Please indicate below the tool you have analyzed for Homework 3, and past the code you have realized.
Your code must include all queries and indexes (if any), and possibly the script you have used to populate the database (in case you used a tool for this, e.g., studio 3T for MongDB, please specify). If you have interacted with the database system through an external programming language, e.g., Pyhton, insert your functions/program.

==
By Leonardo Placidi and Negin Amininodoushan

Our Project has been realized using MongoDB through Studio 3T.

Here follows the code we used from the creation of the database to the queries.
Our tables were all exported from MySQL using the import wizard, so after we created the collection on MongoDB, we imported them through to Studio 3T shortcuts.

Creation of the 4 kinds of databases with the same data:

1) 5 different tables: jan, fsumm, depperf, candiv, arrperf equivalent to our old MySQL db.
2) 1 FinalNested collection starting from jan, obtained from lookup.
3) 1 uniquecollection with all the fields from all the tables at the same level.
4) 1 UniqueIndexed, which is the equivalent to 3) with indexes.

We did this to evaluate difference in the sintax and performance, concluding through the queries that, as it should be, the best database was the 4).

For the reader, in this text we put the comments of the code in this color.

Here follows the code.

//Creation of the Collections and first definitions

```
//This first 5 tables consist in the equivalent to the relational schema we used in MySQL
db.createCollection("jan");
db.createCollection("fsumm");
db.createCollection("depperf");
db.createCollection("candiv");
db.createCollection("arrperf");

db.createCollection("uniquecollection"); //cumulative unnested collection
db.createCollection("UniqueIndexed"); //here start importing the cumulative table later will be
//indexed
//NOW imports using the tools of Studio 3T, in particular after cliking on the collection to import
//from the ones we declared, e.g. jan, we clicked on the import window on Studio 3T, then
//selected the json import and selected the files we exported aslready using the export wizard on
//MySQL.

//let's now create a unique collection from january and with nested all the other collections of our
//db
//using lookup, saving in intermidiate 'bla' collections outing at last
//in the collection FinalNested

db.jan.aggregate([
   {
  $lookup: {
        from: "fsumm",
        localField: "_id",    // field in the jan
        foreignField: "_id",  // field in the other
        as: "fsumm",
     }},
      {$out:'bla'}

])

db.bla.aggregate([
   {
  $lookup: {
        from: "depperf",
        localField: "_id",
        foreignField: "_id",
        as: "depperf",
     }},
      {$out:'bla'}

])

db.bla.aggregate([
   {
  $lookup: {
        from: "candiv",
        localField: "_id",
        foreignField: "_id",
        as: "candiv",
     }},
      {$out:'bla'}

])

db.bla.aggregate([
   {
  $lookup: {
        from: "arrperf",
        localField: "_id",
        foreignField: "_id",
        as: "arrperf",
     }},
      {$out:'FinalNested'}   //final unique collection with embedded documents
])
```

//QUERIES

//Now let's do some queries!!!

'''Query 1'''
//Which carriers in 2020 had most delayed flight in the first 15 minutes? (had more delayed
//flights)

```
//First version on database 1)
db.depperf.aggregate([
{ $match: { Year : 2020, DEP_DEL15 : 1.0}} ,
{$lookup:
{ from:"jan",
localField:"_id",
foreignField:"_id",
as:"delay_details" }},
{ $unwind : "$delay_details" },
{$group : { _id:  "$delay_details.OP_CARRIER_AIRLINE_ID", count:{$sum:1}}},
{ $sort : { count : -1 } },
{ $limit : 10 }
])
//This first version returns in ca 5 seconds

//Second version on 2)
db.FinalNested.aggregate([
{ $match: {  Year : 2020, 'depperf.DEP_DEL15' : 1.0  }},
{$group : { _id:  "$OP_CARRIER_AIRLINE_ID", count:{$sum:1}}},
{ $sort : { count : -1 } },
{ $limit : 10 }
])
//This second version returns in ca 1.6 seconds

//Third version on 3)
db.uniquecollection.aggregate([
{ $match: {  Year : 2020, 'DEP_DEL15' : 1.0  }},
{$group : { _id:  "$OP_CARRIER_AIRLINE_ID", count:{$sum:1}}},
{ $sort : { count : -1 } },
{ $limit : 10 }
])
//This third version returns in ca 1.1 seconds

//Fourth and final Version INDEXED on database 4)
db.UniqueIndexed.createIndex({'OP_CARRIER_AIRLINE_ID':1})
db.UniqueIndexed.createIndex({'DEP_DEL15':1})
db.UniqueIndexed.createIndex({'Year':1})

db.UniqueIndexed.aggregate([
{ $match: {Year : 2020,  'DEP_DEL15' : 1.0  }},
{$group : { _id:  "$OP_CARRIER_AIRLINE_ID", count:{$sum:1}}},
{ $sort : { count : -1 } },
{ $limit : 10 }
])
//This Fourth version returns in ca 0.6 seconds as a document based management system was
// what we have expected!! The fourth version is definetely the best one.

//db.UniqueIndexed.dropIndex('OP_CARRIER_AIRLINE_ID'); and so on in case we want to
//delete indexes

'''Query 2'''
//Which carriers in 2010 have worse performance(count of delayed flights over all flights) than
//the worst carrier in 2020?

db.jan.aggregate([
{ $match: { Year : 2020  }}  ,
{ $project : { _id : 1  ,OP_CARRIER_AIRLINE_ID:1}},

{$lookup:
{ from:"depperf"
```

```
,
localField:"_id",
foreignField:"_id",
as:"delayed_details" }} ,
{ $unwind : "$delayed_details" },
{$project:{ 'carrier_Id' :  '$OP_CARRIER_AIRLINE_ID' , 'delayed' :
'$delayed_details.DEP_DEL15' }}    ,
{ $group: { _id: "$carrier_Id" ,
          count:{$sum:1},
          delayed: {$sum: "$delayed"}}}  ,
 { $project: {'carrier_Id' : '$carrier_Id' , 'performance' : { "$divide": [ "$delayed", "$count" ] } }},
 {$sort :{ performance: -1 }} ,
{ $limit : 1 }
 ])

// so we got the worst performance in 2020, such a long way compared to SQL!
// What is the performances that are worse than this value in 2010


db.jan.aggregate([
{ $match: { Year : 2010 }}  ,
   { $project : { _id : 1  ,OP_CARRIER_AIRLINE_ID:1}},

{$lookup:
{ from:"depperf",
localField:"_id",
foreignField:"_id",
as:"delayed_details" }},
{ $unwind : "$delayed_details" },
{$project:{ 'carrier_Id' :  '$OP_CARRIER_AIRLINE_ID' , 'delayed' :
'$delayed_details.DEP_DEL15' }}    ,
{ $group: { _id: "$carrier_Id" ,
          count:{$sum:1},
          delayed: {$sum: "$delayed"}}}  ,
{ $project: {'carrier_Id' : '$carrier_Id' , 'performance' : { "$divide": [ "$delayed", "$count" ] } }},
{ $match : {"performance": { $gt: 0.1915} } },
{$sort : {'performance':-1}}
])

//So it's interesting to notice that if we do the same but swicthing 2010 and 2020, we get an
empty list!!!!
```

//This is the same query on the optimized collection! 23 seconds of running less thanthe previous one!!!

```
db.UniqueIndexed.createIndex({'$OP_CARRIER_AIRLINE_ID':1})

db.UniqueIndexed.aggregate([
{ $match: { Year : 2010 }},
{$project:{ 'carrier_Id' :  '$OP_CARRIER_AIRLINE_ID' , 'delayed' : '$DEP_DEL15' }} ,
{ $group: { _id: "$carrier_Id" ,
            count:{$sum:1},
            delayed: {$sum: "$delayed"}}}  ,
{ $project: {'carrier_Id' : '$carrier_Id' , 'performance' : { "$divide": [ "$delayed", "$count" ] } }},
{ $match : {"performance": { $gt: 0.1915} } },
{$sort : {'performance':-1}}
])
```

// This is so fast and far better than the performance we had with the same query on MySQL on //hmk2.

'''Query 3'''
//What is the tail number of the planes that have more averaged flied distance in year 2015 ?

//First version in the worst collection

```
db.fsumm.aggregate([
{ $match: { Year : 2015  }} ,
{$lookup:
{ from:"jan",
localField:"_id",
foreignField:"_id",
as:"details" }},
{ $unwind : "$details" },
{ $group: { _id: "$details.TAIL_NUM" ,  AVG: { $avg: "$DISTANCE"}}} ,
{$sort :{ AVG : -1 }} ,
{ $limit : 10 }
 ])
```

//Not very efficient it runs in 23.848 seconds
//Let's try with our fastest collection UniqueIndexed!

```
db.UniqueIndexed.aggregate([
{ $match: { Year : 2015  }} ,
{ $group: { _id: "$TAIL_NUM" ,  AVG: { $avg: "$DISTANCE"}}} ,
{$sort :{ AVG : -1 }} ,
{ $limit : 10 }
 ])
```

// Wow only 1 second now!!!

'''Query 4'''
// Now we want to start a little investigation on the busiest cities!
''' What are the airports in the USA that have the greatest number of flights to or from them in 2020?'''

// To get the sum of arrival and departure flights, first we get the sum of flights from each origin and save it in
// another collection named origin

```
db.jan.aggregate([
{ $match: { Year : 2020  }}  ,
{ $group: {  "_id":  "$ORIGIN_CITY_NAME", count_origin:{$sum:1} }},
{ $out : 'origin'} ])
```

// Now we group by on jan table this time on dest_city field and calculate the count then make a join between it and origin
// collectoin and then match where dest and origin cities are equal and then get the sum of it

```
db.jan.aggregate([
{ $match: { Year : 2020  }}  ,
{ $group: {  "_id":  "$DEST_CITY_NAME", count_dest:{$sum:1} }},
{ $lookup:
{ from:"origin",
localField:"DEST_CITY_NAME",
foreignField:"ORIGIN_CITY_NAME",
as:"cities" }} ,
{ $unwind : "$cities" },
{$match:{$expr:{$eq:["$_id", "$cities._id"]}}}     ,


{ "$project" :   {'Origin' : '$cities._id',
                 'dest'   : '$_id',
                 'Total' : {'$add' : [ '$count_dest', '$cities.count_origin' ] },
      }
      } ,
{$sort :{ Total : -1 }} ,
{ $limit : 10 }
      ])
```

'''Query 5'''

// So as first let's see what cities are origin of flights that in average have longest flights in
//2020!!!

```
db.UniqueIndexed.createIndex({'ORIGIN_CITY_NAME':1})
db.UniqueIndexed.aggregate([
{ $match: { Year : 2020  }} ,
{ $group: { _id : "$ORIGIN_CITY_NAME" ,  AVG: { $avg: "$DISTANCE"}, sum: {$sum:1}}} ,
{$sort :{ AVG : -1 }} ,
{ $limit : 10 }
 ])
```

//Well as expected we see some far turistic destinations like samoa islands!!!

//But wait a moment! In the first 10 there are also 'Los Angeles, CA' and 'Seattle, WA'!!
//Let's explore a little how much of all the distance covered by flights
// in the usa are actually from those 2 cities, as they have actually big airports
//IN fact in the sum field we see that those two us cities are pretty extraordinary since they have an high average and high number of flights!!
//Could they be some of the most active airports in the usa?

//AND YES!!!! WE found that Los Angeles is actually one of the busiest
//airport + has extraordinary long routes as expected from Silicon valley + pacific flights!

'''Query 6'''
//As we are focusing in Los Angeles, what are the 2 busiest days in this city???

```
db.UniqueIndexed.createIndex({'DEST_CITY_NAME':1})

db.UniqueIndexed.aggregate([
{$match : {$or :[{ ORIGIN_CITY_NAME:'Los Angeles, CA', Year:2020}, {
DEST_CITY_NAME:'Los Angeles, CA',Year:2020}]}},
{$group : {_id : "$DAY_OF_WEEK", SUM : {$sum:1}}},
{$sort : {SUM:-1}},
{$limit:2}
])
```

//So the 2 busiest days are Friday and Thursday!!
//Now that we know this we can ask ourselves, what percentage over the total number of flights in the USA, happen between Friday and Thursday?

'''Query 7'''
//What percentage over the total number of flights in the USA, happen between Friday and
//Thursday?

//First let's take into account the number of flights only on Friday
//and Thursday and divide by the total number of flights in 2020!
//Here we use the operator find

```
db.UniqueIndexed.find(
{$or :[{'DAY_OF_WEEK' : 4, 'Year' : 2020},
   {'DAY_OF_WEEK' : 5, 'Year' : 2020}]}).count() / db.UniqueIndexed.find(
   {'Year':2020}).count()
```

//We just discovered that 34% of the flights in january 2020 happened between thursday and friday,
// if the distribution should be even it should be the 28%, it means there is an increment
//Now we can also ask, even if the biggest airports seems to travel most in those days, in general is the weekend busiest than the weekend?

'''Query 8'''
//Is more busy the couple Friday-Saturday-Sunday, or the remaining days?

```
db.UniqueIndexed.find(
{$or :[{'DAY_OF_WEEK' : 7, 'Year' : 2020},
   {'DAY_OF_WEEK' : 6, 'Year' : 2020},
   {'DAY_OF_WEEK' : 5, 'Year' : 2020}]}).count() / db.UniqueIndexed.find(
   {'Year':2020}).count()
```

// Quick answer: NO!!!!!
//59.92% of flights in the usa happens during the week!
//Still it's pretty surprising that in 3 days there are almost the same quantity of flights than in 4 days!!

'''Query 9'''
//What is the total air time in a month in the usa in 2020??

```
db.UniqueIndexed.aggregate([
{$match: {Year : 2020}},
{$group: {_id: '0', airtime :{$sum : '$AIR_TIME'}}}
,{$project : { _id : '',
total : '$airtime'
}}])
```

// wow 167230341 minutes in a month!!!
// it's in hours

```
67230341/60
//1120505.6 more than 1 million hours!!!
1120505.6/24
46687.73 //The equivalent of 46 thousand days of air traveling time every month in the us!!!!
46687.73/365
//at least 127.9 years of air traveling time every month in the usa!!!!

//By Leonardo Placidi and Negin Amininodoushan
```