })
```

1️⃣ Basic $regex Syntax (THIS IS THE FORMAT)

```
db.prod.find({
  field: { $regex: "pattern" }
})
```

Case-insensitive:

```
db.prod.find({
  field: { $regex: "pattern", $options: "i" }
})
```

2️⃣ Match Exact Word

Find name = Apple

```
db.prod.find({
  name: { $regex: "^Apple$" }
})
```

3️⃣ Starts With

Names starting with "App"

```
db.prod.find({
  name: { $regex: "^App" }
})
```

Case-insensitive:

```
db.prod.find({
  name: { $regex: "^app", $options: "i" }
})
```

4️⃣ Ends With

Names ending with "pro"

```
db.prod.find({
  name: { $regex: "pro$" }
})
```

5️⃣ Contains Word Anywhere

Contains "phone"

```
db.prod.find({
  name: { $regex: "phone", $options: "i" }
})
```

6️⃣ Match Any ONE Character (.)

```
db.prod.find({
  code: { $regex: "A.C" }
})
```

Matches:

- ABC
- A1C
- A-C

7️⃣ Zero or More (*)

```
db.prod.find({
  code: { $regex: "ab*" }
})
```

Matches:

- a
- ab
- abbbb

8️⃣ One or More (+)

```
db.prod.find({
  code: { $regex: "ab+" }
})
```

Matches:

- ab
- abbb

9️⃣ Zero or One (?)

```
db.prod.find({
  name: { $regex: "colou?r" }
})
```

Matches:

- color
- colour

🔟 Character Set [ ]

Match a, b, or c

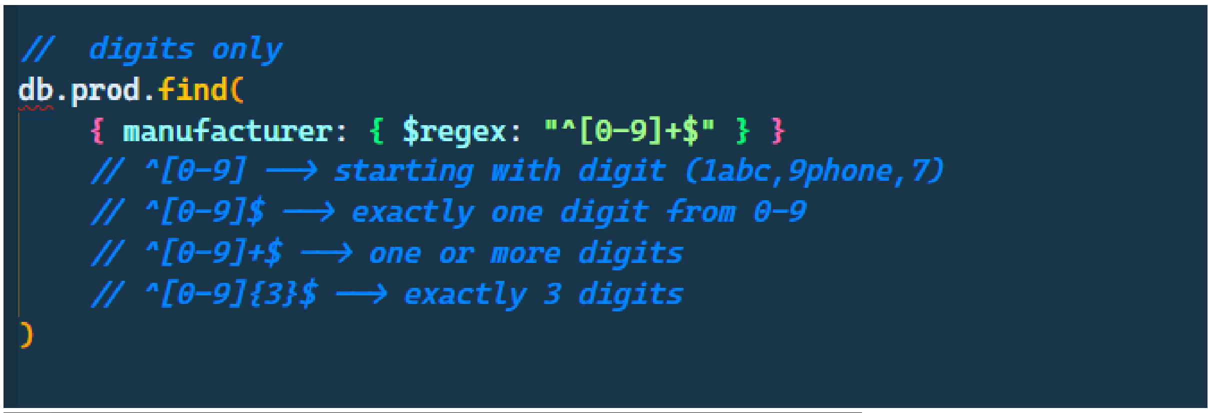
```
db.prod.find({
  tag: { $regex: "[abc]" }
})
```

Range a–z

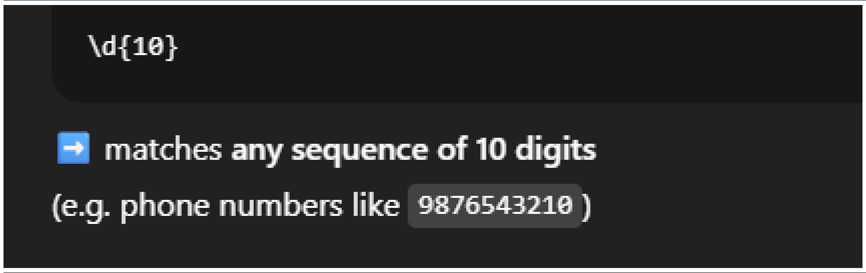
```
db.prod.find({
  name: { $regex: "[a-z]" }
})
```

Digits only

```
db.prod.find({
  price: { $regex: "^[0-9]+$" }
})
```



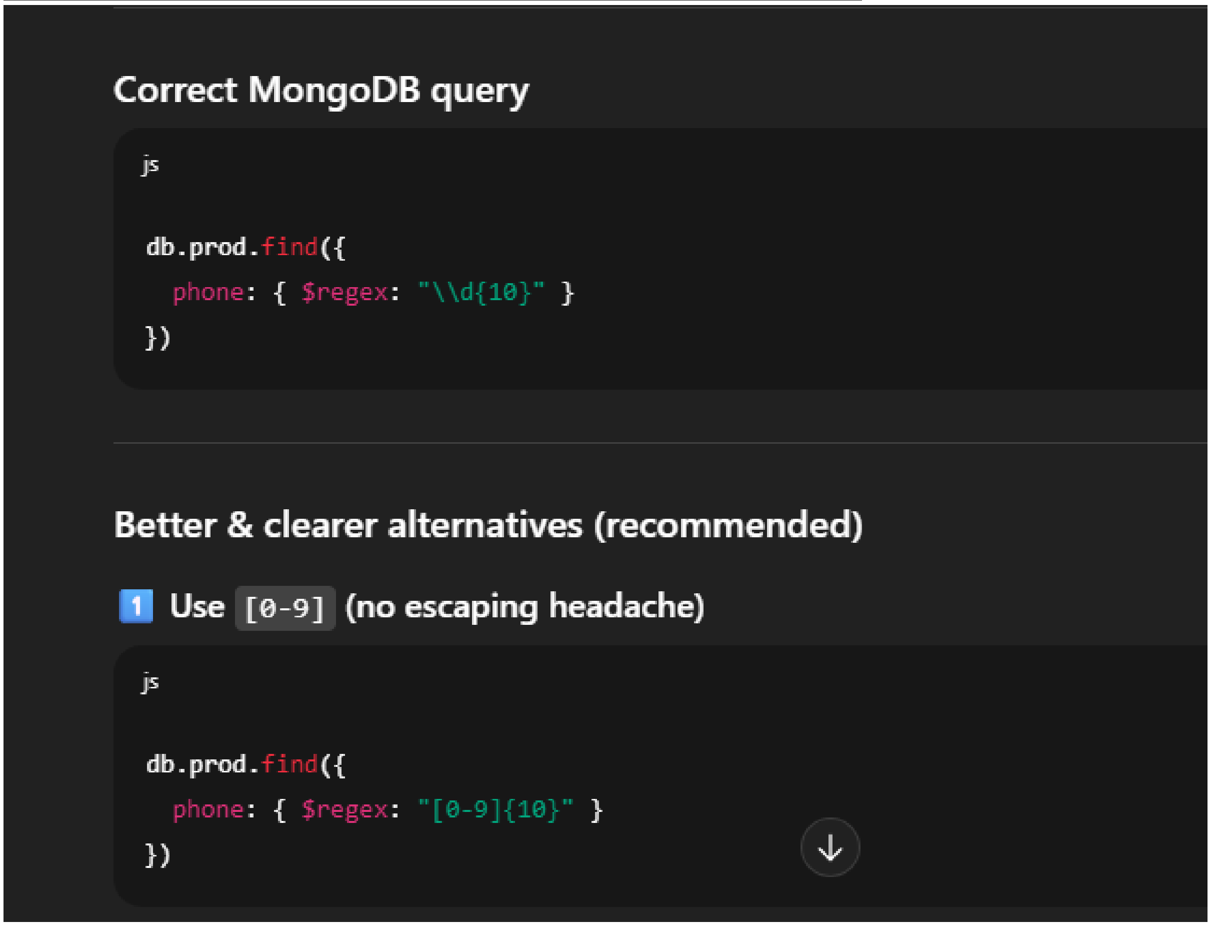
```js
//  digits only
db.prod.find(
    { manufacturer: { $regex: "^[0-9]+$" } }
    // ^[0-9] ⟶ starting with digit (1abc,9phone,7)
    // ^[0-9]$ ⟶ exactly one digit from 0-9
    // ^[0-9]+$ ⟶ one or more digits
    // ^[0-9]{3}$ ⟶ exactly 3 digits
)
```



```
\d{10}
```

➡️ matches **any sequence of 10 digits**

(e.g. phone numbers like `9876543210`)



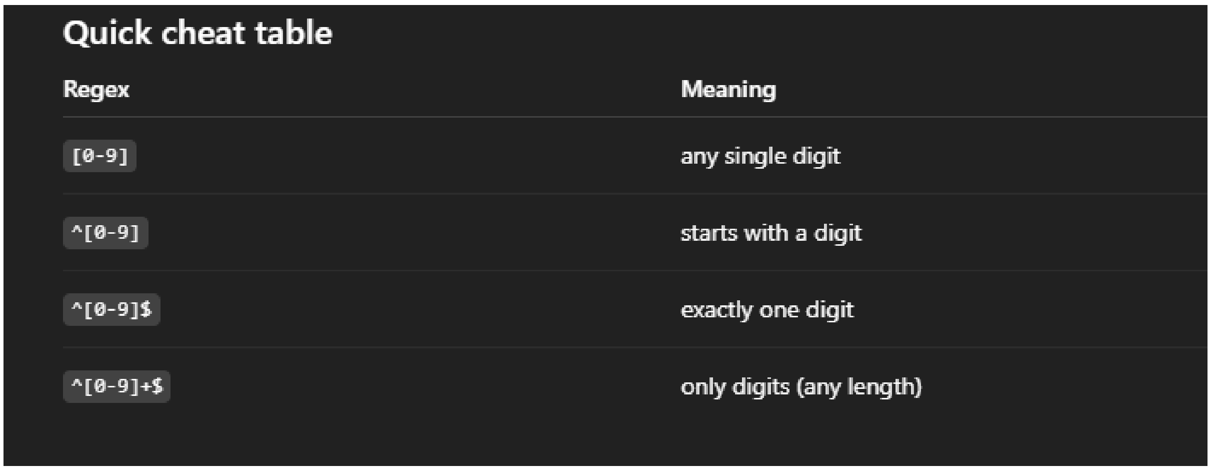
Correct MongoDB query

```js
db.prod.find({
  phone: { $regex: "\\d{10}" }
})
```

Better & clearer alternatives (recommended)

1️⃣ Use `[0-9]` (no escaping headache)

```js
db.prod.find({
  phone: { $regex: "[0-9]{10}" }
})
```



Quick cheat table

Regex	Meaning
`[0-9]`	any single digit
`^[0-9]`	starts with a digit
`^[0-9]$`	exactly one digit
`^[0-9]+$`	only digits (any length)

1️⃣1️⃣ NOT condition (^ inside [])

Not digits

```
db.prod.find({
  code: { $regex: "[^0-9]" }
})
```

1️⃣2️⃣ Predefined Classes

Digits

```
db.prod.find({
  phone: { $regex: "\\d{10}" }
})
```

⚠️ Note: **double backslash** \\ is mandatory

1️⃣ Use [0-9] (no escaping headache)

```
db.prod.find({
  phone: { $regex: "[0-9]{10}" }
})
```

2️⃣ Ensure it's exactly 10 digits

```
db.prod.find({
  phone: { $regex: "^[0-9]{10}$" }
})
```

3️⃣ Using regex literal (NO double slash needed)

```
db.prod.find({
  phone: { $regex: /\d{10}/ }
})
```

✔ Cleanest in Mongo shell

Starts with letter

```
db.prod.find({
  name: { $regex: "^[A-Za-z]" }
})
```

1️⃣3️⃣ Exact Length Using {}

Exactly 6 digits (pincode)

```
db.prod.find({
  pincode: { $regex: "^\\d{6}$" }
})
```

---

2 to 4 digits

```
db.prod.find({
  code: { $regex: "^\\d{2,4}$" }
})
```

1️⃣4️⃣ Indian Mobile Number

```
db.users.find({
  phone: { $regex: "^[6-9]\\d{9}$" }
})
```

1️⃣5️⃣ Email Examples

Gmail only

```
db.users.find({
  email: { $regex: "@gmail\\.com$" }
})
```

NOT Gmail

```
db.users.find({
  email: { $regex: "^(?!.*@gmail\\.com$).*" }
})
```

1️⃣6️⃣ Multiple $regex Conditions

AND

```
db.prod.find({
  name: { $regex: "^S" },
  category: { $regex: "Elect", $options: "i" }
})
```

OR

```
db.prod.find({
  $or: [
    { name: { $regex: "phone", $options: "i" } },
    { name: { $regex: "laptop", $options: "i" } }
  ]
})
```

1️⃣7️⃣ Very Common Interview Patterns

Only letters

{ $regex: "^[A-Za-z]+$" }

Alphanumeric

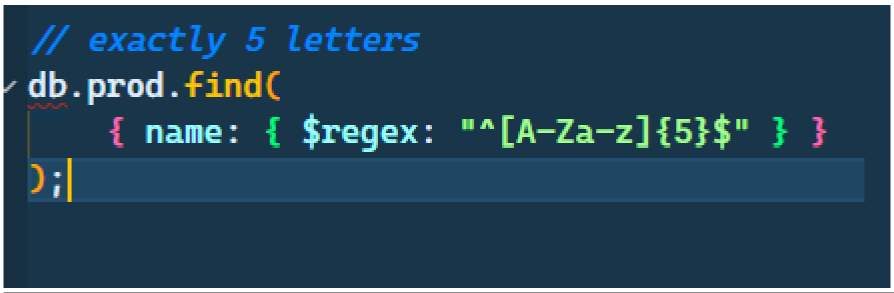
{ $regex: "^[A-Za-z0-9]+$" }

Strong password (simple)

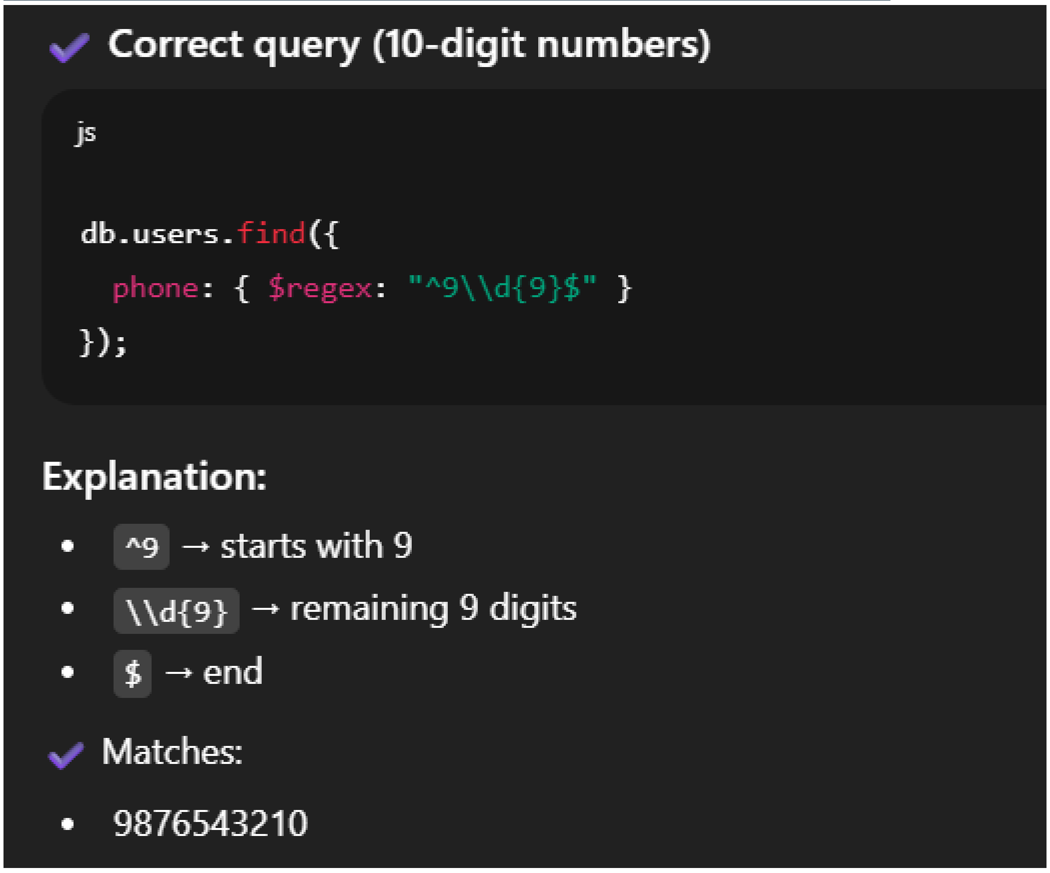
{ $regex: "^(?=.*[A-Z])(?=.*[a-z])(?=.*\\d).{8,}$" }

🔥 Mini Cheat Sheet

Need	Pattern
Starts with	^word
Ends with	word$
Exact match	^word$
Digits only	^[0-9]+$
Length N	^.{N}$
Case-insensitive	$options: "i"



```
// exactly 5 letters
db.prod.find(
    { name: { $regex: "^[A-Za-z]{5}$" } }
);
```



✓ **Correct query (10-digit numbers)**

js

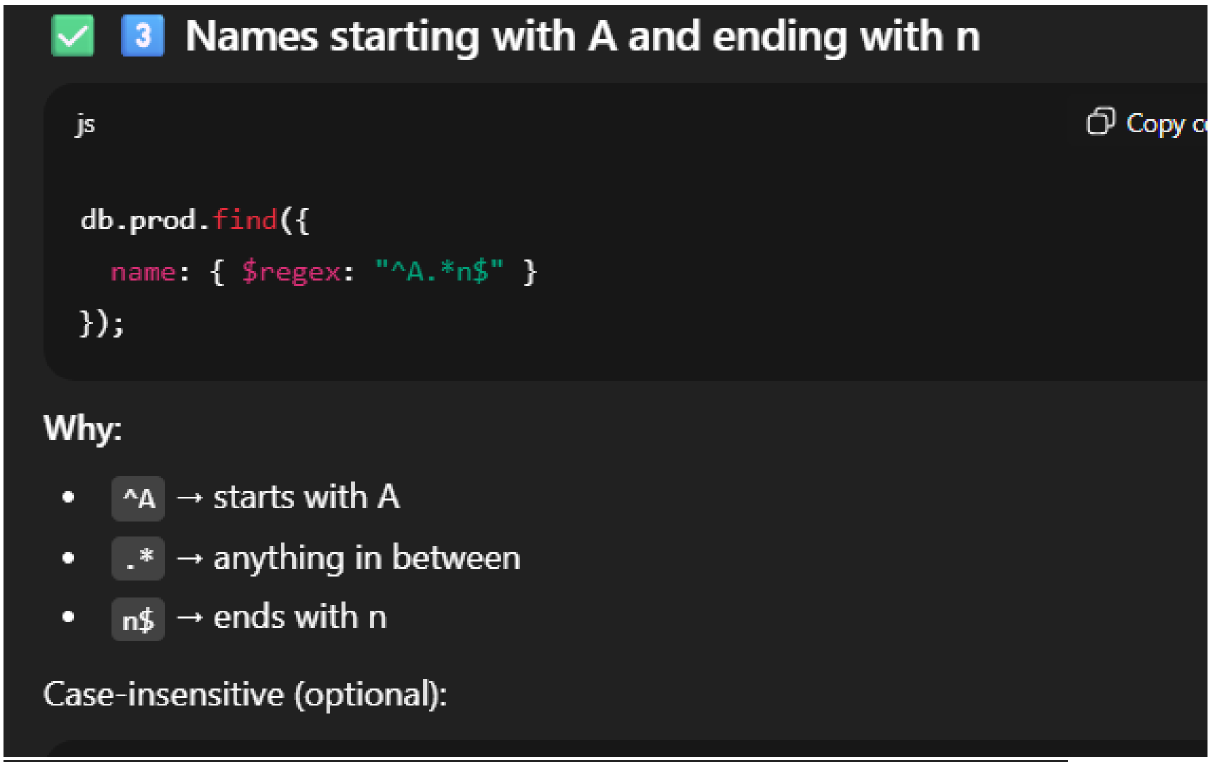
```js
db.users.find({
  phone: { $regex: "^9\\d{9}$" }
});
```

Explanation:

- `^9` → starts with 9
- `\\d{9}` → remaining 9 digits
- `$` → end

✓ Matches:

- 9876543210



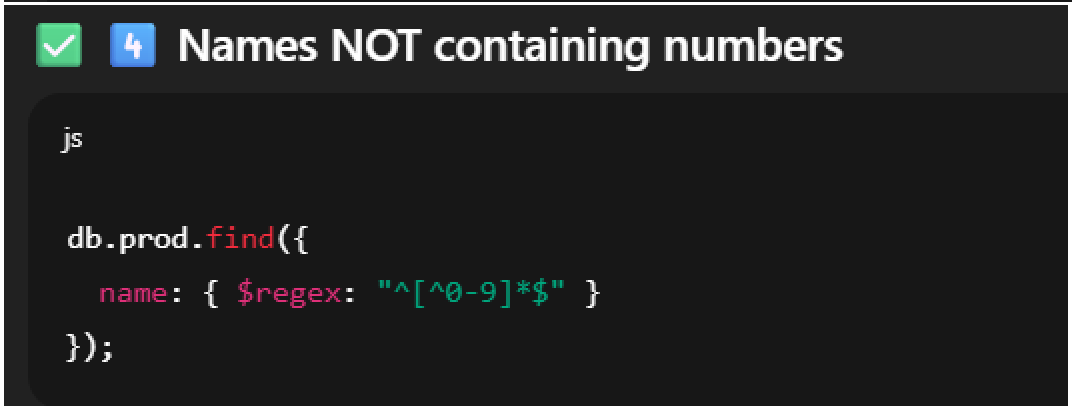
✅ 3️⃣ Names starting with A and ending with n

```js
db.prod.find({
  name: { $regex: "^A.*n$" }
});
```

Why:

- `^A` → starts with A
- `.*` → anything in between
- `n$` → ends with n

Case-insensitive (optional):



✅ 4️⃣ Names NOT containing numbers

```js
db.prod.find({
  name: { $regex: "^[^0-9]*$" }
});
```