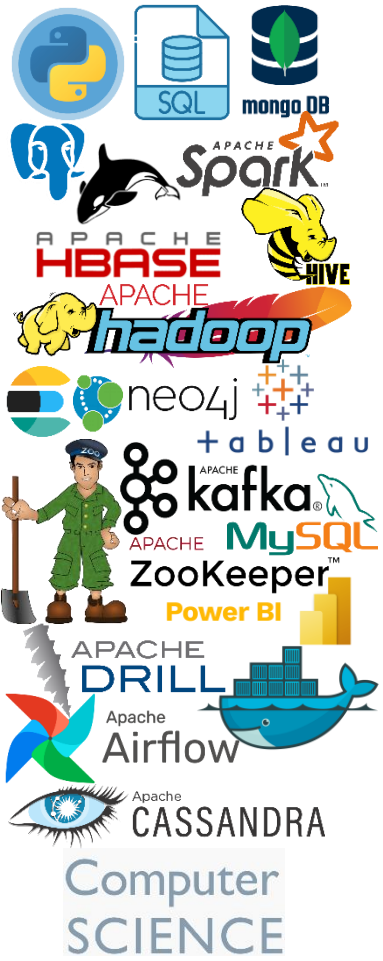
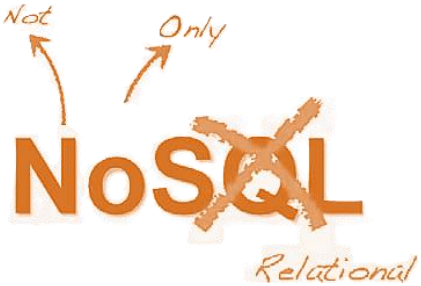
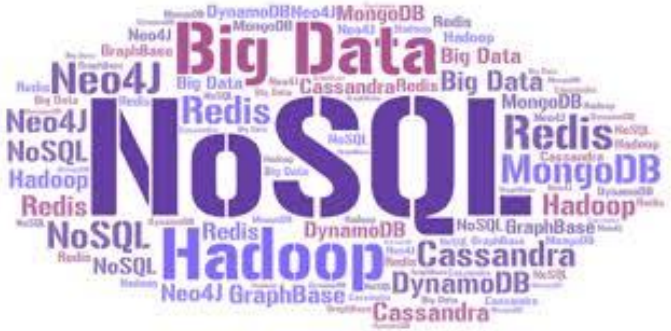






Dr. SOHAIL IMRAN

Commands

sohailimran2@yahoo.com

Ref: Web

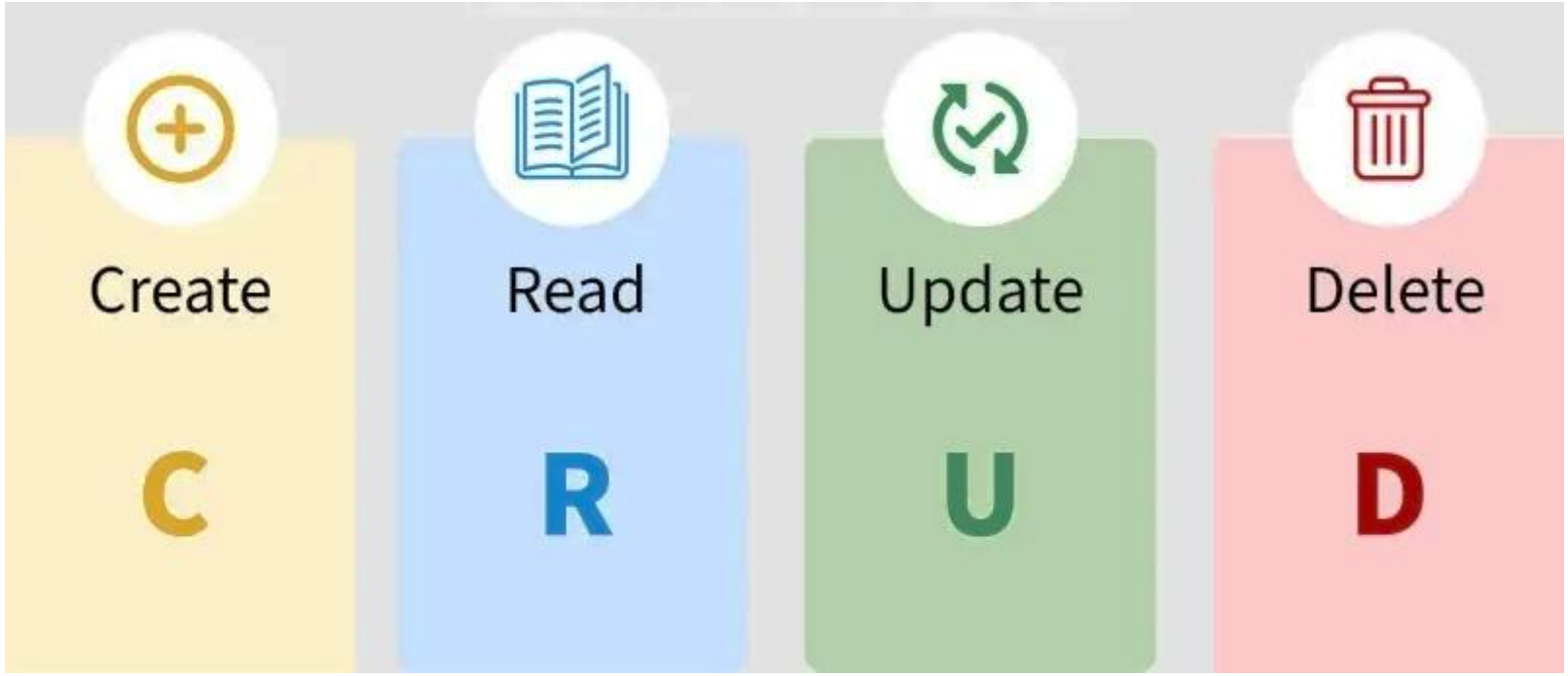
db

To check which db you're using	:	db
Show all databases	:	show dbs
Switch db's/make a new one	:	use <name>
See what collections exist	:	show collections

_id

- By default, each document contains an _id field. This field has a number of special characteristics:
  - Value serves as primary key for collection.
  - Value is unique, immutable, and may be any non-array type.
  - Default data type is ObjectId, which is “small, likely unique, fast to generate, and ordered.” Sorting on an ObjectId value is roughly equivalent to sorting on creation time.

C R U D



To insert documents into a collection/make a new collection:

```
db.<collection>.insert(<document>)
```

- Done on collections.
- Get all docs: `db.<collection>.find()`
  - Returns a cursor, which is iterated over shell to display first 20 results.
  - Add .limit(<number>) to limit results

- Get one doc: `db.<collection>.findOne()`

```
db.<collection>.update(
{<field1>:<value1>},      //all docs in which field = value
{$set: {<field2>:<value2>}},          //set field to value
{multi:true} )                        //update multiple docs
```

upsert: if true, creates a new doc when none matches search criteria.

Remove all records where field = value

```
db.<collection>.remove({<field>:<value>})
```

As above, but only remove first document

```
db.<collection>.remove({<field>:<value>}, true)
```

find

Including/excluding document fields

```
db.<collection>.find({<field1>:<value>}, {<field2>: 0})
```

```
db.<collection>.find({<field>:<value>}, {<field2>: 1})
```

Find documents with or w/o field

```
db.<collection>.find({<field>: { $exists: true}})
```

AND

To match a specific value:

```
db.<collection>.find({<field>:<value>})
“AND”
db.<collection>.find({<field1>:<value1>,
                      <field2>:<value2>
                     })
```

OR

```
OR
db.<collection>.find({ $or: [
<field>:<value1>
<field>:<value2>          ]
})
```

Checking for multiple values of same field

```
db.<collection>.find({<field>: {$in [<value>, <value>]}})
```

update

To remove a field

```
db.<collection>.update({<field>:<value>},
                       { $unset: { <field>: 1}})
```

Replace all field-value pairs

```
db.<collection>.update({<field>:<value>},
                       { <field>:<value>,
                         <field>:<value>})
```

*NOTE: This overwrites ALL the contents of a document, even removing fields.

Comparison

Name	Description
$eq	Matches values that are equal to a specified value.
$gt	Matches values that are greater than a specified value.
$gte	Matches values that are greater than or equal to a specified value.
$in	Matches any of the values specified in an array.
$lt	Matches values that are less than a specified value.
$lte	Matches values that are less than or equal to a specified value.
$ne	Matches all values that are not equal to a specified value.
$nin	Matches none of the values specified in an array.

Logical

Name	Description
$and	Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.
$not	Inverts the effect of a query expression and returns documents that do *not* match the query expression.
$nor	Joins query clauses with a logical NOR returns all documents that fail to match both clauses.
$or	Joins query clauses with a logical OR returns all documents that match the conditions of either clause.