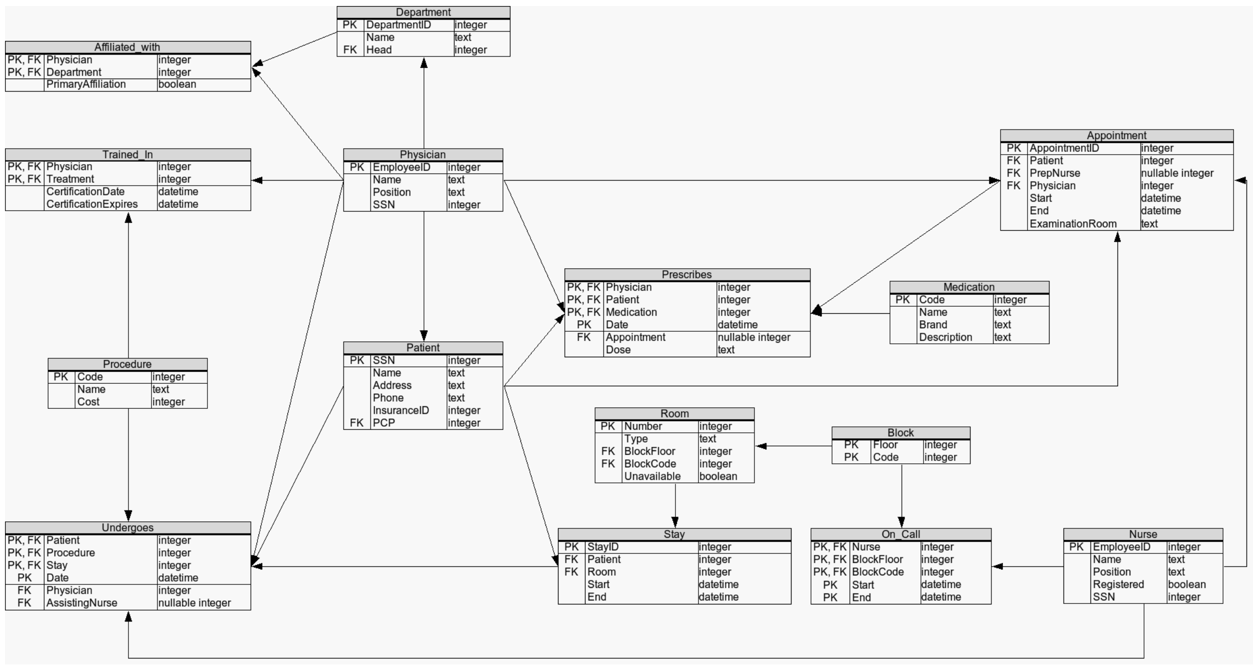
Please indicate below the tool you have analyzed for Homework 3, and past the code, you have realized.
Your code must include all queries and indexes (if any), and possibly the script you have used to populate the database (in case you used a tool for this, e.g., studio 3T for MongoDB, please specify).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

===

Previous MySQL ER-Diagram



Phase I - Migrating MySQL data to MongoDB Database

Step - 1

Using Mongify Tool and Ruby language.

We initially use Mongify to migrate MySQL data to MongoDB as it was before.

In order to Mongify our database, we'll need two things: A database configuration file - Used by Mongify to locate connections to the SQL and MongoDB databases. A translation file - Used to translate your SQL data before saving it into MongoDB.

Configuration File Code

```
sql_connection do
  adapter     "mysql"
  host        "localhost"
  username    "user"
  password    "pass"
  database    "sql_HOSPITAL"
  batch_size  10000
  encoding "utf8"
end

mongodb_connection do
  host        "localhost"
  database    "hospital"
  encoding "utf8"
end
```

In order for Mongify to know what to do with our data, we must provide a translation file: translation.rb

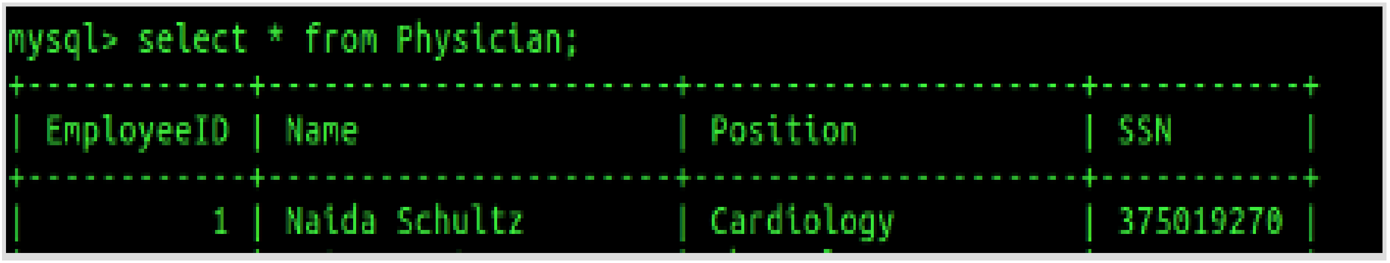
Code

```
table "Affiliated_With" do
    column "Physician", :integer
    column "Department", :integer
    column "PrimaryAffiliation", :boolean
end

table "Appointment" do
    column "AppointmentID", :integer
    column "Patient", :integer
    column "PrepNurse", :integer
    column "Physician", :integer
    column "Start", :datetime
    column "End", :datetime
    column "ExaminationRoom", :text
end

table "Block" do
    column "BlockFloor", :integer
    column "BlockCode", :integer
end

table "Department" do
    column "DepartmentID", :integer
    column "Name", :string
    column "Head", :integer
```

```
end

table "Medication" do
  column "Code", :integer
  column "Name", :string
  column "Brand", :string
  column "Description", :string
end

table "Nurse" do
  column "EmployeeID", :integer
  column "Name", :string
  column "Position", :string
  column "Registered", :boolean
  column "SSN", :integer
end

table "On_Call" do
  column "Nurse", :integer
  column "BlockFloor", :integer
  column "BlockCode", :integer
  column "OnCallStart", :datetime
  column "OnCallEnd", :datetime
end

table "Patient" do
  column "SSN", :integer
  column "Name", :string
  column "Address", :string
  column "Phone", :string
  column "InsuranceID", :string
  column "PCP", :integer
end

table "Physician" do
  column "EmployeeID", :integer
  column "Name", :string
  column "Position", :string
  column "SSN", :integer
end

table "Prescribes" do
  column "Physician", :integer
  column "Patient", :integer
  column "Medication", :integer
  column "Date", :datetime
  column "Appointment", :integer
  column "Dose", :string
end
```

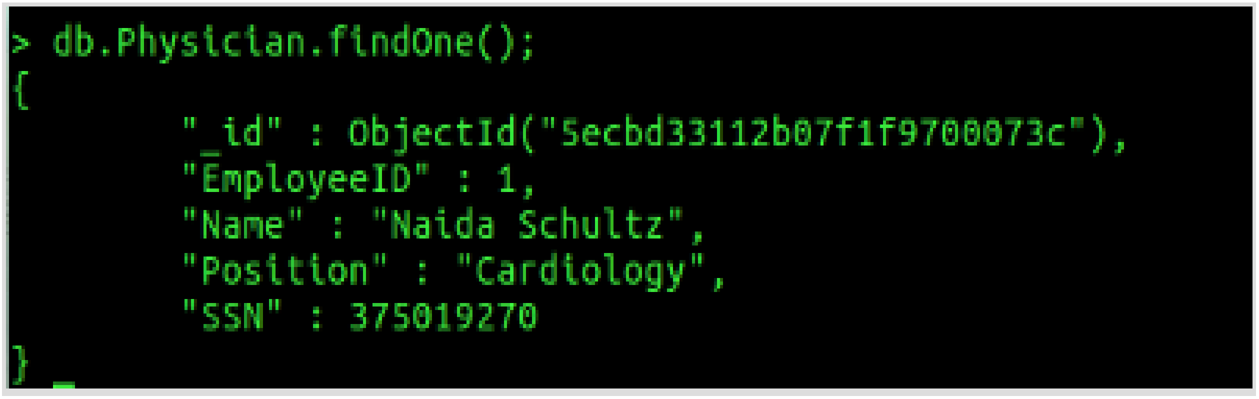
```
table "Procedures" do
  column "Code", :integer
  column "Name", :string
  column "Cost", :float
end

table "Room" do
  column "RoomNumber", :integer
  column "RoomType", :string
  column "BlockFloor", :integer
  column "BlockCode", :integer
  column "Unavailable", :boolean
end

table "Stay" do
  column "StayID", :integer
  column "Patient", :integer
  column "Room", :integer
  column "StayStart", :datetime
  column "StayEnd", :datetime
end

table "Trained_In" do
  column "Physician", :integer
  column "Treatment", :integer
  column "CertificationDate", :datetime
  column "CertificationExpires", :datetime
end

table "Undergoes" do
  column "Patient", :integer
  column "Procedures", :integer
  column "Stay", :integer
  column "DateUndergoes", :datetime
  column "Physician", :integer
  column "AssistingNurse", :integer
end
```

After migration, we will get data in the exact same form



```
mysql> select * from Physician;
+------------+----------------------+-----------------------+-----------+
| EmployeeID | Name                 | Position              | SSN       |
+------------+----------------------+-----------------------+-----------+
|          1 | Naida Schultz        | Cardiology            | 375019270 |
```

Which Is converted to:



```
> db.Physician.findOne();
{
        "_id" : ObjectId("5ecbd33112b07f1f9700073c"),
        "EmployeeID" : 1,
        "Name" : "Naida Schultz",
        "Position" : "Cardiology",
        "SSN" : 375019270
}
```

Phase-II Designing a new MongoDB Schema

Our Proposed Schema Code as below

Procedures:

```
==============
{
        "Code":1,
        "Name":"",
        "Cost":1.1,
}
```

Constraints:

- unique on Code

Notes:

- The procedures are distinct entities.

- So we keep it separate.

Physician:

```
===============
{
	"EmployeeID":1,
	"Name":"",
	"Position":"",
	"SSN":1,
	"Departments":[
		{
			"DepartmentID":1,
			"Name":"",
			"Affiliated_With":{
				"PrimaryAffiliation":true
			}
		}
	],
	"Trained_In":[
		{
			"Treatment":1,  // Procedure code
			"certificationDate":ISODate(""),
			"certificationExpires":ISODate("")
		}
	]
}
```

Constraints:

- unique on EmployeeId

Notes:

- Each department has a head physician and each physician can be head of multiple departments

- Each physician have been trained in multiple procedures with valid or invalid certifications

Nurse:

```
==============
{
	"EmployeeID":1,
	"Name":"",
	"Position":"",
        "Registered":true,
        "SSN":1
}
```

Constraints:
----
unique on EmployeeID

Notes:
- Although the nurse is an employee it has different roles than a Physician.
- Thus we keep it separate.

Medication:

```
----------------
{
        "Code":1,
        "Name":"",
        "Brand":"",
        "Description":""
}
```

Constraints:

- unique on code

Notes:
- Medicine is an individual entity

Patient:

```
==============
{
        "SSN":1,
        "Name":"",
        "Address":"",
        "Phone":"",
        "InsuranceID":"",
        "Appointments":[
                {
                        "AppointmentID":1,
                        "PrepNurse":1, // Nurse's employee ID
                        "Physician":1, // Physician's employee ID
                        "Start":ISODate(""),
                        "End":"ISODate(""),
                "ExaminationRoom":"",
                "Prescribes":[
                    {
                        "Date":ISODate(""),
                        "Code":"",
                        "Dose":""
                    }
                ]
            }
        ],
        "Stays":[
            {
                "StayID":1,
                "StayStart":ISODate(""),
                "StayEnd":ISODate(""),
                "Room":1,  // Room number
                "Undergoes":[
                    {
                        "DateUndergoes":ISODate(""),
                        "Treatment":1, // Procedure code
                        "Physician":1, // Physician's employee id
                        "AssistingNurse":1 // Nurse's employee id
                    }
                ]
            }
        ]
}
```

Constraints:

- unique on SSN

Notes:
- Each patient can have multiple appointments with different physicians
- In each appointment, the physician may give some medications.
- It might be possible that the same physician asks the patient to come back again after some time.
- If the patient is asked to stay at the hospital, he would stay there for some period.
- Whatever treatment is done at the time, will be part of that stay only.
- Each undergoes, has certain procedures that a set of Physicians and Nurses would handle.

Block:

```
=========
{
	"_id":ObjectId(""),  // Mongo generates ID to avoid creating compound primary keys.
	"BlockCode":1,
	"BlockFloor":1
}
```

Notes:
- Block is an individual entity

Room:

```
=========
{
	"RoomNumber":1,
	"RoomType":"",
	"Block":1, // The block ID
	"Unavailable":true
}
```

Constraints:

- unique on number

Notes:
- Room is an individual entity

On_Call:

```
===========
{
	"_id":ObjectId(""), // Mongo generated ID to avoid creating compound primary key
	"Block":1, // The block ID
	"Nurse":1, // The nurse's employee ID
	"OnCallStart":ISODate(""),
	"OnCallEnd":ISODate("")
}
```

Phase-III Migrating Data to the new Schema
To do that, we will use mongo queries itself to do the required aggregation and put the output of aggregation in the parent collection itself. to prepare data for ‘Physician’ collection, we will start aggregation from the current ‘Physician’ collection, lookup into current ‘Department’ and ‘Trained_In’ collections and prepare an output which is how our final Physician collection should look like. That output is forcefully set to replace existing data in the ‘Physician’ collection.

Physician Collection

```
db.Physician.aggregate([
    {
        $lookup:{
            "from":"Department",
            "let":{
                "head":"$EmployeeID"
            },
            "pipeline":[
                {
                    $match:{
                        $expr:{
                            $eq:["$Head","$$head"]
                        }
                    }
                },
                {
                    $lookup:{
                        "from":"Affiliated_With",
                        "let":{
                            "physician":"$Head",
                            "department":"$DepartmentID"
                        },
                        "pipeline":[
                            {
                                $match:{
                                    $expr:{
                                        $and:[
                                            {

$eq:["$Head","$$physician"]
                                            },
                                            {

$eq:["$DepartmentID","$$department"]
                                            }
                                        ]
                                    }
                                }
                            },
                            {
                                $project:{
                                    "_id":0,
                                    "PrimaryAffiliation":1
                                }
                            }
                        ],
                        "as":"Affiliated_With"
                    }
                },
                {
                    $unwind:{
                        "path":"$Affiliated_With",
                        "preserveNullAndEmptyArrays":true
                    }
                },
                {
                    $project:{
                        "_id":0,
                        "DepartmentID":1,
                        "Name":"$Name",
                        "Affiliated_With.PrimaryAffiliation":{

$ifNull:["$Affiliated_With.PrimaryAffiliation",false]
                        }
                    }
                }
            ],
            "as":"Departments"
        }
    },
    {
        $lookup:{
            "from":"Trained_In",
            "let":{
                "physician":"$EmployeeID"
            },
            "pipeline":[
                {
                    $match:{
                        $expr:{
                            $eq:["$Physician","$$physician"]
                        }
                    }
                },
                {
                    $project:{
                        "_id":0,
                        "Physician":0
                    }
                }
            ],
            "as":"Trained_In"
        }
    },
    {
        $out:"Physician"
    }
])
```

On_Call Collection

```
db.On_Call.aggregate([
    {
        $lookup:{
            "from":"Block",
            "let":{
                "blockFloor":"$BlockFloor",
                "blockCode":"$BlockCode"
            },
            "pipeline":[
                {
                    $match:{
                        $expr:{
                            $and:[
                                {
                                    $eq:["$$blockFloor","$BlockFloor"]
                                },
                                {
                                    $eq:["$$blockCode","$BlockCode"]
                                }
                            ]
                        }
                    }
                },
                {
                    $project:{
                        "_id":0,
                        "Block":"$_id"
                    }
                }
            ],
            "as":"blockLookup"
        }
    },
    {
        $unwind:"$blockLookup"
    },
    {
        $project:{
            "_id":1,
            "Nurse" : 1,
            "OnCallStart" : 1,
            "OnCallEnd" : 1,
            "Block" : "$blockLookup.Block"
        }
    },
    {
        $out:"On_Call"
    }

])
```

Room Collection

```
db.Room.aggregate([

    {

        $lookup:{

            "from":"Block",

            "let":{

                "blockFloor":"$BlockFloor",

                "blockCode":"$BlockCode"

            },

            "pipeline":[
                {
                    $match:{
                        $expr:{
                            $and:[
                                {
                                    $eq:["$$blockFloor","$BlockFloor"]
                                },
                                {
                                    $eq:["$$blockCode","$BlockCode"]
                                }
                            ]
                        }
                    }
                },
                {
                    $project:{
                        "_id":0,
                        "Block":"$_id"
                    }
                }
            ],
            "as":"blockLookup"
        }
    },
        {
            $unwind:"$blockLookup"
        },
        {
            $project:{
                "_id":1,
                "RoomNumber" : 1,
                "RoomType" : 1,
                "Block" : "$blockLookup.Block",
                "Unavailable" : 1
            }
        },
        {
            $out:"Room"
        }
    ])
```

Patient Collection

```
db.Patient.aggregate([
    {
        $lookup:{
            "from":"Appointment",
            "let":{
                "patient":"$SSN"
            },
"pipeline":[
    {
        $match:{
            $expr:{
                $eq:["$$patient","$Patient"]
            }
        }
    },
    {
        $project:{
            "_id":0,
            "Patient":0
        }
    },
    {
        $lookup:{
            "from":"Prescribes",
            "let":{
                "appointmentId":"$AppointmentID"
            },
            "pipeline":[
                {
                    $match:{
                        $expr:{
```

```
$eq:["$Appointment","$$appointmentId"]
                                    }
                                }
                            },
                            {
                                $project:{
                                    "_id":0,
                                    "Medication":1,
                                    "Date":1,
                                    "Dose":1
                                }
                            }
                        ],
                        "as":"Prescribes"
                    }
                }
            ],
            "as":"Appointments"
        }
    },
    {
        $lookup:{
            "from":"Stay",
"let":{
    "patient":"$SSN"
},
"pipeline":[
    {
        $match:{
            $expr:{
                $eq:["$Patient","$$patient"]
            }
        }
    },
    {
        $lookup:{
            "from":"Undergoes",
            "let":{
                "stay":"$StayID"
            },
            "pipeline":[
                {
                    $match:{
                        $expr:{
                            $eq:["$Stay","$$stay"]
                        }
                    }
```

```
                        },
                        {
                            $project:{
                                "_id":0,
                                "Patient":0,
                                "Stay":0
                            }
                        }
                    ],
                    "as":"Undergoes"
                }
            },
            {
                $project:{
                    "_id":0
                }
            }
        ],
        "as":"Stays"
    }
},
{
    $out:"Patient"
}
```

```
])
```

Phase IV - Queries (From 1 Line to More than 100 Lines)

// 1. All the distinct positions for Nurses

```
db.Nurse.distinct("Position");
```

// 2. All the details of patients who stayed for two times

```
db.Patient.find({"Stays.1":{$exists:true}}).pretty();
```

// 3. Costs of procedures in desc

```
db.Procedures.aggregate([
 {
  $sort:{
   "Cost":-1
  }
 },
 {
  $limit:5
 }
]).pretty()
```

// 4. Physicians and department relations with certification start and end dates

```
db.Physician.aggregate([
 {
  $project:{
   "Name":1,
   "Position":1,
   "Departments":1,
   "Certifications":"$Trained_In",
   "_id":0
  }
 }
]).pretty()
```

// 5. Calculating The Availability Of Rooms

```
db.Room.aggregate([
 {
   $group:{
     "_id":{
       $cond:[
         "$Unavailable",
         "Unavailable",
         "Available"
       ]
     },
     "count":{
       $sum:1
     }
   }
 }
]).pretty();
```

// 6. Getting Brands Of Medicines and calculate the total Availability of that Medicine.

```
db.Medication.aggregate([
 {
   $group:{
     "_id":"$Brand",
     "Brand":{
       $first:"$Brand"
     },
     "count":{
       $sum:1
     }
   }
 },
 {
   $project:{
     "_id":0
   }
 }
]).pretty()
```

```
// 7. rooms in each block in each floor
// We need to look up into Room collection and fetch all Rooms by Block object ID.
// After lookup, we get an array of Rooms for each Block code and Block floor
// Finally to get the number of rooms, we will calculate the size of the ‘Rooms' array using $size operator

db.Block.aggregate([
   {
      $lookup: {
         "from": "Room",
         "localField": "_id",
         "foreignField": "Block",
         "as": "Rooms"
      }
   },
   {
      $project: {
         "_id": 0,
         "BlockCode": 1,
         "BlockFloor": 1,
         "Rooms": {
            $size: "$Rooms"
         }
      }
   },
   {
      $sort: {
         "BlockCode": 1,
         "BlockFloor": 1
      }
   }
]).pretty()
```

// 8. PATIENTS THAT HAD APPOINTMENTS WHICH PROCEEDED TO AN UNDERGOE

```
db.Patient.aggregate([
 {
   $match:{
     $expr:{
       $and:[
         {
           $ne:[
             {
               $size:"$Appointments"
             },
             0
           ]
         },
         {
           $ne:[
             {
               $size:"$Stays.Undergoes"
             },
             0
           ]
         }
       ]
     }
   }
 },
 {
   $project:{
     "_id":0,
     "Stays":0,
     "Appointments":0
   }
 }
]).pretty();
```

```
// 9. Patients who got APPOINTMENTS, but had NOT been PRESCRIBED

db.Patient.aggregate([
 {
   $match:{
     $expr:{
       $gt:[
         {
           $size:{
             $filter:{
               "input":"$Appointments",
               "as":"appointment",
               "cond":{
                 $eq:[
                   {
                     $size:"$$appointment.Prescribes"
                   },
                   0
                 ]
               }
             }
           }
         },
         0
       ]
     }
   }
 },
 {
   $project:{
     "_id":0,
     "Stays":0,
     "Appointments":0
   }
 }
]).pretty()
```

// 10. Patients that STAYED but did NOT UNDERGO any procedure between two dates

```
db.Patient.aggregate([
 {
  $unwind:"$Stays"
 },
 {
  $match:{
    $expr:{
      $eq:[
        {
          $size:{
            $filter:{
              "input":"$Stays.Undergoes",
              "as":"undergo",
              "cond":{
                $and:[
                  {
                    $gte:["$$undergo.DateUndergoes",ISODate("2020-04-20T00:00:00Z")]
                  },
                  {
                    $lte:["$$undergo.DateUndergoes",ISODate("2020-04-25T00:00:00Z")]
                  }
                ]
              }
            }
          }
        },
        0
      ]
    }
  }
 },
 {
  $project:{
    "_id":0,
    "SSN" : 1,
    "Name" : 1,
    "Address" : 1,
    "Phone" : 1,
    "InsuranceID" : 1,
    "PCP" : 1,
    "StayStart":"$Stays.StayStart",
    "StayEnd":"$Stays.StayEnd"
    }
  }
]).pretty()
```

// 11. Find out details of prescribed medicines for patients between the 1st to 5th of April.

```
db.Patient.aggregate([
  {
    $unwind:"$Appointments"
  },
  {
    $unwind:"$Appointments.Prescribes"
  },
  {
    $match:{
      "Appointments.Prescribes.Date":{
        $gte:ISODate("2020-04-20T00:00:00Z"),
        $lte:ISODate("2020-04-25T00:00:00Z")
      }
    }
  },
  {
    $lookup:{
      "from":"Medication",
      "let":{
        "code":"$Appointments.Prescribes.Medication"
      },
      "pipeline":[
        {
          $match:{
            $expr:{
              $eq:["$Code","$$code"]
            }
          }
        },

        {
          $project:{
            "_id":0,
            "Name":1
          }
        }
      ],
      "as":"MedicationLookup"
    }
  },
  {
    $unwind:"$MedicationLookup"
  },
  {
    $group:{
      "_id":"$SSN",
      "SSN":{
        $first:"$SSN"
      },
      "Name":{
        $first:"$Name"
      },
      "Address":{
        $first:"$Address"
      },
      "Phone":{
        $first:"$Phone"
      },
      "InsuranceID":{
        $first:"$InsuranceID"
      },
      "PCP":{
        $first:"$PCP"
      },
      "Medications":{
        $push:{
          "Dose":"$Appointments.Prescribes.Dose",
          "Name":"$MedicationLookup.Name"
        }
      }

    }
  },
  {
    $project:{
      "_id":0
    }
  }
]).pretty()
```

// 12. Details of patients undergoing the particular procedure on "2020-04-22" with doctor's name and procedure name.

```
db.Patient.aggregate([
 {
  $unwind:"$Stays"
 },
 {
  $unwind:"$Stays.Undergoes"
 },
 {
  $match:{
   "Stays.Undergoes.DateUndergoes":{
    $gte:ISODate("2020-04-22T00:00:00Z"),
    $lt:ISODate("2020-04-23T00:00:00Z")
   },
   "Stays.Undergoes.Treatment" : 8
  }
 },
 {
  $lookup:{
   "from":"Physician",
   "let":{
    "physician":"$Stays.Undergoes.Physician"
   },
   "pipeline":[
    {
     $match:{
      $expr:{
       $eq:[
        "$EmployeeID",
        "$$physician"
       ]
      }
     }
    },
    {
     $project:{
      "_id":0,
      "Name":1
```

```
        }
      }
    ],
    "as":"PhysicianLookup"
  }
},
{
  $unwind:{
    "path":"$PhysicianLookup",
    "preserveNullAndEmptyArrays":true
  }
},
{
  $lookup:{
    "from":"Procedures",
    "let":{
      "code":"$Stays.Undergoes.Treatment"
    },
    "pipeline":[
      {
        $match:{
          $expr:{
            $eq:[
              "$Code",
              "$$code"
            ]
          }
        }
      },
      {
        $project:{
          "_id":0,
          "Name":1
        }
      }
    ],
    "as":"ProceduresLookup"
  }
},
{
  $unwind:{
    "path":"$ProceduresLookup",
    "preserveNullAndEmptyArrays":true
  }
},
{
  $project:{
    "_id":0,
    "SSN" : 1,
    "Name" : 1,
      "Address" : 1,
      "Phone" : 1,
      "InsuranceID" :1,
      "PCP" : 1,
      "PhysicianName":{
        $ifNull:["$PhysicianLookup.Name","NA"]
      },
      "ProcedureName":{
        $ifNull:["$ProceduresLookup.Name","NA"]
      }
    }
  }
]).pretty()
```