    ///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  3. GUN  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

////\*-----------------------------------------------------------------

//// 5) Maasi en az olan personelin adini, noname, ise baslama tarihini

////    nodate, maasini ise 0 yapiniz.

////-----------------------------------------------------------------\*/

db.personel.findOneAndReplace({},{"isim":"Noname", "iseBaslama":"nodate" "maas":0}, {sort:{"maas":1}});

////-----------------------------------------------------------------

//// 6) Maasi 7250 olan personelin maasini 8000 yapiniz, diger kismilari

////    degistirmeyiniz.

////-----------------------------------------------------------------

db.personel.findOneAndUpdate({"maas":7200},{$set:{"maas":8000}});

////-----------------------------------------------------------------

//// 7) Maasi 7500 olan personelin maasini %10 zam yapiniz,

////     diger kismilari degistirmeyiniz.

////-----------------------------------------------------------------

db.personel.findOneAndUpdate({"maas":7500},{$mul:{"maas":1.1}});

////-----------------------------------------------------------------

//// 8) Maasi 4800 olan personelin maasini 500 TL zam yapiniz,

////     diger kismilari degistirmeyiniz.

////-----------------------------------------------------------------

db.personel.findOneAndUpdate({maas:4800},{$inc:{maas:500}})

////-----------------------------------------------------------------

//// 9) Maasi 15000 olan personelin Adini "Mehmet Can" ve ise baslama

////    tarihini su an olarak degisitriniz.diger kismilari degistirmeyiniz.

//// Eğer böyle bir kayıt yoksa, yeni kayıt olarak ekle.

////--------------------------------------------------------

db.personel.findOneAndUpdate({maas:15000},

{$set: {isim:"Mehmet Can", iseBaslama:new Date()}},{upsert:true});

// NOT: Eger günceleme (replace veya update ) işleminde sorgu ile eslesmme dumum

//      olduğunda dokumanın oluşturulması istenilirse {upstert:true} kullanılmalıdır.

////=================================================================

////           updateOne - updateMany - update (depreceated)

////

////  update komutlari ile de bir dokumandaki bigilerin bir kismi yada

////  tamami guncellenebilir. update komutlari findOneAndUpdate komutuna

////  gore daha hizli calisir. Bunun sebebi bu komutlarin sorguyu cevap

////  olarak dondurmemesidir. Bu komutlar sadece istatistik olarak kisa

////  bilgiler dondurur.

////=================================================================

////-----------------------------------------------------------------

//// 9)ismi can tak olan personelin maaşı 15000 olarak günceleyin.

////-----------------------------------------------------------------

db.personel.updateOne({isim:"Can Tak"},{$set:{maas:15000}});

////-----------------------------------------------------------------

//// 11) maasi 15000 olan personellerin dokumanina prim:1000 bilgisini ekleyiniz.

////-----------------------------------------------------------------

db.personel.updateMany({maas: 15000}, {$set: {prim: 1000}});

////-----------------------------------------------------------------

//// 12) maasi 5000 olan personellerin dokumanina prim:200 bilgisini

///     eger boyle bir personel yok ise yeni bir kayit aciniz.

////-----------------------------------------------------------------

db.personel.updateMany({maas: 5000}, {$set: {prim: 200}}, {upsert:true});

////-----------------------------------------------------------------

//// 13) maasi 7000 az olan personellerin dokumanina prim:500 bilgisini ekleyiniz.

////-----------------------------------------------------------------

db.personel.update({"maas":{$lt:7000}},{$set:{"prim":500}}, {multi:true});

////=================================================================

////           deleteOne - deleteMany - remove (depreceated)

////

////=================================================================

////-----------------------------------------------------------------

//// 12) ismi Mehmet Can olan kaydi siliniz.

////-----------------------------------------------------------------

db.personel.deleteOne({isim:"Mehmet Can"});

////-----------------------------------------------------------------

//// 13) maasi 15000 olan dokumanlari siliniz.

//-----------------------------------------------------------------

db.personel.deleteMany({maas:15000});

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

////==================================================================================

////                                  AGGREGATION

////==================================================================================

// 1) Aggregiation, dokumanlardaki verilerin islenmesi ve  hesaplanan

//    sonuclarin donmesini saglayan islemlerdir.

//

// 2) Aggregation islemleri, farklı dokumanlardaki degerleri gruplandirabilir.

//

// 3) Bu gruplanan veriler uzerinde cesitli islemlerin gereceklestirelerek tek

//    bir sonuc degerinin donmesi saglanabilir.

//

//  4) MongoDB, 3 farklı yontem ile aggregation gerceklestirmeye izin verir.

//     A) aggregation pipeline

//     B) map-reduce function

//     C) single-purpose aggregation

//

//  5) Aggregiation, SQL'deki Join islemlerine benzetilebilir.

//

//     SQL         MONGO

//----------------------

//     WHERE       $match

//     GROUP BY    $group

//     HAVING      $match

//     SELECT      $project

//     ORDER BY    $sort

//     LIMIT       $limit

//     SUM()       $sum $count

//     COUNT()     $sortByCount

//     JOIN        $lookup

//     MERGE       $merge (MongoDB 4.2 ve sonrasi)

////==================================================================================

///==================================================================================

////                          AGGREGATION PIPELINE

////==================================================================================

// SYNTAX

//

//   pipeline = [

 //   { $match : { … },

 //   { $group : { … },

 //   { $sort : { … },

//      ...

//   ]

//   db.collectionName.aggregate(pipeline, options)

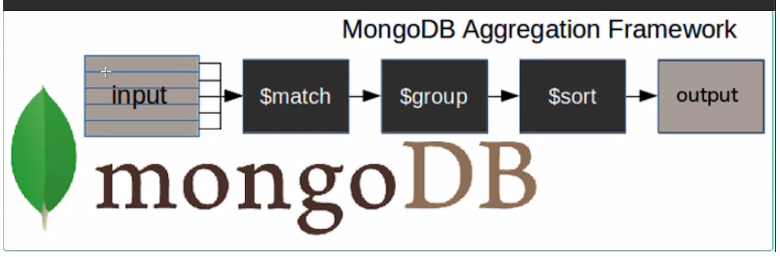
//

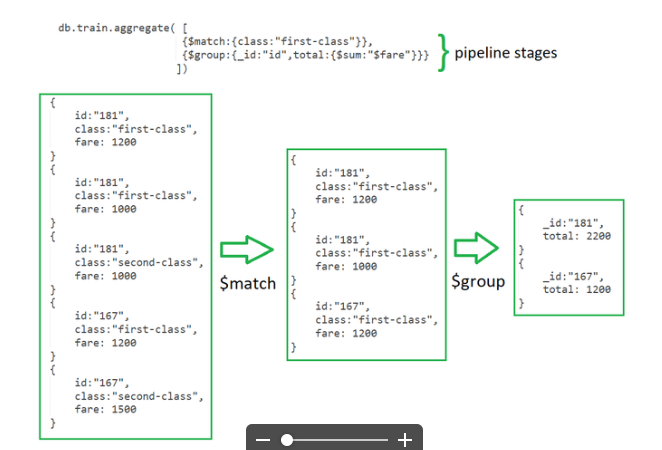
//  $match() – Verileri secerken flitrelemek icin

//  $group() - islenen verilerin grupplanmasi icin

//  $sort()  Sonuclarin siralanmasi icin

////





////---------------------------------------------------------------------------------

//// 1) "kisiler" veritabaninda "ogrenciler" collection'i tanimlayalim.

////---------------------------------------------------------------------------------

use("kisiler");

db.createCollection("ogrenciler");

////---------------------------------------------------------------------------------

//// 2) asagidaki dokumanlari collection'a kaydedelim.

// ////---------------------------------------------------------------------------------

db.ogrenciler.insertMany([

{"isim": "Tarik", "yas":17, "sube": "A", "not":97, "dersler":["fizik", "mat"]},

{"isim": "Selim", "yas":37, "sube": "A","not":77,},

{"isim": "Nalan", "yas":17, "sube": "B","not":55, "dersler":["fizik", "inglizce"]},

{"isim": "Burhan", "yas":21, "sube": "B","not":43, "dersler":["fizik", "mat"]},

{"isim": "Niyazi", "yas":40, "sube": "B","not":99, "dersler":["inglizce"]},

{"isim": "Galip", "yas":37, "sube": "A","not":34, "dersler":["inglizce"]},

{"isim": "Teslime", "yas":24, "sube": "A","not":88, "dersler":["mat","fizik","kimya"]}]);

////---------------------------------------------------------------------------------

//// 3) Yaşı 30'dan büyük olan öğrencileri sorgulayınız

// ////---------------------------------------------------------------------------------

db.ogrenciler.aggregate({$match:{yas:{$gt:30}}});

////------------------------------------------------------------------------

//// 4) B şubesindeki yaşı 30 dan küçük olanları listeleyin

//----------------------------------------------------------------------------

db.ogrenciler.aggregate({$match:{sube:"B", yas:{$lt:30}}});

////------------------------------------------------------------------------

//// 5) B şubesindeki ogrencilerin sayısını yazınız.

//// NOT: saymak için $MATCH içerisinde $COUNT methodu kullanılır

//----------------------------------------------------------------------------

db.ogrenciler.aggregate( {$match:{sube:"B"}}, {$count:"B-Subesi ogrenci sayisi"})

// NOT:Sadece saymak icin dogrudan db.colection.count() metoduda kullanilabilir.

//     Ancak bu islem aggregetion’daki gibi sorgu cevabi dondurmez. Sadece hesaplanan

//     sonucu dondurur.

// Bu yöntem single purpose aggregetion methodu denilmektider.

db.ogrenciler.count({sube:"B"});

// Bu yontemde sadece sonuç yazar. açıklama yazılamaz

////-----------------------------------------------------------------

//// 6) Tum subelerdeki toplam ogrenci sayisini hesaplayan sorguyu yaziniz.

////    NOT: Gruplandirirak saymak icin $sum:1 kullanilabilir. $sum:1, eslesen

////    her bir grup icin 1 sayisinin donmesini ve bunlarin toplanmasini saglar.

////-------------------------------------------

db.ogrenciler.aggregate({$group:{\_id:"$sube",toplamOgrenci:{$sum:1}}})

// {$sum:1} teker teker sayarak sonucu yazdırır.

////---------------------------------------------------------------------------------

//// 7) Tum subelerdeki toplam ogrenci sayilarini ve her subedeki maks notu hesaplayan

////    sorguyu yaziniz.

////  NOT: Group isleminde degiskenlerin onune $ isareti konulmalidir.

////-----------------------------------

db.ogrenciler.aggregate({$group:{\_id:"$sube", toplamOgrenci:{$sum:1},maxNot:{$max:"$not" }}});

////-------------------------------------------------------------------

//// 8) yasi 30'Dan kucuk olanlarin subelere gore not ortalamalarinihesaplayan

////    sorguyu yaziniz.

db.ogrenciler.aggregate([

{$match:{yas:{$lt:30}}},

{$group:{\_id:"$sube", ortlamaNot:{$avg:"$not"}}}

]);