USE CASE STUDY REPORT

Group No.: Group 23

Student Names: Aryak Bodkhe and Shreyans Thesia

I. Introduction

Business Problem:

Many overseas students pursue their dreams of attending university in the United States each year. Due to universities requiring students to have approved health insurance, students must obtain decent health insurance that covers treatment for most common diseases and injuries, where the student pays less, and the insurance company covers most of the treatment costs. Students frequently lack access to resources where they can view and compare the best-priced plans that are necessary for them. In order to address this issue, we are going to develop a health insurance database system where we can supply all required information about all insurance plans, and which would be the best option depending on student desire.

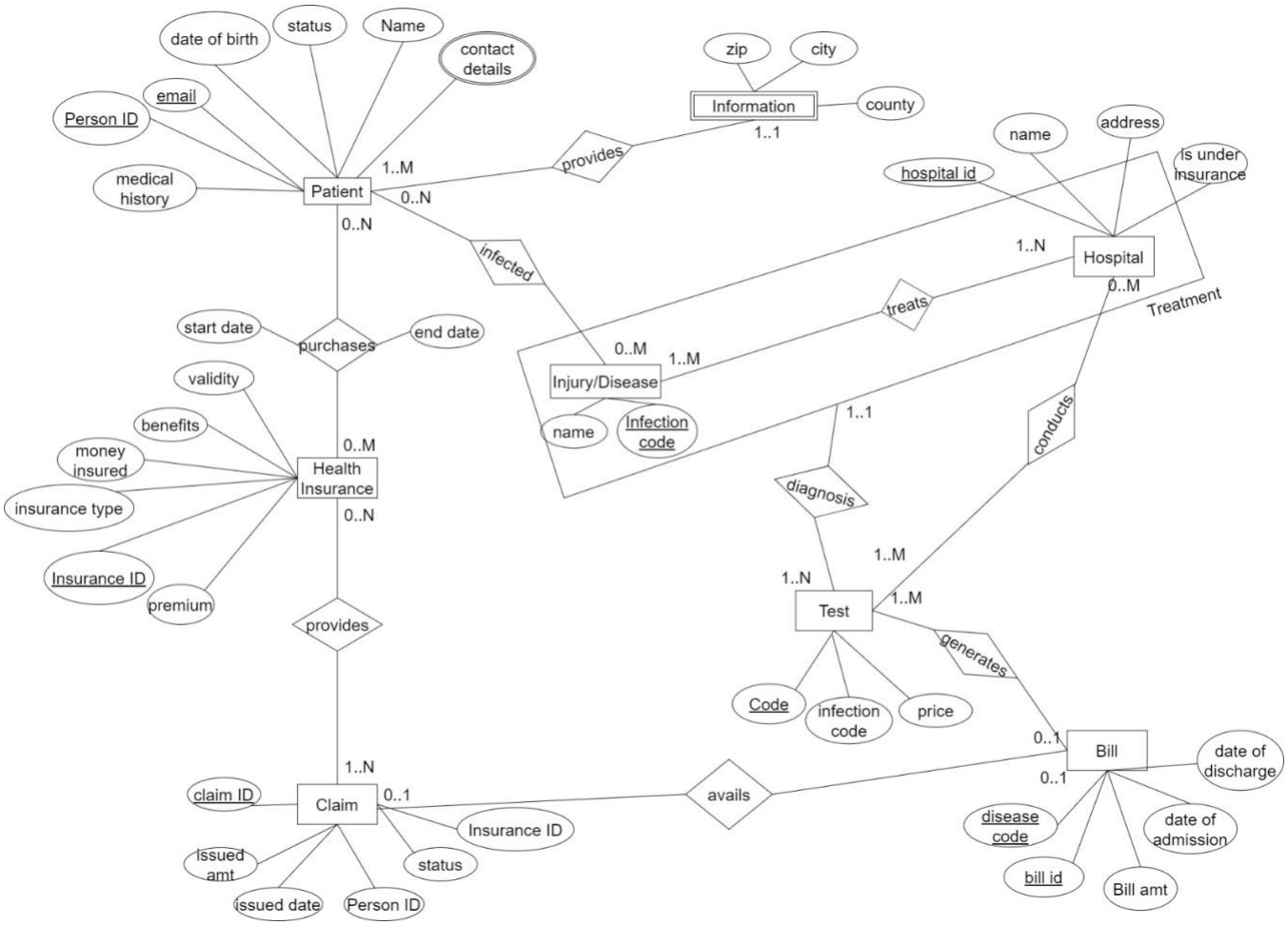
Our DBMS will provide access to various health insurance plans and affiliated hospitals and the cost of treatment for various diseases and injuries so that students can find the appropriate health plan in one place in their budget.

In the model if a person- student or a foreign visitor who comes to an accident and does not have any insurance, but he is a patient as he/she is injured.

We can claim the health insurance multiple times and a person can have multiple insurances such as generic health insurance, dental insurance, eye insurance etc.

II. Conceptual Data Modeling

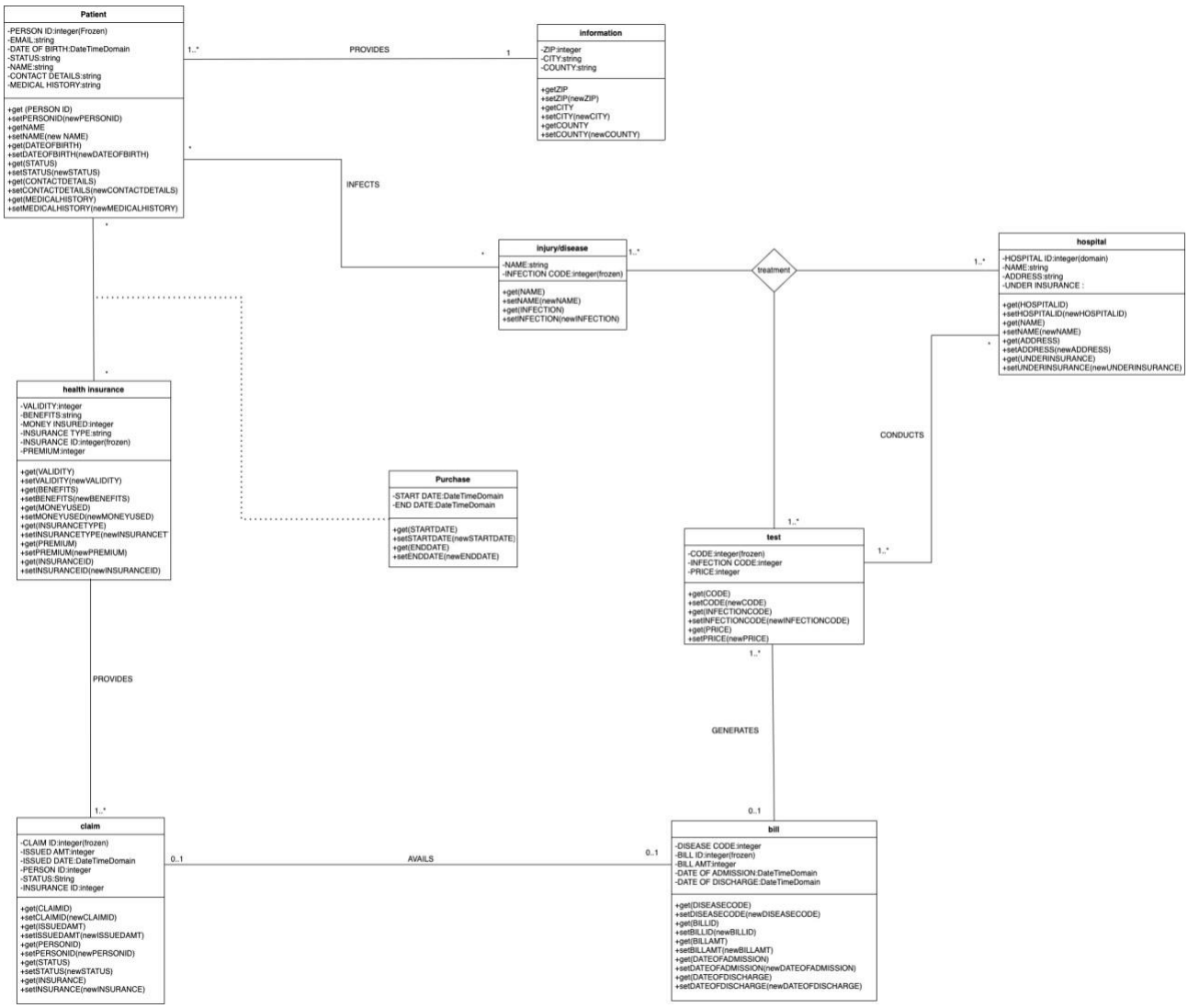
EER Diagram:



Here we have reference data in injury/disease entity where we have stored information about tests required for any disease in the database.

Here when health insurances are sold, the transaction referenced to the relevant Insurance ID and Patient ID. The insurance and patient records do not need to be modified for the new transactions.

UML Diagram:


Patient
-PERSON ID:integer(Frozen)
-EMAIL:string
-DATE OF BIRTH:DateTimeDomain
-STATUS:string
-NAME:string
-CONTACT DETAILS:string
-MEDICAL HISTORY:string
+get (PERSON ID)
+setPERSONID(newPERSONID)
+getNAME
+setNAME(new NAME)
+get(DATEOFBIRTH)
+setDATEOFBIRTH(newDATEOFBIRTH)
+get(STATUS)
+setSTATUS(newSTATUS)
+get(CONTACTDETAILS)
+setCONTACTDETAILS(newCONTACTDETAILS)
+get(MEDICALHISTORY)
+setMEDICALHISTORY(newMEDICALHISTORY)
1..*
PROVIDES
1
information
-ZIP:integer
-CITY:string
-COUNTY:string
+getZIP
+setZIP(newZIP)
+getCITY
+setCITY(newCITY)
+getCOUNTY
+setCOUNTY(newCOUNTY)
*
INFECTS
*
injury/disease
-NAME:string
-INFECTION CODE:integer(frozen)
+get(NAME)
+setNAME(newNAME)
+get(INFECTION)
+setINFECTION(newINFECTION)
1..*
treatment
1..*
hospital
-HOSPITAL ID:integer(domain)
-NAME:string
-ADDRESS:string
-UNDER INSURANCE :
+get(HOSPITALID)
+setHOSPITALID(newHOSPITALID)
+get(NAME)
+setNAME(newNAME)
+get(ADDRESS)
+setADDRESS(newADDRESS)
+get(UNDERINSURANCE)
+setUNDERINSURANCE(newUNDERINSURANCE)
*
CONDUCTS
health insurance
-VALIDITY:integer
-BENEFITS:string
-MONEY INSURED:integer
-INSURANCE TYPE:string
-INSURANCE ID:integer(frozen)
-PREMIUM:integer
+get(VALIDITY)
+setVALIDITY(newVALIDITY)
+get(BENEFITS)
+setBENEFITS(newBENEFITS)
+get(MONEYUSED)
+setMONEYUSED(newMONEYUSED)
+get(INSURANCETYPE)
+setINSURANCETYPE(newINSURANCET
+get(PREMIUM)
+setPREMIUM(newPREMIUM)
+get(INSURANCEID)
+setINSURANCEID(newINSURANCEID)
Purchase
-START DATE:DateTimeDomain
-END DATE:DateTimeDomain
+get(STARTDATE)
+setSTARTDATE(newSTARTDATE)
+get(ENDDATE)
+setENDDATE(newENDDATE)
1..*
test
-CODE:integer(frozen)
-INFECTION CODE:integer
-PRICE:integer
+get(CODE)
+setCODE(newCODE)
+get(INFECTIONCODE)
+setINFECTIONCODE(newINFECTIONCODE)
+get(PRICE)
+setPRICE(newPRICE)
1..*
1..*
PROVIDES
GENERATES
1..*
0..1
claim
-CLAIM ID:integer(frozen)
-ISSUED AMT:integer
-ISSUED DATE:DateTimeDomain
-PERSON ID:integer
-STATUS:String
-INSURANCE ID:integer
+get(CLAIMID)
+setCLAIMID(newCLAIMID)
+get(ISSUEDAMT)
+setISSUEDAMT(newISSUEDAMT)
+get(PERSONID)
+setPERSONID(newPERSONID)
+get(STATUS)
+setSTATUS(newSTATUS)
+get(INSURANCE)
+setINSURANCE(newINSURANCE)
0..1
AVAILS
0..1
bill
-DISEASE CODE:integer
-BILL ID:integer(frozen)
-BILL AMT:integer
-DATE OF ADMISSION:DateTimeDomain
-DATE OF DISCHARGE:DateTimeDomain
+get(DISEASECODE)
+setDISEASECODE(newDISEASECODE)
+get(BILLID)
+setBILLID(newBILLID)
+get(BILLAMT)
+setBILLAMT(newBILLAMT)
+get(DATEOFADMISSION)
+setDATEOFADMISSION(newDATEOFADMISSION)
+get(DATEOFDISCHARGE)
+setDATEOFDISCHARGE(newDATEOFDISCHARGE)

III. Mapping Conceptual Model to Relational Model

Relational Mapping :

Primary key (underlined), foreign key (in italics)

- Patient(PersonID, *info_id*, medicalhistory, email, dateofbirth, status, Name, contactdetails)
    - Info_id

Here, Info_id is the foreign key which refers to Information and it is NULL NOT ALLOWED

- Purchases(*PersonID, InsuranceID*, startdate,enddate)
    - PersonID, InsuranceID

Here they are the foreign keys that refer to Patient and Health insurance respectively and it is NULL NOT ALLOWED

- Infected(*PersonID,Infectioncode*)
    - personID, Infectioncode

Here these are the foreign keys that refer to Patient and injury respectively and it is NULL NOT ALLOWED

- Information (zip, city, county)

- Health insurance (insuranceID, insurancetype,moneyinsured,benefits,validity,premium)

- Provides(*insuranceid* , *claimid*)
    - insuranceid, claimid

Here these are the foreign keys that refer to Health insurance and claim respectively and it is NULL NOT ALLOWED

- Claim(claimID, *billid* , issuedamt, issueddate, status, *personID, InsuranceID*)
    - personID, InsuranceID

Here they are the foreign keys that refer to Patient and Health insurance respectively and it is NULL NOT ALLOWED

- billid

this is the foreign key that refers to BILL and is NULL ALLOWED

- Bill(billid , dateofdischarge, dateofadmission, billamt, *injurycode*)
- injurycode

Here it is the foreign key refering to injury and it is NULL NOT ALLOWED

- Injury(<u>infectioncode</u>, name)

- Treats(*infectioncode, hospitalid*)

    - infectioncode, hospitalid

Here it is the foreign key that refers to injury and hospital and it is NULL NOT ALLOWED

- Hospital(<u>hospitalid,</u> hname, haddress, isunderinsurance)

- Conducts(*hospitalid,testcode*)

    - hospitalid, testcode

Here it is the foreign key that refers to Hospital and Test and it is NULL NOT ALLOWED

- Test(<u>code</u>,*billid*, price)

- Billid

Here it is the foreign key that refers to bill and it is NULL NOT ALLOWED

- Treatment(*infectioncode,hospitalid,code*)
    - hospitalid, testcode, code

Here it is a foreign key that refers to hospital, test and injury and it is NULL NOT ALLOWED.

Normalization :

- Patient(<u>PersonID</u>, *info_id,* medicalhistory, email, dateofbirth, status, Name, contactdetails)
    - Already in 1NF and 2NF
    - 3NF and BCNF: Patient(Name,contactdetails)

P_info(<u>PersonID</u>, *info_id*,medicalhistory,email,dateofbirth,status,Name)

- Purchases(*PersonID, InsuranceID,* startdate,enddate)
    - ALREADY IN 1NF and 2NF
    - 3NF and BCNF:

Purchases*(PersonID,* startdate, enddate) R(*PersonID, InsuranceID*)

- Infected(*PersonID,<u>Infectioncode</u>*)
    - Already normalized

- Information (zip, city, county)
    - Already normalized

- Health insurance (<u>insuranceID,</u> insurancetype,moneyinsured,benefits,validity,premium)

- Already in 1NF and 2NF
- 3NF and BCNF:

Healthinsurance (insurancetype,benefits,premium,moneyinsured,validity) Type (insuranceID,insurancetype)

- Provides(*insuranceid* , *claimid*)
  - Already normalized

- Claim(claimID, *billid* , issuedamt, issueddate, status, *personID, InsuranceID*)
  - Satisfies 1NF and 2NF
  - 3NF and BCNF: Insure(InsuranceID,claimID) Claim_details(claimID,status,billid) Bill_details(billid,issueddate,issuedamt) Person(*personID,InsuranceID*)

- Bill(billid , dateofdischarge, dateofadmission, billamt, *injurycode*)
  - ALREADY in 1NF, 2NF, 3NF, BCNF
  - Injury(infectioncode, name)
  - ALREADY in 1NF, 2NF, 3NF, BCNF

- Treats(*infectioncode, hospitalid*)
  - ALREADY in 1NF, 2NF, 3NF, BCNF

- Hospital(hospitalid, hname, haddress, isunderinsurance)
  - ALREADY in 1NF, 2NF, 3NF, BCNF

- Conducts(*hospitalid,testcode*)
  - ALREADY in 1NF, 2NF, 3NF, BCNF

- Test(code,*billid*, price)
  - ALREADY in 1NF, 2NF, 3NF, BCNF

- Treatment(*infectioncode,hospitalid,code*)
  ALREADY in 1NF, 2NF, 3NF, BCNF

IV. Implementation of Relation Model via MySQL and NoSQL

Implementing Analytical SQL Queries

Query1:

```
SELECT
    d.money_insured as MONEY_INSURED,
    d.insurance_type AS INSURANCE_TYPE,
    a.insurance_id AS INSURANCE_ID,
    b.status AS STATUS,
    f.P_Name as PatientName
FROM
    insure a,
    claim_details b,
    insurance_type c,
    health_insurance d,
    person e,
    patient_info f
WHERE
    a.claim_id = b.claim_id
    AND b.status = 'TRUE'
    AND a.insurance_id = c.insurance_id
    AND c.insurance_type = d.insurance_type
    and e.person_insurance_id=c.insurance_id
    and e.personID=f.Person_ID
LIMIT 10;
```

MONEY_INSURED	INSURANCE_TYPE	INSURANCE_ID	STATUS	PatientName
92254	PPO	6	TRUE	Erina Wye
92254	PPO	149	TRUE	Krystyna Quirk
74421	POS	141	TRUE	Barbara-anne Parbrook
74421	POS	195	TRUE	Chrisy Hessentaler
72931	EPO	131	TRUE	Sherman Joint
87407	HMO	184	TRUE	Tabb Borleace
74421	POS	167	TRUE	Nickey Pache
87407	HMO	41	TRUE	Noll Sayton
92254	PPO	108	TRUE	Layne Hagergham
92254	PPO	25	TRUE	Ferris Trustrie

This query will output the patient name whose insurance has been passed alongwith their insurance ID and which insurance type they have bought and the amount of money which is insurd to them.

Query 2:

```
SELECT
    a.bill_amt AS Amount,
    b.injury_Name AS Injury,
    b.infection_code AS Infection_Code
FROM
    bill a,
    injury b
WHERE
    b.infection_code = a.injury_code
GROUP BY b.injury_Name
ORDER BY a.bill_amt DESC
LIMIT 10;
```

Amount	Injury	Infection_Code
99501	Poisoning by unsp topical agent, undetermined,…	20
99329	Fracture of mandible, unsp, subs for fx w routn…	150
97940	Displ commnt fx r patella, 7thR	21
97683	Cholera, unspecified	100
97157	Corrosion of second degree of left lower leg, init…	75
96573	Malignant neoplasm of appendix	187
96498	Nondisp commnt fx shaft of unsp femr, 7thE	86
96455	Allergy status to unsp drug/meds/biol subst status	184
96284	Non-pressure chronic ulcer of unspecified ankle	6

This query outputs the injuries and the infection code which has the highest billamount to treat that disease or injury

Analytical NoSQL Queries:

Query to find the average of the entire bill amount of treating various injuries

```
db.getCollection("bill").find({})

db.bill.aggregate([
    {
        $group : {
            _id: "aggregates",
            average_insured: {
                $avg: "$bill_amt"
            },
        }
    }
])
```

```
{
    "_id" : "aggregates",
    "average_insured" : 54632.075
}
```

Count of patients who were insured.

```
db.getCollection("patient_info").find({})

db.patient_info.aggregate(
  [
    {
      $match: {
        Status: {
          $eq: "true"
        }
      }
    },
    {
      $count: "Patients who had Insurance"
    }
  ]
)
```

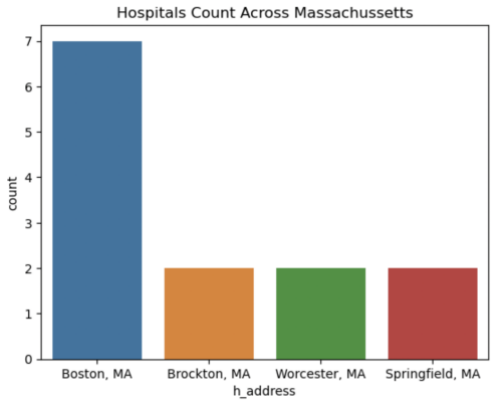
```
{
    "Active_Patient_Count" : NumberInt(90)
}
```

V. Database Access via R or Python

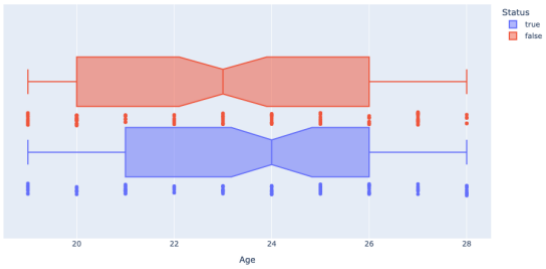
The database is accessed using Python's library such as Pandas library and using Matplotlib to plot the graphs for the analytics.

Analytics Using python:

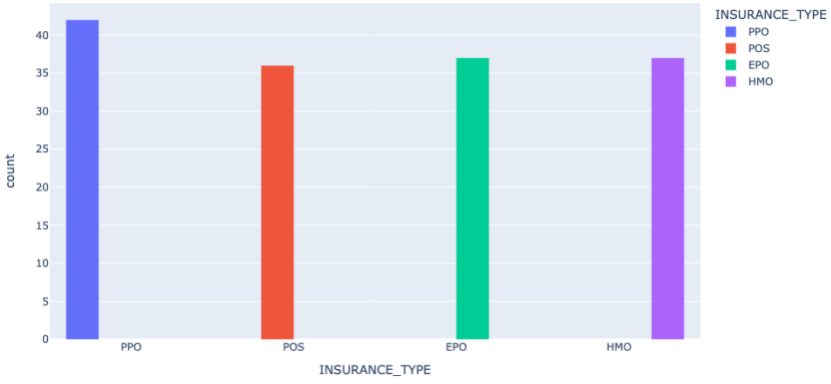
1. The number of hospitals that are under insurance of PPO type which has the most number of tests being performed at the hospital.



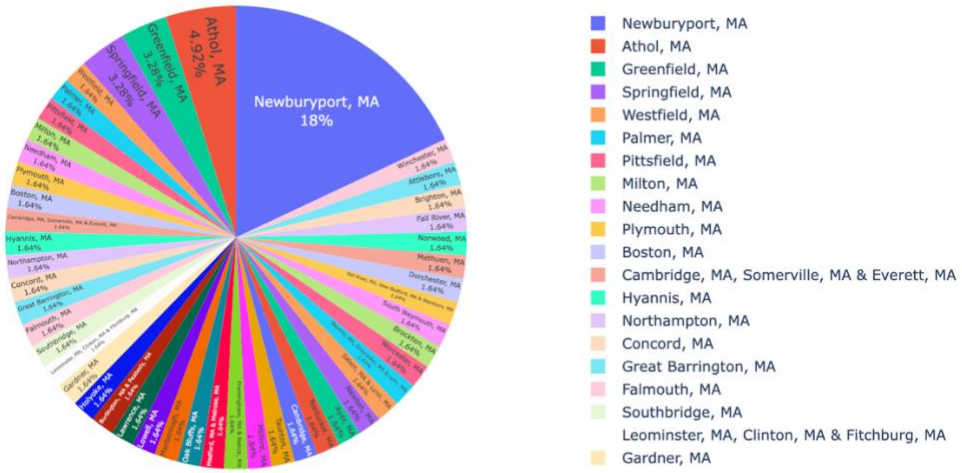
2. Box plot distribution of age of patients vs status of whether they have active insurance plans



3. Count of Insurance Plans that have been successfully claimed at least one time



4. Pie chart of no of hospitals in Massachusetts that are under insurance depending on the percentage



VII. Summary and recommendation

Project Conclusion:

-We have conveniently administered the problem statement, conceptual Modeling using EER & UML Diagram in SQL database as well as NoSQL database.

-Also, we have successfully implemented the database access via python & demonstrated practical visualization analytics.