In this lecture, we will discuss…

✧ Find By Criteria

- ‘lt’ & ‘gt’
- Evaluations
- Regex
- Exists
- Not
- Type

`Find` controls with lt and gt operator

-
  ```
  db[:zips].find(:city => {:$lt => 'D'}).limit(2).to_a.each { |r| pp r}
  ```

-
  ```
  db[:zips].find(:city => {:$lt => 'P', :$gt => 'B'}).limit(3).to_a.each { |r| pp r}
  ```

Find By - Regex

- Regex – supports regular expression capabilities for pattern matching *strings* in queries.
- `db[:zips].find(:city => {:$regex => 'X'}).limit(5).each {|r| pp r}`
    - Will retrieve cities containing X in their names (5 documents only)

Find By - Regex

- `db[:zips].find(:city => {:$regex => 'X$'}).limit(5).each {|r| pp r}`
- Displays cities ending with X

Find By - Regex

-
  ```
  db[:zips].find(:city => {:$regex =>
  '^X'}).projection({:_id =>
  false}).limit(5)
  .to_a.each {|r| pp r}
  ```
- Displays cities starting with X

Find By - Regex

-
  ```
  db[:zips].find(:city => {:$regex => '^[A-E]'}).projection({:_id => false})
  .limit(5).to_a.each {|r| pp r}
  ```
- Displays cities that match the regex (A to E)

$exists

- Will check to see of the document exists when the boolean is true

-
  ```
  db[:zips].find(:city => {:$exists => true}).projection({:_id => false}).limit(3).to_a.each {|r| pp r}
  ```

$not

- `$not` performs a logical NOT operation
- Selects the documents that do not match the <operator-expression>

-
  ```
  db[:zips].find(:pop =>
  {'$not' => {'$gt' => 9500}}).projection
  ({_id:false}).limit(20).to_a.each
  {|r| pp r}
  ```

`$type`

- `$type` – selects the documents where the value of the field is an instance of the specified numeric BSON type

- Handy when dealing with unstructured data where data types are not predictable

-
  ```
  db[:zips].find({:state=> {:$type =>
  2}}).first
  ```

Summary

- ✧ Find by (Evaluations, Regex, Exists, Not, Type) provides an useful way to fetch/filter data from the collection

What’s Next?

- ✧ replace_one
- ✧ update_one
- ✧ update_many
- ✧ delete_one
- ✧ delete_many
- ✧ upsert