```
// basic regular expression syntax

// /pattern/

// ex: /A/ --> matches string containing 'A'

use("product");

// 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

db.prod.find();

// find exact match
// find prod name=iphone 7
db.prod.find(
    { prodname: { $regex: "^iphone 7" } }
);

// starts with "cool"
db.prod.find(
    { prodname: { $regex: "^Cool" } }
);

// case insensitive
db.prod.find(
    { prodname: { $regex: "^cool", $options: "i" } }
);

// name ending (we use /word$)
db.prod.find(
    { prodname: { $regex: "ifb$" } }
);
```

```
// contains word anywhere
db.prod.find(
    { prodname: { $regex: "phone", $options: "i" } }
);

// other way
// starts with iphone
db.prod.find(
    {prodname:{$regex:/^iphone/i}} // same as {$regex:"^iphone",$options:"i"}
)

// retrieve products that does not start with big
db.prod.find(
    { prodname: { $not: { $regex: /^big/i } } }
);

// . is used to match any one character
db.prod.find(
    { manufacturer: { $regex: /g.o/ } } // same as {$regex:"g.o"} // matches g..c
any format
);

// '*' is used for 0 or more
db.prod.find(
    { manufacturer: { $regex: "go*" } } // means g must appear once, o can appear
any number of times(0 or more )
);

// one or more (+)
db.prod.find(
    { manufacturer: { $regex: "go+" } } // matches go,goo..
);

// zero or one (?)
db.prod.find(
    { manufacturer: { $regex: "goog?le" } }
);

//  character set [ ]
// match g,or o or ,e
db.prod.find(
    { manufacturer: { $regex: "[goe]" } }
);

// range a-z
db.prod.find(
    { manufacturer: { $regex: "[a-z]" } }
);

//  digits only
db.prod.find(
    { prodid: { $regex: "^[0-9]+$" } }
    // ^[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
)

// Not condition (keeping ^ inside [ ])
// not digits
db.prod.find(
    { prodid: { $regex: "[^0-9]" } }
);

// predefined classes
db.prod.find(
    { prodid: { $regex: "\\d{10}" } } //any sequence of 10digits
);

// starts with letter
db.prod.find(
    { prodname: { $regex: "^[A-Za-z]" } }
);

// exact length using { }
db.prod.find(
    {prodid:{$regex:"^\\d{6}$"}} // exact 6 digits
)

// 2 to 4 digits
db.prod.find(
    {prodid:{$regex:"^\\d{2,4}$"}}
)

// indian mobile number
db.prod.find(
    { phone: { $regex: "^[6-9]\\d{9}$" } }
);

// gmail example
db.prod.find(
    { email: { $regex: "@gmail\\.com$" } }
);

// not gmail
db.prod.find(
    { email: { $regex: "^(?!.*@gmail\\.com$).*" } }
);

// multiple regex conditions
// and
db.prod.find({
    prodname: { $regex: "^c" } ,
     "categories.main": { $regex: "elect",$options:"i"}
});

// or
db.prod.find(
    {
        $or: [
            { prodname: { $regex: "phone", $options: "i" } },
            { prodname: { $regex: "cool", $options: "i" } }
        ]
    }
);

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

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


// practice
db.prod.find(
    {name:{$regex:"^A"}}
)

db.prod.find(
    { name: { $regex: "er$" } }
);

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

//
```