1. Grouped by state to get the count of female that live in such a state.

* db*.*persons*.*aggregate([ { *$match*: {*gender*: 'female'} },  
   { *$group*: { *\_id*: { *state*: '$location.state' }, *totalPersons*: { *$sum*: 1 } } },  
   { *$sort*: { *totalPersons*: -1 } }  
  ])*.*pretty()

1. Find the average and also get two fields.

* db*.*persons*.*aggregate([  
   {*$match*: {'dob.age': {*$gt*: 50}}},  
   {*$group*: {*\_id*: { *gender*: '$gender'}, *thereIs*: {*$sum*: 1}, *averageAge*: {*$avg*: '$dob.age'}}}  
  ])

1. Concatenate fields and project the ones that you need.

* db*.*persons*.*aggregate([  
   {*$project*: {*\_id*: 0, *gender*: 1, *fullName*: {*$concat*: [{*$toUpper*: '$name.first'}, ' ', {*$toLower*: '$name.last'}]}}}  
  ])
* db*.*persons*.*aggregate([  
   {*$project*: {*\_id*: 0, *gender*: 1, *fullName*: {*$concat*: [  
   {*$toUpper*: {*$substrCP*: ['$name.first', 0, 1]}},  
   {*$substrCP*: ['$name.first', 1, {*$subtract*: [{*$strLenCP*: '$name.first'},1]}]},  
   ' ',  
   {*$toLower*: '$name.last'}  
   ]}}}  
  ])
* db*.*persons*.*aggregate([  
   {*$project*: {  
   *\_id*: 0,  
   *name*: 1,  
   *email*: 1,  
   *gender*: 1,  
   *location*: {  
   *type*: 'Point',  
   *coordinates*: [  
   {*$convert*: {*input*: '$location.coordinates.longitude', *to*: 'double', *onError*: 0, *onNull*: 0}},  
   {*$convert*: {*input*: '$location.coordinates.latitude', *to*: 'double', *onError*: 0, *onNull*: 0}}  
   ]  
   }  
   }},  
   {*$project*: {*\_id*: 0, *gender*: 1, *location*: 1, *fullName*: {*$concat*: [  
   {*$toUpper*: {*$substrCP*: ['$name.first', 0, 1]}},  
   {*$substrCP*: ['$name.first', 1, {*$subtract*: [{*$strLenCP*: '$name.first'},1]}]},  
   ' ',  
   {*$toLower*: '$name.last'}  
   ]}}}  
  ])
* db*.*persons*.*aggregate([  
   {*$project*: {  
   *\_id*: 0,  
   *birthdate*: {*$toDate*: '$dob.date'},  
   *age*: '$dob.age',  
   *email*: '$email'  
   }  
   }  
  ])

1. Hobbies that people have grouped by age.

* db*.*friends*.*aggregate([  
   {*$group*: {*\_id*: { *age*: '$age' }, *allHobbies*: {*$push*: '$hobbies'}}}  
  ])*.*pretty()

1. Get element out of an array and put them into another.

* db*.*friends*.*aggregate([  
   {*$unwind*: '$hobbies'},  
   {*$group*: {*\_id*: { *age*: '$age' }, *allHobbies*: {*$push*: '$hobbies'}}}  
  ])*.*pretty()

(no duplicity)

* db*.*friends*.*aggregate([  
   {*$unwind*: '$hobbies'},  
   {*$group*: {*\_id*: { *age*: '$age' }, *allHobbies*: {*$addToSet*: '$hobbies'}}}  
  ])*.*pretty()

1. Project and slice the elements in an array.

* db*.*friends*.*aggregate([  
   {*$project*: {*\_id*: 0, *examScore*: {*$slice*: ['$examScores', 1]}}}  
  ])

1. Length of a projected array

* db*.*friends*.*aggregate([  
   {*$project*: {*\_id*: 0, *exams*: {*$size*: '$examScores'}}}  
  ])

1. Filtering an array in the projection phase

* db*.*friends*.*aggregate([  
   {*$project*: {*\_id*: 0, *examScores*: {*$filter*: {*input*: '$examScores', *as*: 'sc', *cond*:{*$gt*: ['$$sc.score', 60]}}}}}  
  ])

1. Manipulate an array and get a certain value from it.

* db*.*friends*.*aggregate([  
   {*$unwind*: '$examScores'},  
   {*$project*: {*\_id*: 1, *name*: 1, *age*: 1, *score*: '$examScores.score'}},  
   {*$sort*: {*score*: -1}},  
   {*$group*: {*\_id*: '$\_id', *name*: {*$first*: '$name'}, *maxScore*: {*$max*: '$score'}}},  
   {*$sort*: {*maxScore*: -1}}  
  ])

BUCKETS

1. creation of buckets or category brackets that we can create.

* db*.*persons*.*aggregate([  
   {*$bucket*: {*groupBy*: '$dob.age', *boundaries*: [18,30,40,50,60,120], *output*: {*numPersons*: {*$sum*: 1}, *averageAge*: {*$avg*: '$dob.age'}}}}  
  ])  
    
  db*.*persons*.*aggregate([  
   {*$bucketAuto*: {*groupBy*: '$dob.age', *buckets*: 5, *output*: {*numPersons*: {*$sum*: 1}, *averageAge*: {*$avg*: '$dob.age'}}}}  
  ])

GEO

1. Gets the closes point using aggregation framework.

* db*.*transformedPersons*.*aggregate([  
   {  
   *$geoNear*: {  
   *near*: {*type*: 'Point', *coordinates*: [-18.4, -42.8]},  
   *maxDistance*: 100000,  
   *$limit*: 10,  
   *query*: {*age*: {*$gt*: 30}},  
   *distanceField*: 'distance'  
   }  
   }  
  ])