Health Insurance Database System Management

Professor: Xuemin Jin

Group Member

Urjasvit Sinha

sinha.u@husky.neu.edu

Chia-Han Chiang

chiang.ch@husky.neu.edu

01 Problem Definition

02 Requirement

03 Data Modeling

04 Implementation

01

Problem Definition

DATABASE

Manage the insertion, storage and efficient retrieval of the data of the insurance company

IMPLEMENT

- Listing the required information
- Assumed a health insurance product

PURPOSE

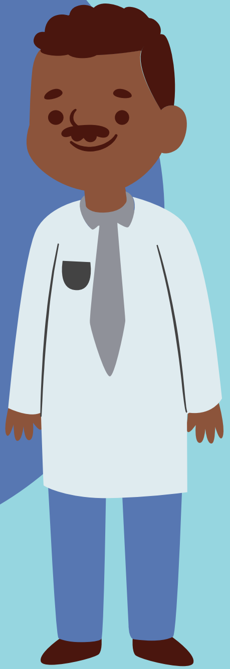
- Maintain data of multiple entities across the enterprise: restrict the accessibility of the resources
- Provide permissions to the users: the restriction they met

ASSUME

- **Company:** Liberty Life
- **Main Product:** Health insurance
- **Demand:** solve data management issues and process management of their policyholders easily

Requirement

02



Data Modeling

Diagram

Enhanced Entity Relation
Unified Modeling Language
Relational Model

MySQL

MySQL Workbench

Various real world situations
Using query to retrieve

NoSQL

MongoDB

MapReduce pipeline
Aggregate pipeline

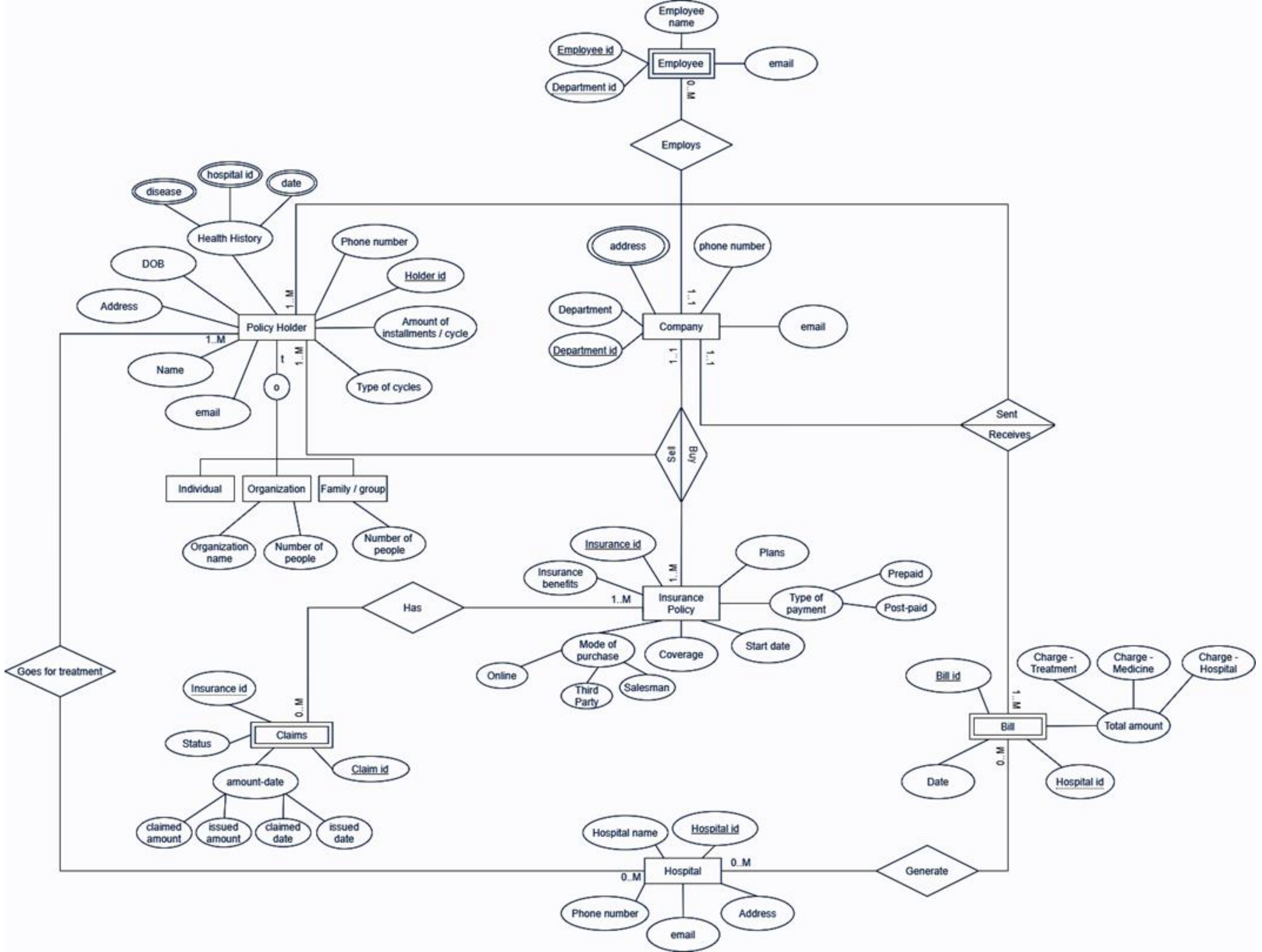
Analysis

RStudio

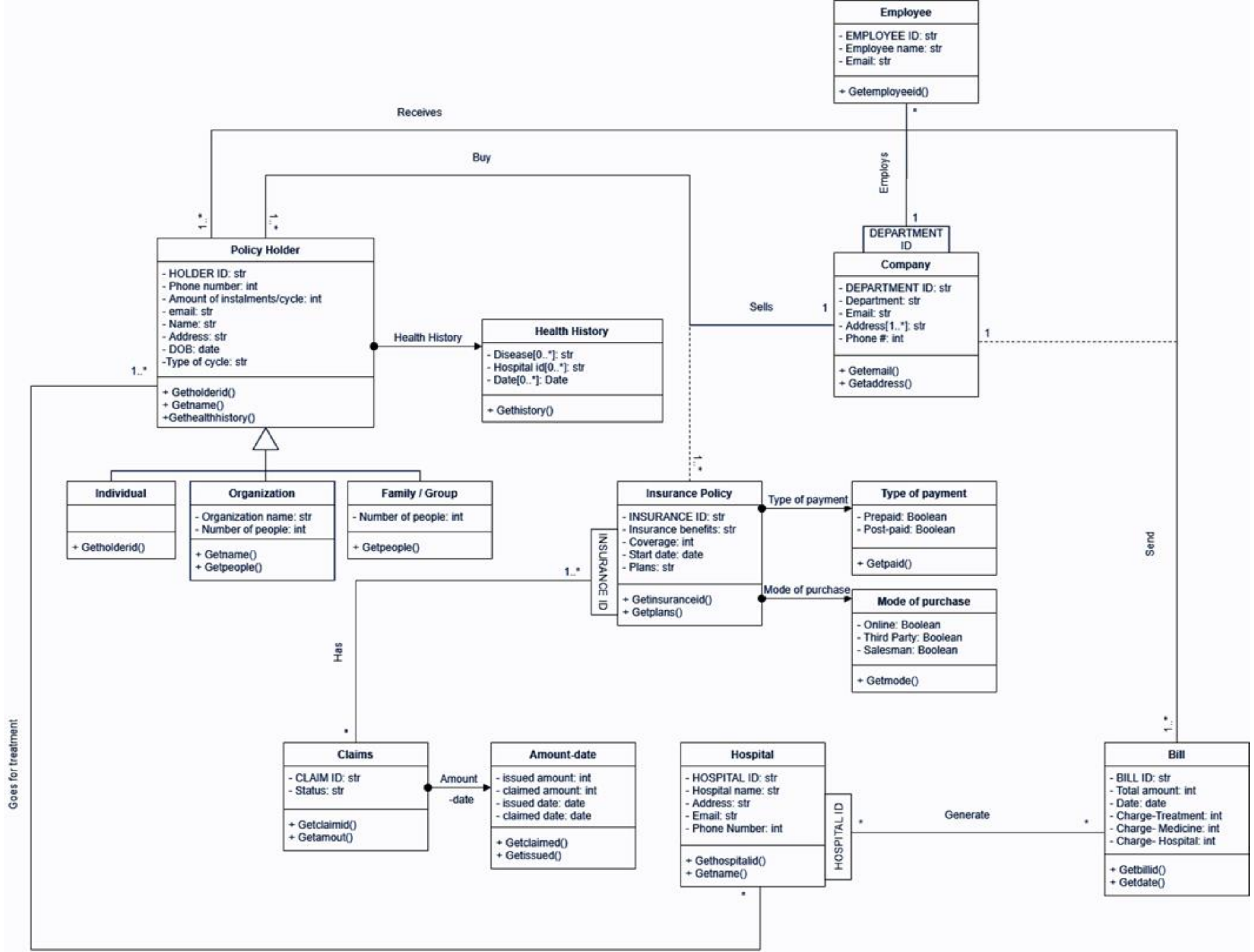
Library: RMySQL, dbConnect
Compare details between
policyholders and products

Data Modeling

EER


Employee name
Employee id
Employee
email
Department id
0..M
Employs
disease
hospital id
date
Health History
Phone number
DOB
Holder id
Address
Amount of installments / cycle
Policy Holder
1..M
1..M
1..M
t
Name
email
o
Type of cycles
Individual
Organization
Family / group
Organization name
Number of people
Number of people
address
phone number
Department
Company
email
Department id
1..1
1..1
1..1
Sent
Receives
Sell
Buy
Insurance id
Plans
Insurance benefits
1..M
Prepaid
Has
1..M
Insurance Policy
Type of payment
Post-paid
Mode of purchase
Coverage
Start date
Online
Third Party
Salesman
Goes for treatment
Insurance id
0..M
Claims
Status
Claim id
amount-date
claimed amount
issued amount
claimed date
issued date
Bill id
Charge - Treatment
Charge - Medicine
Charge - Hospital
1..M
Bill
Total amount
0..M
Date
Hospital id
Hospital name
Hospital id
0..M
Hospital
0..M
Generate
Phone number
email
Address

UML


Employee
- EMPLOYEE ID: str
- Employee name: str
- Email: str
+ Getemployeeid()
Receives
Buy
Employs
*
1
DEPARTMENT ID
Company
- DEPARTMENT ID: str
- Department: str
- Email: str
- Address[1..*]: str
- Phone #: int
+ Getemail()
+ Getaddress()
Sells
1
1
1..*
1..*
Policy Holder
- HOLDER ID: str
- Phone number: int
- Amount of instalments/cycle: int
- email: str
- Name: str
- Address: str
- DOB: date
-Type of cycle: str
+ Getholderid()
+ Getname()
+Gethealthhistory()
Health History
Health History
- Disease[0..*]: str
- Hospital id[0..*]: str
- Date[0..*]: Date
+ Gethistory()
1..*
Individual
+ Getholderid()
Organization
- Organization name: str
- Number of people: int
+ Getname()
+ Getpeople()
Family / Group
- Number of people: int
+ Getpeople()
1..*
Insurance Policy
- INSURANCE ID: str
- Insurance benefits: str
- Coverage: int
- Start date: date
- Plans: str
+ Getinsuranceid()
+ Getplans()
INSURANCE ID
1..*
Type of payment
Type of payment
- Prepaid: Boolean
- Post-paid: Boolean
+ Getpaid()
Mode of purchase
Mode of purchase
- Online: Boolean
- Third Party: Boolean
- Salesman: Boolean
+ Getmode()
Send
Has
*
Goes for treatment
Claims
- CLAIM ID: str
- Status: str
+ Getclaimid()
+ Getamout()
Amount
-date
Amount-date
- issued amount: int
- claimed amount: int
- issued date: date
- claimed date: date
+ Getclaimed()
+ Getissued()
Hospital
- HOSPITAL ID: str
- Hospital name: str
- Address: str
- Email: str
- Phone Number: int
+ Gethospitalid()
+ Getname()
HOSPITAL ID
*
*
Generate
*
1..*
Bill
- BILL ID: str
- Total amount: int
- Date: date
- Charge-Treatment: int
- Charge- Medicine: int
- Charge- Hospital: int
+ Getbillid()
+ Getdate()

Relational Model

BOLD(Primary) and ***BOLD***(Foreign key) is NOT NULL

POLICY HOLDER: { **Holder id** | Phone number | DOB | Address | Name | Email | Insurance id | Bill id | Hospital id | Department id}
POLICYHOLDER-CYCLE: { **Holder id** | Amount of installment / cycle | Type of Cycle }
HOLDER_HEALTH_HISTORY: { **Holder id** | Disease | hospital id | date }
INDIVIDUAL: { **Holder id** }
ORGANIZATION: { **Holder id** l Organization name | Number of people }
FAMILY / GROUP: { **Holder id** | Number of people }

EMPLOYEE: { **Employee id** | ***Department id***| Employee name | Email }

COMPANY: { **Department id** | Department }
DEPARTMENT DETAILS: { **Department** | Address | Phone number | Email }

INSURANCE: { **Insurance id** | Plans | Start date | Mode of purchase | Type of payment}
INSURANCE-PLAN: { Insurance benefit | **Plans** | Coverage }

CLAIMS: { **Claim id** | *Insurance id* | Status }
CLAIMS DETAIL: { **Claim id** | Issued Amount | Claimed Amount | Issued date | Claimed date }

HOSPITAL: { **Hospital id** | Hospital name | Phone number | Email | Address }

BILL: { **Bill id** | *Hospital id* | Date }
BILL-TOTAL-AMOUNT: { **Bill id** | Treatment charges | Medicine charges | Hospital charges }

Buyer-hospital: { **Buyer id** | **Hospital id** }
Insurance-claims: { **Claim id** | **Insurance id** }
Hospital-bill: { **Hospital id** | **Bill id** }

04 Implementation

MySQL

7 Tables

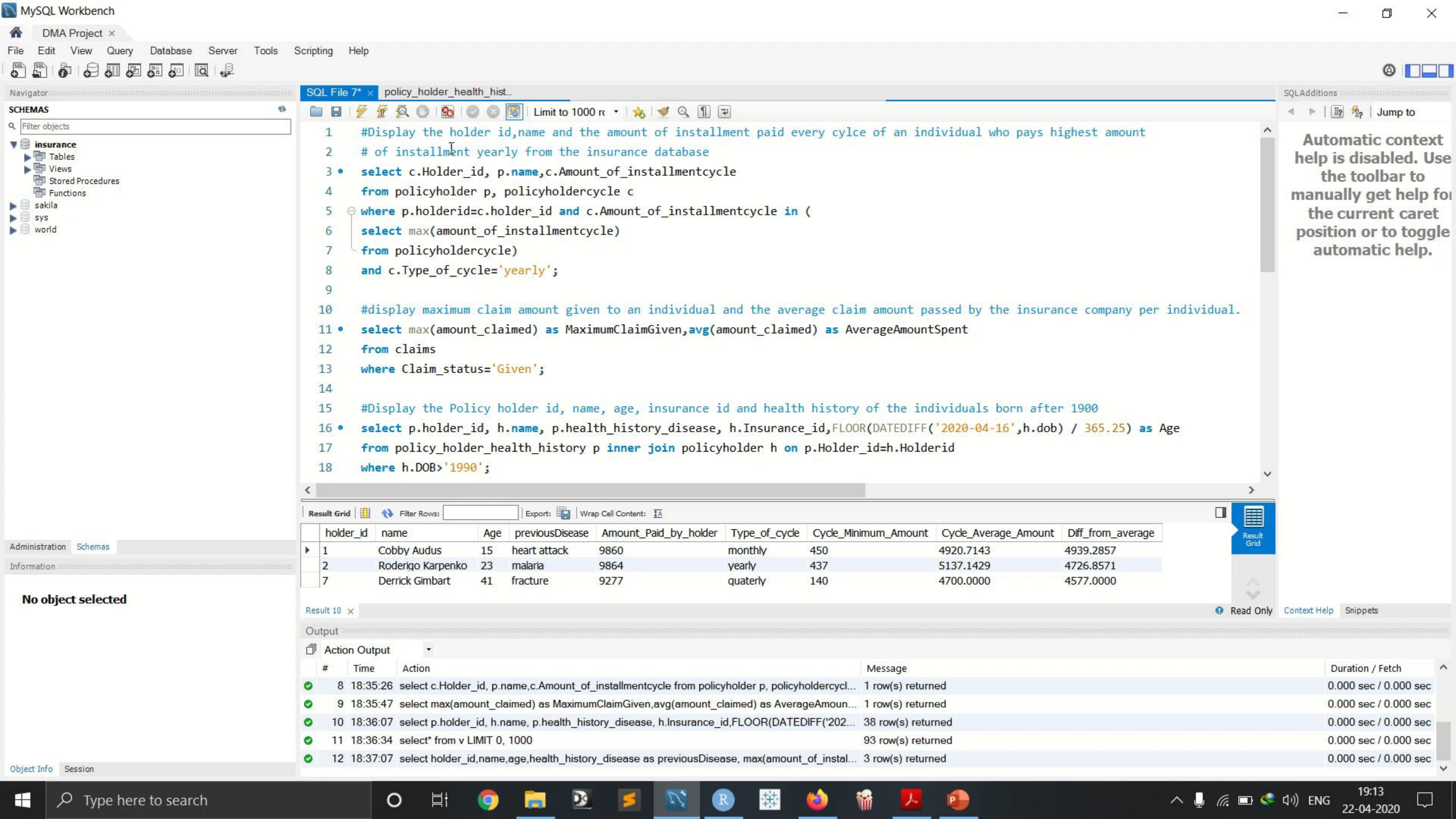
5 Queries
- Creation & Insertion
- Sub-Queries
- Aggregation Function
- Join Tables
- View Creation

NoSQL

2 Collections

5 Queries
- Creation & Insertion
- Basic Queries
- AggregatePipeline
- Map-Reduce Pipeline

MYSQL Demonstration

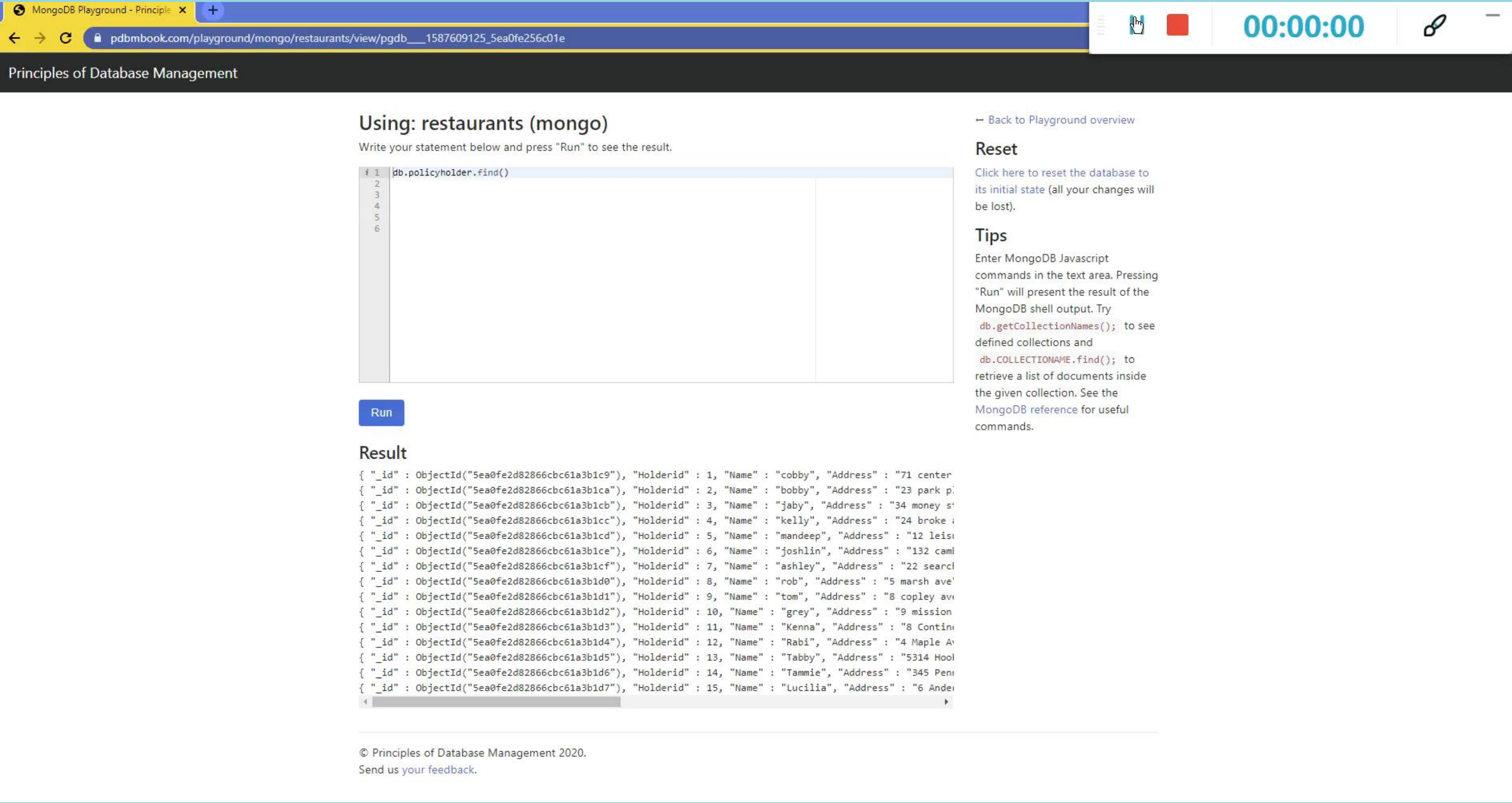


```
#Display the holder id,name and the amount of installment paid every cylce of an individual who pays highest amount
# of installment yearly from the insurance database
select c.Holder_id, p.name,c.Amount_of_installmentcycle
from policyholder p, policyholdercycle c
where p.holderid=c.holder_id and c.Amount_of_installmentcycle in (
select max(amount_of_installmentcycle)
from policyholdercycle)
and c.Type_of_cycle='yearly';

#display maximum claim amount given to an individual and the average claim amount passed by the insurance company per individual.
select max(amount_claimed) as MaximumClaimGiven,avg(amount_claimed) as AverageAmountSpent
from claims
where Claim_status='Given';

#Display the Policy holder id, name, age, insurance id and health history of the individuals born after 1900
select p.holder_id, h.name, p.health_history_disease, h.Insurance_id,FLOOR(DATEDIFF('2020-04-16',h.dob) / 365.25) as Age
from policy_holder_health_history p inner join policyholder h on p.Holder_id=h.Holderid
where h.DOB>'1990';
```

holder_id	name	Age	previousDisease	Amount_Paid_by_holder	Type_of_cycle	Cycle_Minimum_Amount	Cycle_Average_Amount	Diff_from_average
1	Cobby Audus	15	heart attack	9860	monthly	450	4920.7143	4939.2857
2	Roderigo Karpenko	23	malaria	9864	yearly	437	5137.1429	4726.8571
7	Derrick Gimbart	41	fracture	9277	quaterly	140	4700.0000	4577.0000

#	Time	Action	Message	Duration / Fetch
8	18:35:26	select c.Holder_id, p.name,c.Amount_of_installmentcycle from policyholder p, policyholdercycl...	1 row(s) returned	0.000 sec / 0.000 sec
9	18:35:47	select max(amount_claimed) as MaximumClaimGiven,avg(amount_claimed) as AverageAmoun...	1 row(s) returned	0.000 sec / 0.000 sec
10	18:36:07	select p.holder_id, h.name, p.health_history_disease, h.Insurance_id,FLOOR(DATEDIFF('202...	38 row(s) returned	0.000 sec / 0.000 sec
11	18:36:34	select* from v LIMIT 0, 1000	93 row(s) returned	0.000 sec / 0.000 sec
12	18:37:07	select holder_id,name,age,health_history_disease as previousDisease, max(amount_of_instal...	3 row(s) returned	0.000 sec / 0.000 sec

NoSQL Demonstration


MongoDB Playground - Principle
pdbmbook.com/playground/mongo/restaurants/view/pgdb___1587609125_5ea0fe256c01e
00:00:00

Principles of Database Management

Using: restaurants (mongo)

Write your statement below and press "Run" to see the result.

```
db.policyholder.find()
```

Result

```
{ "_id" : ObjectId("5ea0fe2d82866cbc61a3b1c9"), "Holderid" : 1, "Name" : "cobby", "Address" : "71 center
{ "_id" : ObjectId("5ea0fe2d82866cbc61a3b1ca"), "Holderid" : 2, "Name" : "bobby", "Address" : "23 park p
{ "_id" : ObjectId("5ea0fe2d82866cbc61a3b1cb"), "Holderid" : 3, "Name" : "jaby", "Address" : "34 money s
{ "_id" : ObjectId("5ea0fe2d82866cbc61a3b1cc"), "Holderid" : 4, "Name" : "kelly", "Address" : "24 broke
{ "_id" : ObjectId("5ea0fe2d82866cbc61a3b1cd"), "Holderid" : 5, "Name" : "mandeep", "Address" : "12 leis
{ "_id" : ObjectId("5ea0fe2d82866cbc61a3b1ce"), "Holderid" : 6, "Name" : "joshlin", "Address" : "132 cam
{ "_id" : ObjectId("5ea0fe2d82866cbc61a3b1cf"), "Holderid" : 7, "Name" : "ashley", "Address" : "22 searc
{ "_id" : ObjectId("5ea0fe2d82866cbc61a3b1d0"), "Holderid" : 8, "Name" : "rob", "Address" : "5 marsh ave
{ "_id" : ObjectId("5ea0fe2d82866cbc61a3b1d1"), "Holderid" : 9, "Name" : "tom", "Address" : "8 copley av
{ "_id" : ObjectId("5ea0fe2d82866cbc61a3b1d2"), "Holderid" : 10, "Name" : "grey", "Address" : "9 mission
{ "_id" : ObjectId("5ea0fe2d82866cbc61a3b1d3"), "Holderid" : 11, "Name" : "Kenna", "Address" : "8 Contin
{ "_id" : ObjectId("5ea0fe2d82866cbc61a3b1d4"), "Holderid" : 12, "Name" : "Rabi", "Address" : "4 Maple A
{ "_id" : ObjectId("5ea0fe2d82866cbc61a3b1d5"), "Holderid" : 13, "Name" : "Tabby", "Address" : "5314 Hoo
{ "_id" : ObjectId("5ea0fe2d82866cbc61a3b1d6"), "Holderid" : 14, "Name" : "Tammie", "Address" : "345 Pen
{ "_id" : ObjectId("5ea0fe2d82866cbc61a3b1d7"), "Holderid" : 15, "Name" : "Lucilia", "Address" : "6 Ande
```

← Back to Playground overview

Reset

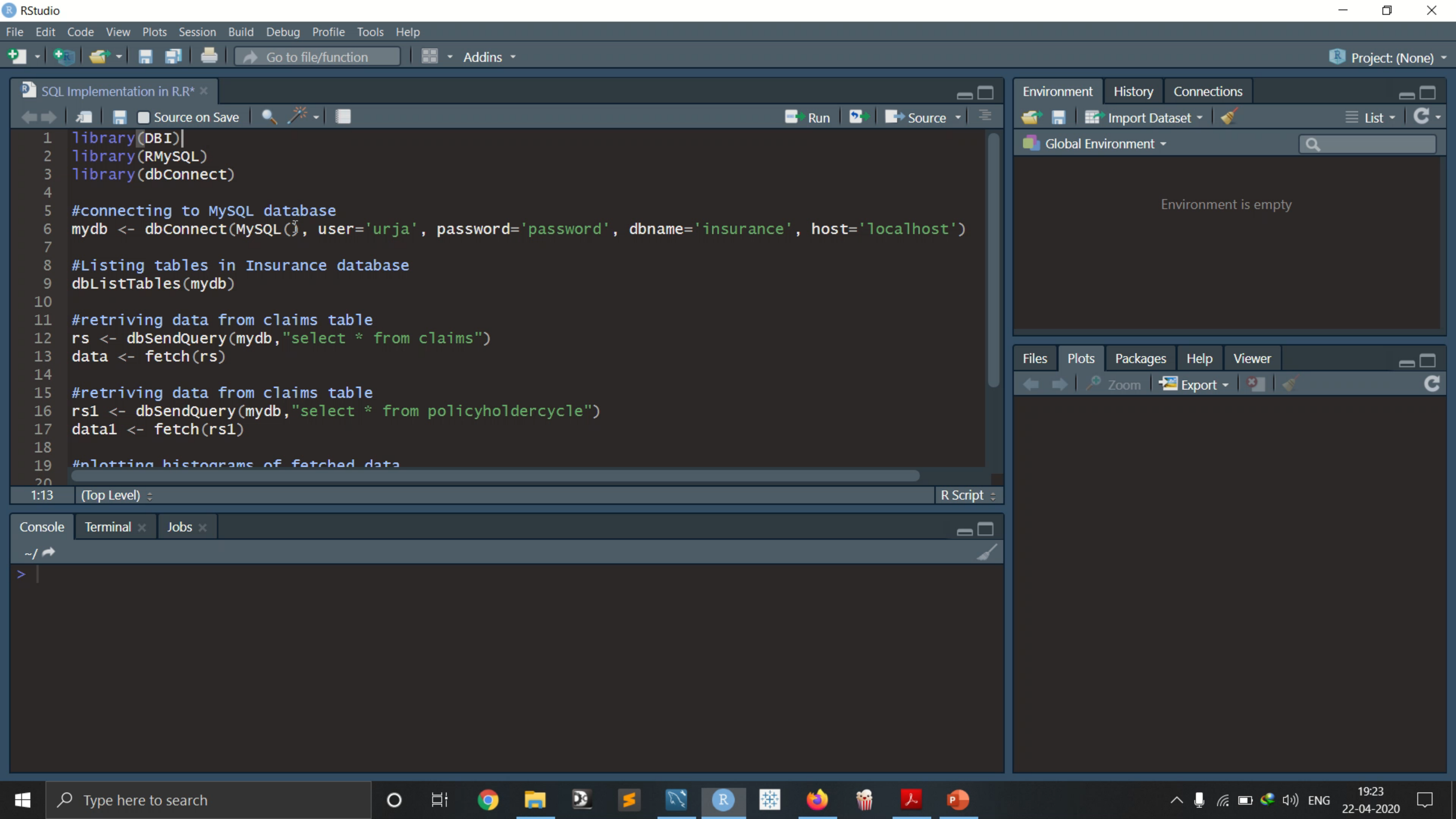
Click here to reset the database to its initial state (all your changes will be lost).

Tips

Enter MongoDB Javascript commands in the text area. Pressing "Run" will present the result of the MongoDB shell output. Try `db.getCollectionNames();` to see defined collections and `db.COLLECTIONAME.find();` to retrieve a list of documents inside the given collection. See the MongoDB reference for useful commands.

© Principles of Database Management 2020.
Send us your feedback.

R Analysis Demonstration



```
library(DBI)
library(RMySQL)
library(dbConnect)

#connecting to MySQL database
mydb <- dbConnect(MySQL(), user='urja', password='password', dbname='insurance', host='localhost')

#Listing tables in Insurance database
dbListTables(mydb)

#retriving data from claims table
rs <- dbSendQuery(mydb,"select * from claims")
data <- fetch(rs)

#retriving data from claims table
rs1 <- dbSendQuery(mydb,"select * from policyholdercycle")
data1 <- fetch(rs1)

#plotting histograms of fetched data
```

```
>
```

THANKS !