Insurance System Database

Alex Richin

Project Index

Introduction……..3

ER Diagram……4

Tables……16

Views……50

Stored Procedures……56

Reports……58

Security……64

Triggers……58

Conclusion……70→ 74

Problem Description (part 1)

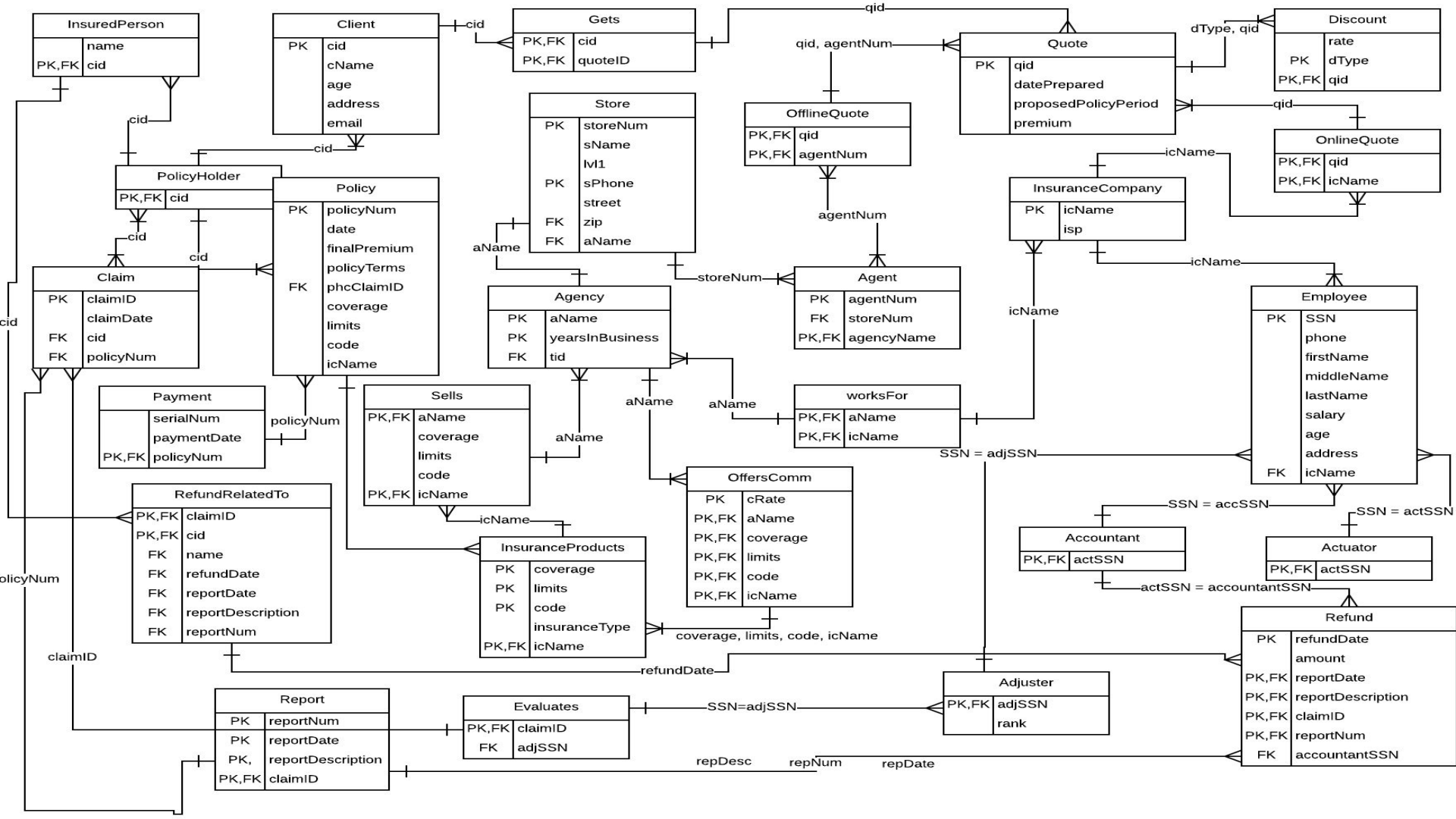
The premises of my proposal is to implement a database that consists of an insurance management system that will provide services to clients, insurance companies and agents. We will design, develop, and test the Insurance Management System in order to expand my knowledge and experience with designing relational databases.

-A client can get insurance quotes and file claims more than once. Each client contains a Client ID, Name, Age, Address, Phone Number, and an Email address that is recorded into the database system.

Provides 2 ways for clients to get quotes on their insurance plans. These ways are online with insurance companies and more physical with the insurance agents. Each quote contains the attributes Quote ID, Date Prepared, Proposed Policy Period, and Premium. This all pertains to an individual client. Some of these quotes have discounts which are directly related to discount name and the quote itself. Each individual quote can contain more than one discount within it.

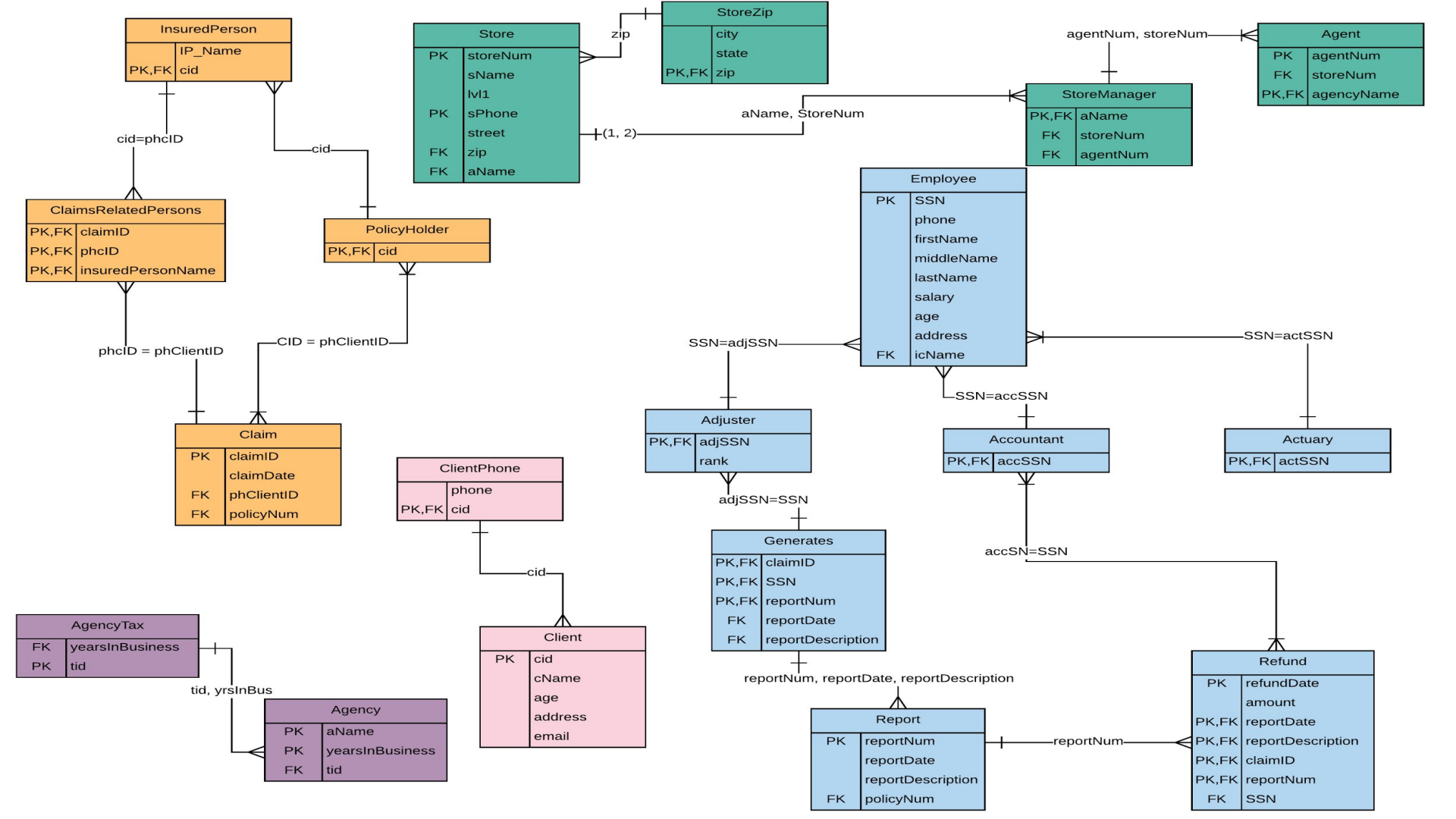
ER Diagrams

Main ER Diagram

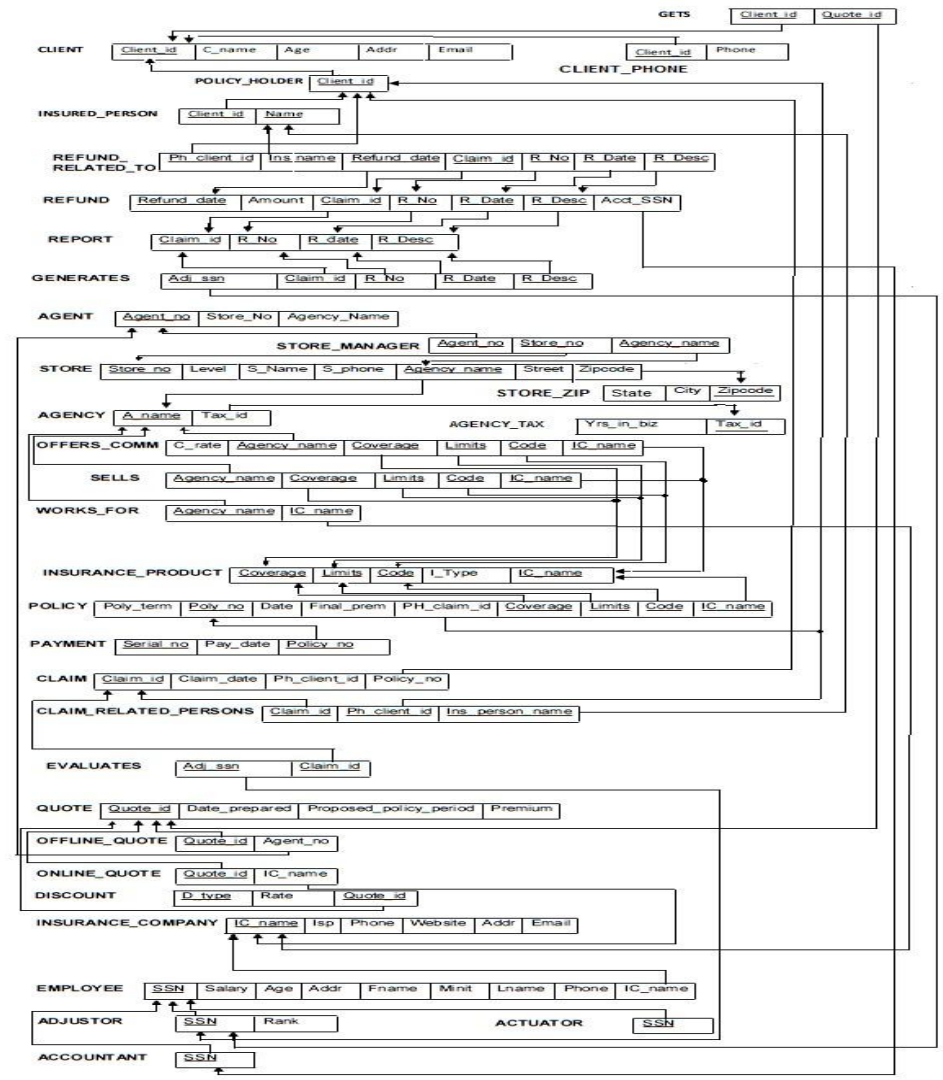

InsuredPerson
name
PK,FK cid
Client
PK cid
cName
age
address
email
Gets
PK,FK cid
PK,FK quoteID
Quote
PK qid
datePrepared
proposedPolicyPeriod
premium
Discount
rate
PK dType
PK,FK qid
OfflineQuote
PK,FK qid
PK,FK agentNum
OnlineQuote
PK,FK qid
PK,FK icName
PolicyHolder
PK,FK cid
Policy
PK policyNum
date
finalPremium
policyTerms
FK phcClaimID
coverage
limits
code
icName
Store
PK storeNum
sName
lvl1
PK sPhone
street
FK zip
FK aName
InsuranceCompany
PK icName
isp
Claim
PK claimID
claimDate
FK cid
FK policyNum
Agency
PK aName
PK yearsInBusiness
FK tid
Agent
PK agentNum
FK storeNum
PK,FK agencyName
Employee
PK SSN
phone
firstName
middleName
lastName
salary
age
address
FK icName
Payment
serialNum
paymentDate
PK,FK policyNum
Sells
PK,FK aName
coverage
limits
code
PK,FK icName
worksFor
PK,FK aName
PK,FK icName
RefundRelatedTo
PK,FK claimID
PK,FK cid
FK name
FK refundDate
FK reportDate
FK reportDescription
FK reportNum
OffersComm
PK cRate
PK,FK aName
PK,FK coverage
PK,FK limits
PK,FK code
PK,FK icName
InsuranceProducts
PK coverage
PK limits
PK code
insuranceType
PK,FK icName
Accountant
PK,FK actSSN
Actuator
PK,FK actSSN
Refund
PK refundDate
amount
PK,FK reportDate
PK,FK reportDescription
PK,FK claimID
PK,FK reportNum
FK accountantSSN
Report
PK reportNum
PK reportDate
PK, reportDescription
PK,FK claimID
Evaluates
PK,FK claimID
FK adjSSN
Adjuster
PK,FK adjSSN
rank
cid
qid
qid, agentNum
dType, qid
icName
agentNum
aName
storeNum
policyNum
SSN = adjSSN
SSN = accSSN
SSN = actSSN
actSSN = accountantSSN
coverage, limits, code, icName
claimID
refundDate
SSN=adjSSN
repDesc
repNum
repDate

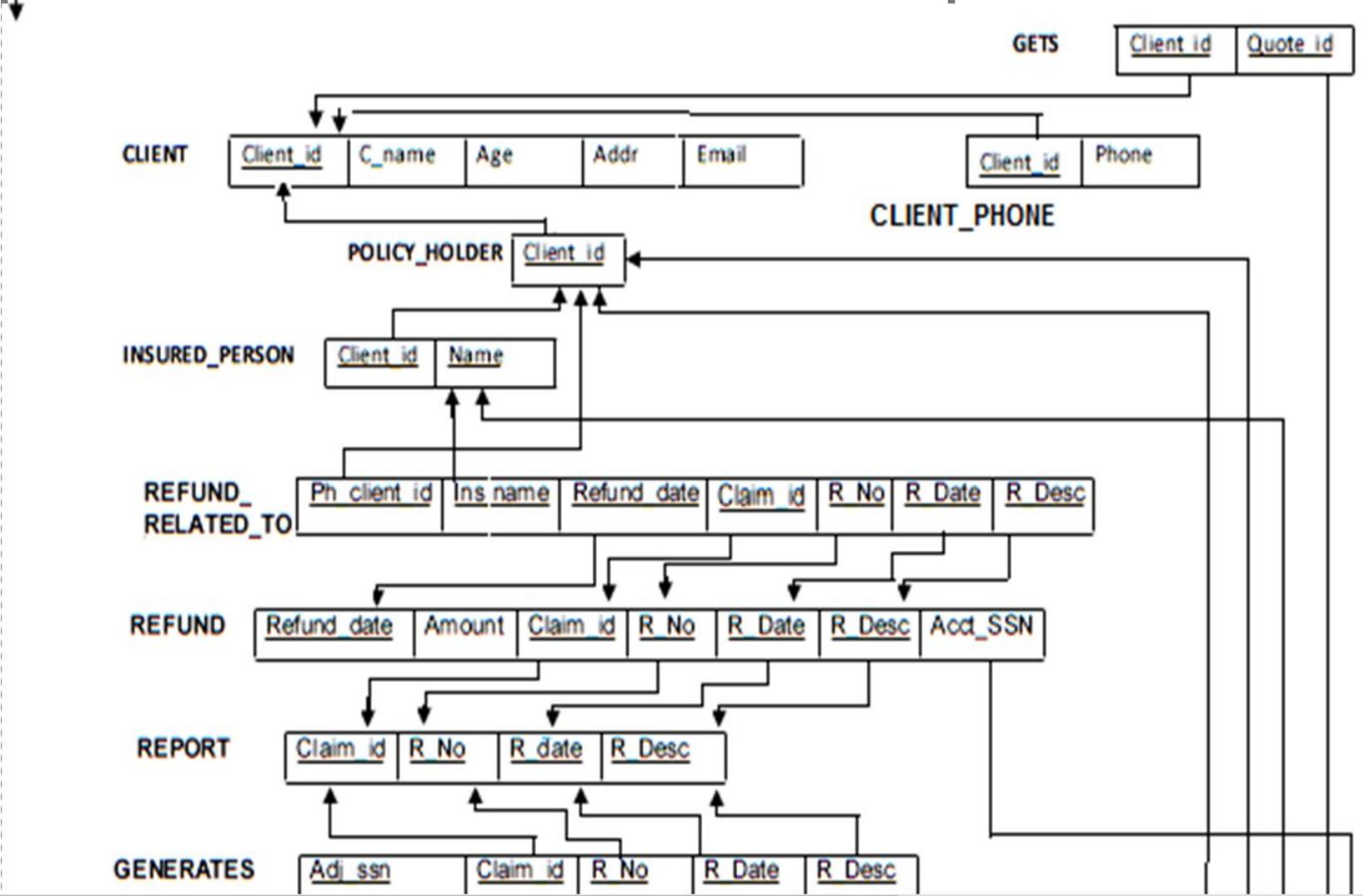
Partial ER Diagram

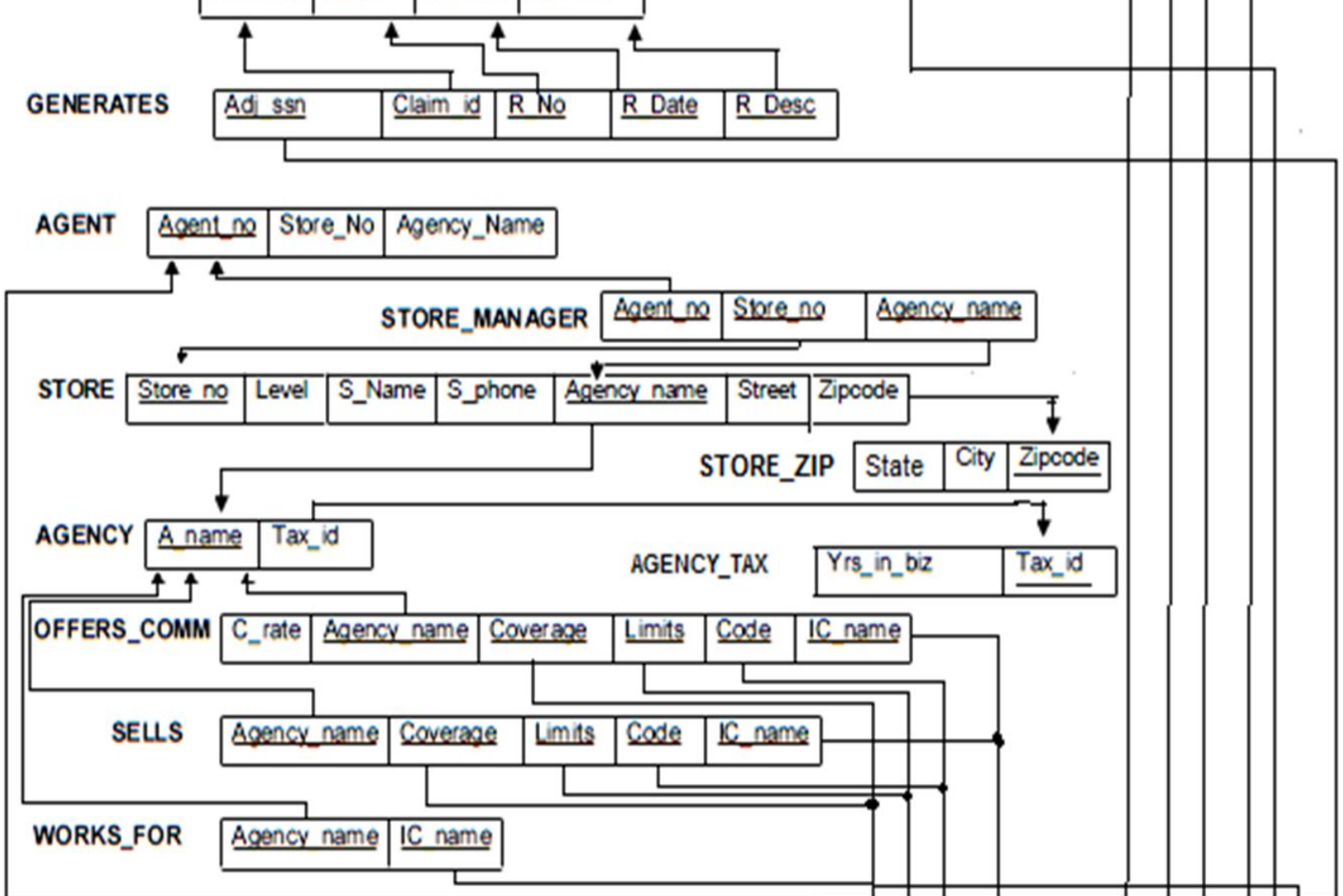
was not allowed to use any more objects w/o paying via lucidchart

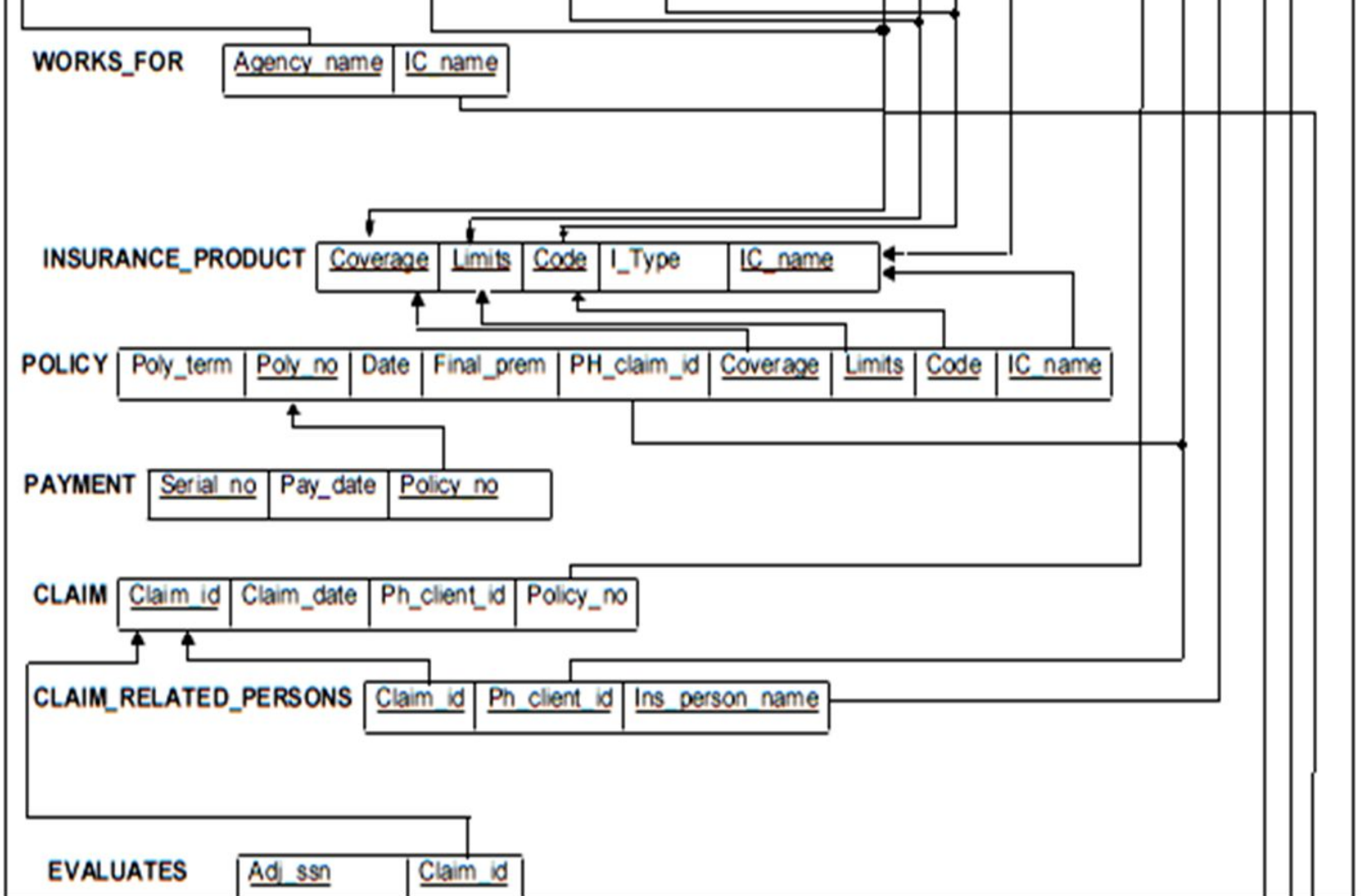

InsuredPerson
IP_Name
PK,FK cid
Store
PK storeNum
sName
lvl1
PK sPhone
street
FK zip
FK aName
StoreZip
city
state
PK,FK zip
zip
Agent
PK agentNum
FK storeNum
PK,FK agencyName
agentNum, storeNum
StoreManager
PK,FK aName
FK storeNum
FK agentNum
aName, StoreNum
(1, 2)
cid=phcID
cid
ClaimsRelatedPersons
PK,FK claimID
PK,FK phcID
PK,FK insuredPersonName
PolicyHolder
PK,FK cid
Employee
PK SSN
phone
firstName
middleName
lastName
salary
age
address
FK icName
SSN=adjSSN
SSN=actSSN
SSN=accSSN
phcID = phClientID
CID = phClientID
Claim
PK claimID
claimDate
FK phClientID
FK policyNum
Adjuster
PK,FK adjSSN
rank
Accountant
PK,FK accSSN
Actuary
PK,FK actSSN
ClientPhone
phone
PK,FK cid
adjSSN=SSN
Generates
PK,FK claimID
PK,FK SSN
PK,FK reportNum
FK reportDate
FK reportDescription
accSN=SSN
cid
Client
PK cid
cName
age
address
email
AgencyTax
FK yearsInBusiness
PK tid
tid, yrsInBus
Agency
PK aName
PK yearsInBusiness
FK tid
reportNum, reportDate, reportDescription
Report
PK reportNum
reportDate
reportDescription
FK policyNum
reportNum
Refund
PK refundDate
amount
PK,FK reportDate
PK,FK reportDescription
PK,FK claimID
PK,FK reportNum
FK SSN

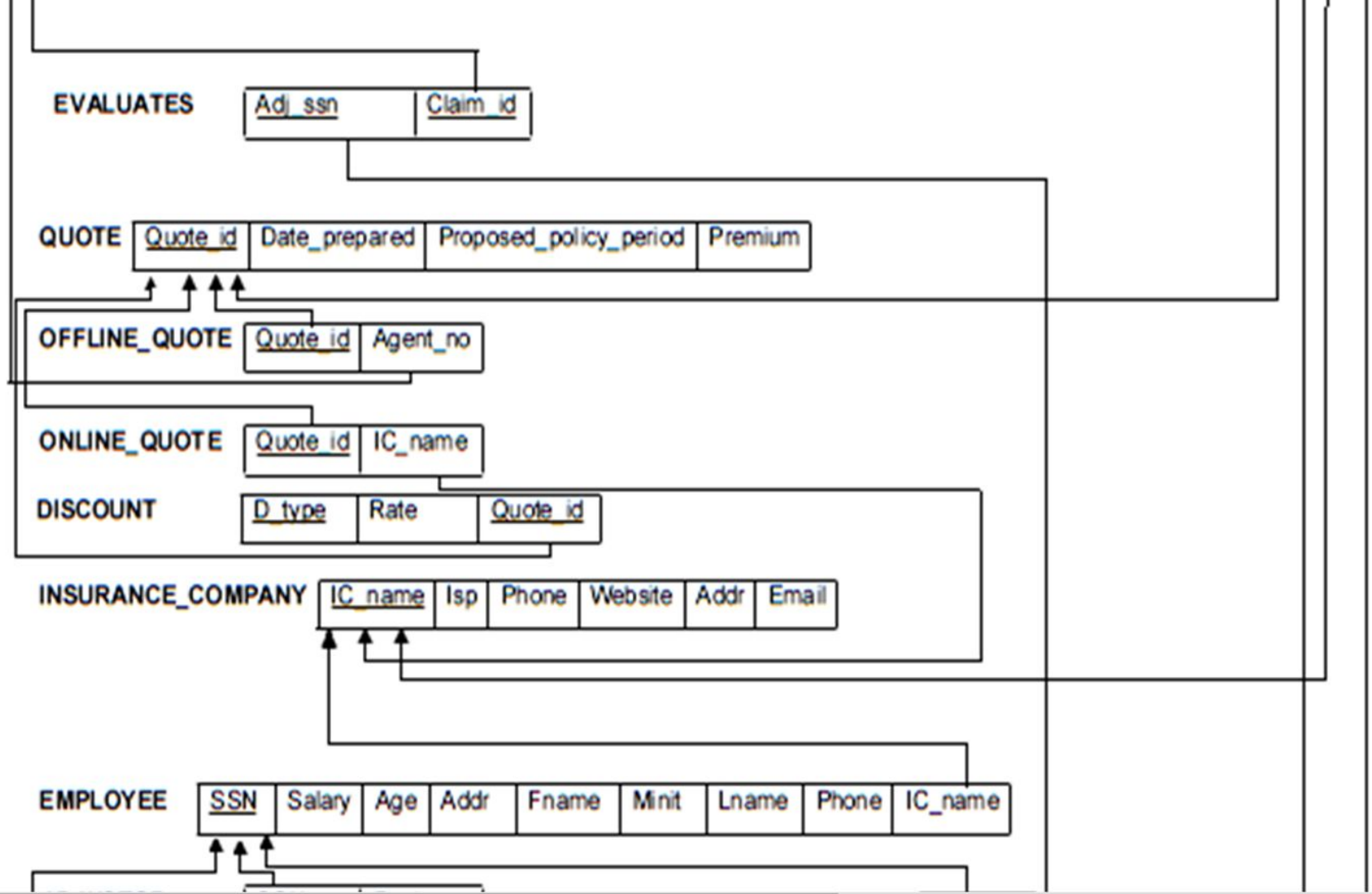
Additional Diagrams

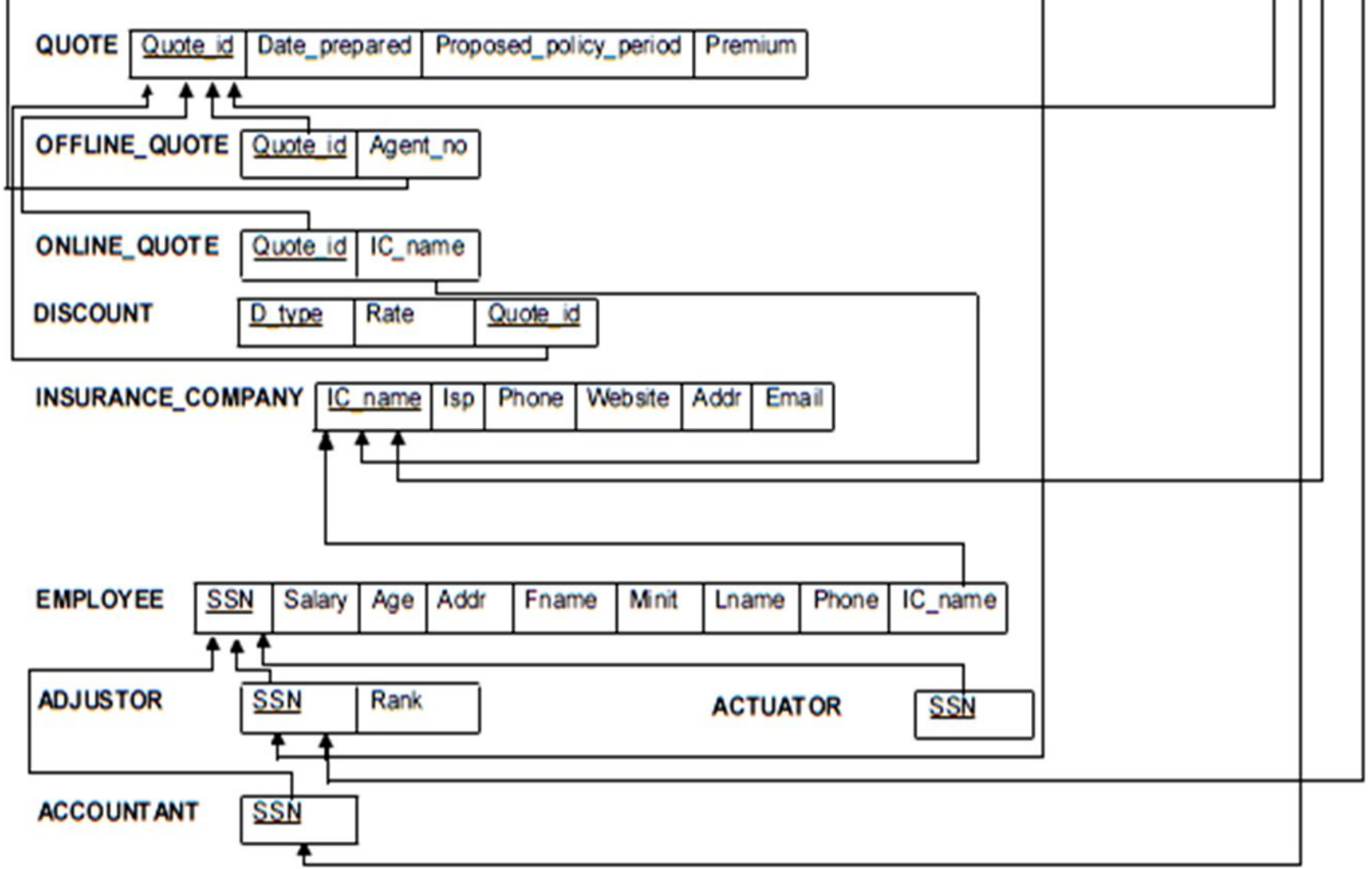

GETS
Client_id
Quote_id
CLIENT
Client_id
C_name
Age
Addr
Email
Client_id
Phone
CLIENT_PHONE
POLICY_HOLDER
Client_id
INSURED_PERSON
Client_id
Name
REFUND_RELATED_TO
Ph_client_id
Ins_name
Refund_date
Claim_id
R_No
R_Date
R_Desc
REFUND
Refund_date
Amount
Claim_id
R_No
R_Date
R_Desc
Acct_SSN
REPORT
Claim_id
R_No
R_date
R_Desc
GENERATES
Adj_ssn
Claim_id
R_No
R_Date
R_Desc
AGENT
Agent_no
Store_No
Agency_Name
STORE_MANAGER
Agent_no
Store_no
Agency_name
STORE
Store_no
Level
S_Name
S_phone
Agency_name
Street
Zipcode
STORE_ZIP
State
City
Zipcode
AGENCY
A_name
Tax_id
AGENCY_TAX
Yrs_in_biz
Tax_id
OFFERS_COMM
C_rate
Agency_name
Coverage
Limits
Code
IC_name
SELLS
Agency_name
Coverage
Limits
Code
IC_name
WORKS_FOR
Agency_name
IC_name
INSURANCE_PRODUCT
Coverage
Limits
Code
I_Type
IC_name
POLICY
Poly_term
Poly_no
Date
Final_prem
PH_claim_id
Coverage
Limits
Code
IC_name
PAYMENT
Serial_no
Pay_date
Policy_no
CLAIM
Claim_id
Claim_date
Ph_client_id
Policy_no
CLAIM_RELATED_PERSONS
Claim_id
Ph_client_id
Ins_person_name
EVALUATES
Adj_ssn
Claim_id
QUOTE
Quote_id
Date_prepared
Proposed_policy_period
Premium
OFFLINE_QUOTE
Quote_id
Agent_no
ONLINE_QUOTE
Quote_id
IC_name
DISCOUNT
D_type
Rate
Quote_id
INSURANCE_COMPANY
IC_name
Isp
Phone
Website
Addr
Email
EMPLOYEE
SSN
Salary
Age
Addr
Fname
Minit
Lname
Phone
IC_name
ADJUSTOR
SSN
Rank
ACTUATOR
SSN
ACCOUNTANT
SSN


GETS
Client_id
Quote_id
CLIENT
Client_id
C_name
Age
Addr
Email
Client_id
Phone
CLIENT_PHONE
POLICY_HOLDER
Client_id
INSURED_PERSON
Client_id
Name
REFUND_
RELATED_TO
Ph_client_id
Ins_name
Refund_date
Claim_id
R_No
R_Date
R_Desc
REFUND
Refund_date
Amount
Claim_id
R_No
R_Date
R_Desc
Acct_SSN
REPORT
Claim_id
R_No
R_date
R_Desc
GENERATES
Adj_ssn
Claim_id
R_No
R_Date
R_Desc


GENERATES
Adj_ssn
Claim_id
R_No
R_Date
R_Desc
AGENT
Agent_no
Store_No
Agency_Name
STORE_MANAGER
Agent_no
Store_no
Agency_name
STORE
Store_no
Level
S_Name
S_phone
Agency_name
Street
Zipcode
STORE_ZIP
State
City
Zipcode
AGENCY
A_name
Tax_id
AGENCY_TAX
Yrs_in_biz
Tax_id
OFFERS_COMM
C_rate
Agency_name
Coverage
Limits
Code
IC_name
SELLS
Agency_name
Coverage
Limits
Code
IC_name
WORKS_FOR
Agency_name
IC_name


WORKS_FOR
Agency_name
IC_name
INSURANCE_PRODUCT
Coverage
Limits
Code
I_Type
IC_name
POLICY
Poly_term
Poly_no
Date
Final_prem
PH_claim_id
Coverage
Limits
Code
IC_name
PAYMENT
Serial_no
Pay_date
Policy_no
CLAIM
Claim_id
Claim_date
Ph_client_id
Policy_no
CLAIM_RELATED_PERSONS
Claim_id
Ph_client_id
Ins_person_name
EVALUATES
Adj_ssn
Claim_id


EVALUATES
Adj_ssn
Claim_id
QUOTE
Quote_id
Date_prepared
Proposed_policy_period
Premium
OFFLINE_QUOTE
Quote_id
Agent_no
ONLINE_QUOTE
Quote_id
IC_name
DISCOUNT
D_type
Rate
Quote_id
INSURANCE_COMPANY
IC_name
Isp
Phone
Website
Addr
Email
EMPLOYEE
SSN
Salary
Age
Addr
Fname
Minit
Lname
Phone
IC_name


QUOTE
Quote_id
Date_prepared
Proposed_policy_period
Premium
OFFLINE_QUOTE
Quote_id
Agent_no
ONLINE_QUOTE
Quote_id
IC_name
DISCOUNT
D_type
Rate
Quote_id
INSURANCE_COMPANY
IC_name
Isp
Phone
Website
Addr
Email
EMPLOYEE
SSN
Salary
Age
Addr
Fname
Minit
Lname
Phone
IC_name
ADJUSTOR
SSN
Rank
ACTUATOR
SSN
ACCOUNTANT
SSN

Create Table Statements

Client Table

```
CREATE TABLE Client (
        cid                 char(6) not null unique,
        clientName          char(30),
        age                 numeric check (age > 0),
        address             char(40),
        email               char(40),
PRIMARY KEY(cid)
);
```

Functional Dependencies

cid → clientName, age, address, email

Sample Data

SELECT * FROM Clients;

```
5   SELECT * FROM Clients;
```

Data Output | Explain | Messages | History

cid character	clientname character	age numeric	address character	email character
c001	Alan Labouseur	25	3399 N Rd	Alan.Labouseur@marist.edu
c002	Alex Richin	21	37 Kelsey Rd	Alexander.Richin1@marist.edu
c003	Danielle DiLonardo	29	313 W High St	DanielleDilonardo@gmail.com
c004	John Smith	42	219 West Cedar St	John.Smith@gmail.com
c005	Daniel Nelson	21	29 Parker Pl	Daniel.Nelson@yahoo.com
c006	Sean Hayden	35	88 Houstan Ave	Sean.Hayden@marist.edu
c007	John Locke	41	64 Geremy Rd	J_locke@aol.com
c008	Sarah Thea	32	711 Rodger Pl	S.R@gmail.com
c009	Jaiden Martin	41	32 Pike Rd	Jaiden_Sm@yahoo.com
c010	George Wright	54	981 Main St	GEWR@microsoft.com
c011	Jeremy Renner	37	21 HollyWood Ave	J_renner@stars.com
c012	Liam Johnson	62	45 Hawthorne Rd	L_johnson@gmail.com
c013	Andreas Karsiotis	32	21 Anglewood Ave	Andreas.Karsiotis@rowan.edu
c014	Jamie Ricky	26	467 Killington Pl	Jamie.Rick@yahoo.com
c015	Martin Henry	72	71 Almond Tr	MangoMartin@aol.com
c016	Henry Pratt	35	23 Angel Rd	Henry.Pratt@marist.edu
c017	Patrick Trainor	43	10 Allendale Rd	Ptrain@gmail.com
c018	Dr. Labouseur	27	55 Alan Dr	Dr.Alan.Labouseur@marist.edu
c019	Professor Alan	29	23 Labouseur Ln	Prof_AlanL@marist.edu

ClientPhone Table

CREATE STATEMENT:

```
CREATE TABLE ClientsPhone (
      cid         char(6) references Clients(cid),
      phone     char(12),
PRIMARY KEY(cid)
);
```

Functional Dependencies

cid → phone

Sample Output:

SELECT * FROM ClientPhone;

```
5   SELECT * FROM ClientPhone;
```

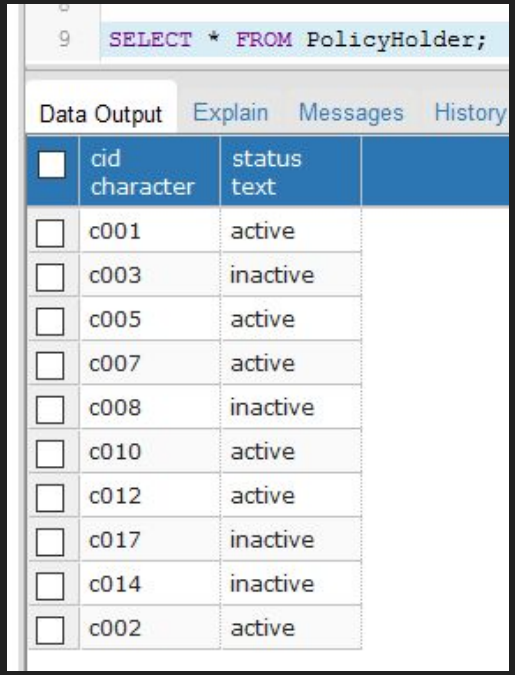
Data Output | Explain | Messages | History

cid character	phone character
c001	201-994-9394
c002	201-321-5432
c004	345-213-4576
c006	980-234-1289
c005	265-342-9820
c009	450-342-2154
c007	321-465-9082
c017	654-321-9087
c015	540-645-7329
c014	430-908-8600
c008	350-345-9374
c003	785-832-8970

PolicyHolder Table

```
CREATE TABLE PolicyHolder (
        cid             char(6) REFERENCES Clients(cid),
        status          text,
PRIMARY KEY(cid)
);
```

Sample Data →



```
9  SELECT * FROM PolicyHolder;
```

Data Output | Explain | Messages | History

cid character	status text
c001	active
c003	inactive
c005	active
c007	active
c008	inactive
c010	active
c012	active
c017	inactive
c014	inactive
c002	active

FUNCTIONAL DEPENDENCIES:

cid → status

InsuredPerson Table

```
CREATE TABLE InsuredPerson (
        name        char(30) NOT NULL,
        cid     char(6) not null unique references Clients(cid),
PRIMARY KEY( cid)
);
```

Functional Dependencies:

cid → name

Sample Data:

```
5
6   SELECT * FROM InsuredPerson;
```

Data Output | Explain | Messages | History

name character	cid character
Dr. Labouseur	c018
Alan Labouseur	c001
Professor Alan	c019
Alex Richin	c002
Patrick Trainor	c017
Liam Johnson	c012
John Smith	c004
George Wright	c010
Henry Pratt	c016

Quote Table

```
CREATE TABLE Quote (
        qid                     char(6) NOT NULL UNIQUE,
        datePrepared            date,
        proposedPolicyPeriod    numeric,
        premium                 numeric(10,2),
PRIMARY KEY(qid)
);
```

Functional Dependencies

qid → datePrepared, proposedPolicyPeriod, premium

Sample Data:

```
4
5  SELECT * FROM Quote;
```

Data Output | Explain | Messages | History

qid character	dateprepared date	propose... numeric	premium numeric (10,2)
q001	2016-04-10	5	100025.5
q002	2016-04-11	2	100055.72
q003	2016-04-10	6	12075.2
q004	2016-04-13	9	15000
q005	2016-04-12	20	175000
q006	2016-04-12	15	15000
q007	2016-03-19	10	20000
q008	2016-03-18	4	25000.5
q009	2016-03-20	10	785000.76
q010	2016-05-16	28	1235110
q011	2016-05-19	51	2222222.22
q012	2016-10-20	25	50000050.05

AgencyTaxID Table

```
create table AgencyTaxID (
        tid                     char(10) not null,
        yrsInBus                numeric,
primary key(tid)
);
```

Sample Data →

```
4
5   SELECT * FROM AgencyTaxID;
```

Data Output | Explain | Messages | Histor

tid character	yrsinbus numeric
t001	5
t002	3
t003	6
t004	7
t005	10
t006	15
t007	4
t008	5
t009	7
t010	27

Functional Dependencies:

tid → yrsInBus

Agency Table

Query: SELECT * FROM Agency;

```
CREATE TABLE Agency (
        aName           char(20) not null unique,
        tid             char(10) not null references AgencyTaxID(tid),
PRIMARY KEY(tid)
);
```

Sample Data →

```
4
5   SELECT * FROM Agency;
```

Data Output | Explain | Messages

aname character	tid character
Milford	t003
Jennings	t008
Mendleton	t005
Pierson	t009
Richard Co.	t010

Functional Dependencies:

tid → aName

StoreZip Table

```
CREATE TABLE StoreZip (
        zid     char(4) NOT NULL UNIQUE,
        city    char(20),
        state   char(20),
        zip     numeric not null unique,
PRIMARY KEY(zid)
);
```

Functional Dependencies

zid, zip → state, city

Sample Data:

```
5   SELECT * FROM StoreZip;
```

Data Output | Explain | Messages | History

city character	state character	zip numeric (5)
Alanville	New York	12604
Denver	Colorado	8723
Boston	Mass.	26734
Seattle	Washington	73512
Sacremento	California	8341

Store Table

```
CREATE TABLE Store (
        storeNum        numeric not null unique,
        storeName       char(20),
        lvl1            integer,
        sPhone          char(12),
        zip             numeric(5,0) references StoreZip(zip),
        aName           char(20) references Agency(aName),
        street          numeric,
PRIMARY KEY(storeNum, aName)
);
```

Functional Dependencies:

storeNum, aName → storeName, lvl1, sPhone, street, zip

```
9   SELECT * FROM Store;
```

Data Output | Explain | Messages | History

storenum numeric	storename character	lvl1 integer	sphone character	zip numeric ...	aname character	street numeric
100	Alans Store	2	201-443-2345	12604	Milford	80
101	Alexs store	5	201-342-5432	26734	Jennings	30
102	Johns store	10	543-234-4567	8723	Mendleton	67
103	JLB	15	321-456-3254	73512	Richard Co.	187
104	Pierson Place	10	211-345-9823	8341	Pierson	95

← Sample Data

Agent Table

```
CREATE TABLE Agent (
        agentNum            numeric not null unique,
        storeNum            numeric references Store(storeNum),
        aName               char(20) references Agency(aName),
PRIMARY KEY(agentNum)
);
```

Functional Dependencies:

agentNum → storeNum, aName

Sample Data

```
8  SELECT * FROM Agent;
```

Data Output | Explain | Messages | History

agentnum numeric	storenum numeric	aname character
100	100	Milford
101	102	Mendleton
102	102	Mendleton
103	101	Jennings
104	100	Milford
105	103	Richard Co.
106	104	Pierson
107	103	Richard Co.
108	104	Pierson

InsuranceCompany Table

```
CREATE TABLE InsuranceCompany (
        icName          char(20) not null unique,
        isp             char(20),
        website         char(40),
        address         char(40),
        email           char(20),
        phone           char(12),
PRIMARY KEY(icName)
);
```

Sample Data

```
5  SELECT * FROM InsuranceCompany;
```

Data Output | Explain | Messages | History

icname character	isp character	website character	address character	email character	phone character
State Farm Insurance	ISP-0101	statefarm.com	31 milton rd	statefarm@insurance.com	111-111-1111
Geico Insurance	ISP-0202	geico.com	24 george st	insurance@geico.com	201-342-3434
Allstate Insurance	ISP-0303	allstate.com	1 cedar st	insurance@allstate.com	321-342-3421
Alan Insurance	ISP-0404	alan.com	2 alan ct	alan_insurance@alan.com	201-342-3421
Labouseur Insurance	ISP-0505	Labouseur.com	3 old chimney rd	labouseur_inurance@la...	845-321-3452
RichIn Insurance	ISP-0606	richIn.com	81 lane rd	richin_insurance@richIn....	201-446-7400

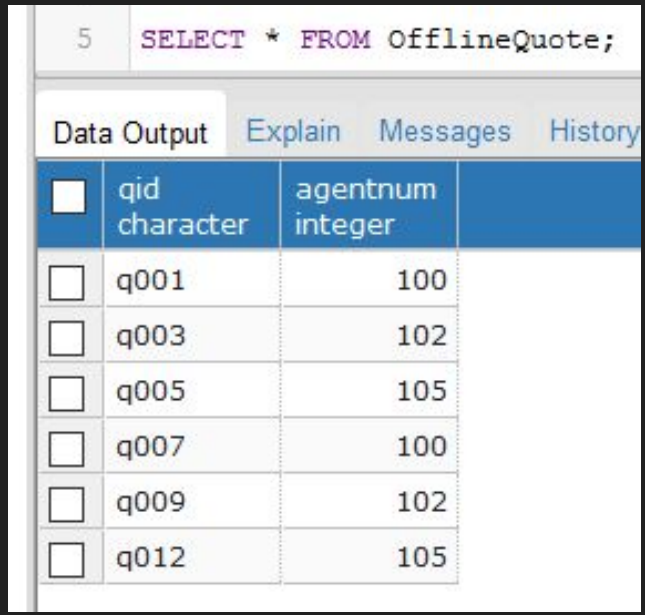
Functional Dependencies:

icName → isp, website, address, email, phone

OfflineQuote Table

```
CREATE TABLE OfflineQuote (
        qid             char(7) references Quote(qid) not null,
        agentNum        integer references Agent(agentNum),
PRIMARY KEY(qid)
);
```

Sample Data →



```
5   SELECT * FROM OfflineQuote;
```

Data Output | Explain | Messages | History

qid character	agentnum integer
q001	100
q003	102
q005	105
q007	100
q009	102
q012	105

Functional Dependencies:

qid → agentNum

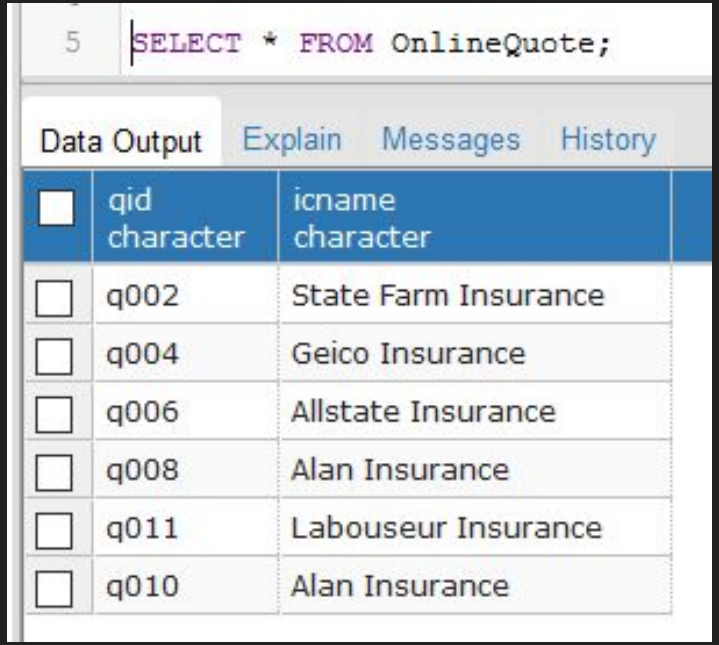
OnlineQuote Table

```
CREATE OnlineQuote (
        qid             char(7) references Quote(qid) not null,
        icName          char(20) references InsuranceCompany(icName),
PRIMARY KEY(qid)
);
```

Functional Dependencies:

qid → icName

Sample Data:



qid character	icname character
q002	State Farm Insurance
q004	Geico Insurance
q006	Allstate Insurance
q008	Alan Insurance
q011	Labouseur Insurance
q010	Alan Insurance

InsuranceProducts Table

```
CREATE TABLE InsuranceProducts (
        coverage            char(40) not null unique,
        limits              char(40) not null unique,
        code                char(20) not null unique,
        insuranceType       char(10),
        icName              char(20)  not null unique references InsuranceCompany(icName),
PRIMARY KEY(coverage, limits, code, icName)
);
```

Sample Data →

```
5   SELECT * FROM InsuranceProducts;
```

Data Output | Explain | Messages | History

coverage character	limits character	code character	insurancetype character	icname character
cov-50	lim-20	c000	it-00-10	Alan Insurance
cov-51	lim-21	c001	it-00-11	Labouseur Insurance
cov-52	lim-22	c002	it-00-12	RichIn Insurance
cov-53	lim-23	c003	it-00-13	Geico Insurance
cov-54	lim-24	c004	it-00-14	Allstate Insurance
cov-55	lim-25	c005	it-00-15	State Farm Insurance

Functional Dependencies:

Coverage, limits, code, icName → insuranceType

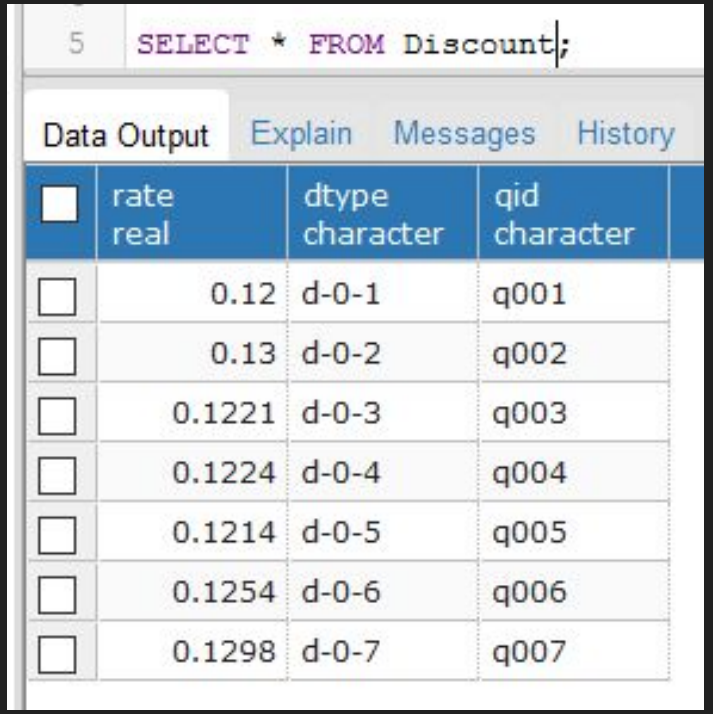
Discount Table

```
CREATE TABLE Discount (
        rate            float(8),
        dType           char(20) not null unique,
        qid             char(7) references Quote(qid),
PRIMARY KEY(dType, qid)
);
```

Functional Dependencies:

dType, qid → rate

Sample Data



rate real	dtype character	qid character
0.12	d-0-1	q001
0.13	d-0-2	q002
0.1221	d-0-3	q003
0.1224	d-0-4	q004
0.1214	d-0-5	q005
0.1254	d-0-6	q006
0.1298	d-0-7	q007

Employee Table

```
CREATE TABLE Employee (
        SSN             char(11) not null unique,
        icName          char(20) REFERENCES InsuranceCompany(icName),
        phone           char(12),
        firstName       char(20),
        middleInitial   char(20),
        lastName        char(20),
        salary          float(8),
        age             integer,
        address         char(40),
PRIMARY KEY(SSN)
);
```

Sample Data →

```
8  SELECT * FROM Employee;
```

Data Output | Explain | Messages | History

ssn character	phone character	firstname character	middleini... character	lastname character	salary real	age integer	address character	icname character
10-2000-200	201-456-7892	John	F	Doe	80000	35	10 mill rd	Alan Insurance
20-2000-200	202-456-7892	Alan	G	Labouseur	8000000	28	9 mill rd	Alan Insurance
30-3000-300	203-456-7892	Alex	J	Richin	85000	31	11 mill rd	Labouseur Insurance
40-4000-400	204-456-7892	Pete	L	Mattheus	100000	27	12 mill rd	RichIn Insurance
50-5000-500	205-456-7892	George	A	Johnson	90000	39	15 mill rd	Geico Insurance
60-6000-600	201-201-2012	John	K	Finnigan	60000	43	10 morto...	Labouseur Insurance
70-7000-700	230-230-2302	Ken	P	Rodgers	90000	27	115 hillto...	State Farm Insurance

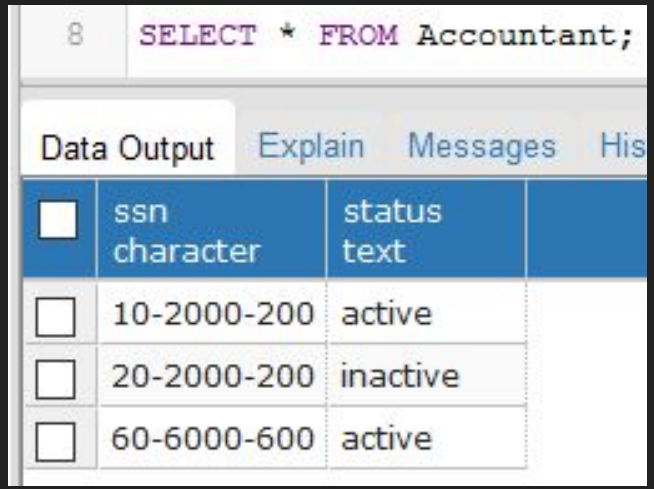
Functional Dependencies:

SSN → phone, firstName, middleInital, lastName, salary, age, address, icName

Accountant Table

```
CREATE TABLE Accountant (
        SSN             char(11) REFERENCES Employee(SSN),
        status          text,
PRIMARY KEY(Ssn)
);
```

Sample Data



Functional Dependencies:

SSN → status

Employee(SSN) ⟶ Accountant(SSN)

Adjuster Table

```
CREATE TABLE Adjuster (
        SSN             char(11) references Employee(SSN),
        rank            integer,
PRIMARY KEY(SSN)
);
```

Sample Data

```
5   SELECT * FROM Adjuster;
```

Data Output | Explain | Messages | Hist

ssn character	rank integer
30-3000-300	2
40-4000-400	5

Functional Dependencies:

SSN → rank

Also,

SSN from employee → SSN for Adjuster a.k.a adjSSN

Actuator Table

```
CREATE TABLE Actuary (
        SSN             char(11) REFERENCES Employee(SSN),
        status          text,
PRIMARY KEY(SSN)
);
```

Functional Dependencies:

SSN → status

Also,

SSN → Acturary(SSN)

Sample Data

```
8   SELECT * FROM Actuator;
```

Data Output | Explain | Messages

ssn character	status text
50-5000-500	active
70-7000-700	active

```
CREATE TABLE Policy (
        policyNum           integer,
        pDate               char(20),
        finalPremium        float(8),
        policyTerms         char(40),
        phClientID          char(12) references PolicyHolder(cid),
        coverage            char(40) references InsuranceProducts(coverage),
        limits              char(40) references InsuranceProducts(limits),
        code                char(20) references InsuranceProducts(code),
        icName              char(20) references InsuranceProducts(icName),
PRIMARY KEY(policyNum)
);
```

Policy Table

Functional Dependencies:

policyNum→ pDate, finalPremium, policyTerms, phClientID, coverage, limits, code, icName

```
5  SELECT * FROM Policy;
```

Data Output | Explain | Messages | History

policynum integer	pdate date	finalpre... real	policyterms character	cid character	coverage character	limits character	code character	icname character
1001	2016-10-10	250	policy terms - 1	c001	cov-50	lim-20	c000	Alan Insurance
1002	2016-10-10	200	policy terms - 2	c003	cov-51	lim-21	c001	Labouseur Insurance
1003	2016-10-11	250	policy terms - 3	c005	cov-52	lim-22	c002	RichIn Insurance
1004	2016-10-11	225	policy terms - 4	c007	cov-53	lim-23	c003	Labouseur Insurance
1005	2016-10-12	150	policy terms - 5	c010	cov-55	lim-25	c005	State Farm Insurance

Sample Data ←

Claim Table

```
CREATE TABLE Claim (
        claimID             char(12),
        claimDate           date,
        phClientID          char(6) references PolicyHolder(cid),
        policyNum           integer references Policy(policyNum),
PRIMARY KEY(claimID)
);
```

Functional Dependencies:

claimID→ claimDate, phClientID, policyNum

```
5  SELECT * FROM claim;
```

Data Output | Explain | Messages | History

claimid character	claimdate character	cid character	policynum integer
claim-100	10-10-2016	c001	1001
claim-101	10-10-2016	c003	1002
claim-102	10-10-2016	c005	1003
claim-103	10-10-2016	c010	1005
claim-104	10-10-2016	c007	1004

Payment Table

```
CREATE TABLE Payment (
        serialNum               integer not null unique,
        paymentDate             char(20),
        policyNum               integer references Policy(policyNum),
PRIMARY KEY(policyNum)
);
```

Functional Dependencies:

serialNum, policyNum → paymentDate

```
5   SELECT * FROM Payment;
```

Data Output | Explain | Messages | History

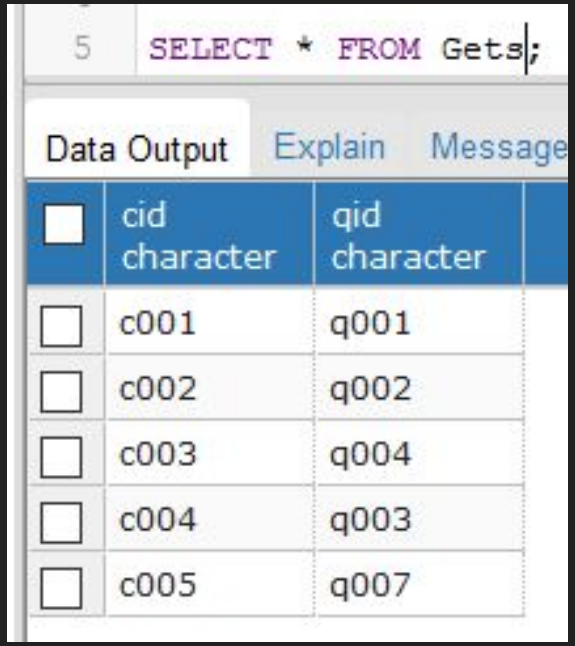
serialnum integer	paymentdate character	policynum integer
12345	10-10-2016	1001
12346	10-10-2016	1002
12347	10-11-2016	1003
12348	10-11-2016	1004
12349	10-12-2016	1005

Gets Table

```
CREATE TABLE Gets (
        cid             char(6) references Clients(cid),
        qid             char(7) references Quote(qid),
PRIMARY KEY(cid, qid)
);
```

Functional Dependencies:

Clients cid, Quote qid → Gets cid, Gets qid



Report Table

```
CREATE TABLE Report (
        reportNum                       integer not null unique,
        reportDate                      char(20) not null unique,
        reportDescription       char(40) not null unique,
        claimID                         char(12) not null unique references Claim(claimID),
        status                          text,
PRIMARY KEY(reportNum, reportDate, reportDescription, claimID)
);
```

Sample Data →

```
5   SELECT * FROM Report;
```

Data Output | Explain | Messages | History

reportnum integer	reportdate character	reportdescripti... character	claimid character
100	10-10-2016	description-1	claim-100
101	10-11-2016	description-2	claim-101
102	10-12-2016	description-3	claim-102
103	10-13-2016	description-4	claim-103

Functional Dependencies:

reportNum, reportDate, reportDescription, claimID → status

WorksFor Table

```
CREATE TABLE WorksFor (
        aName        char(20) references Agency(aName),
        icName       char(20) references InsuranceCompany(icName),
PRIMARY KEY(aName,icName)
);
```

Sample Data →

```
5   SELECT * FROM WorksFor;
```

Data Output | Explain | Messages | Hist

aname character	icname character
Pierson	Alan Insurance
Mendleton	Geico Insurance
Jennings	RichIn Insurance
Milford	Allstate Insurance

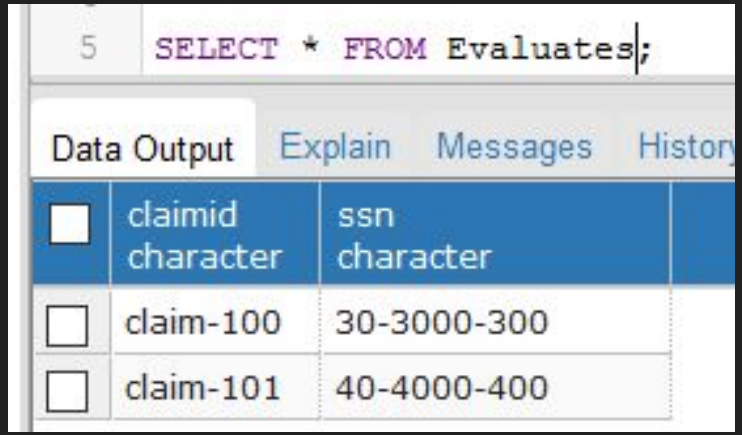
Functional Dependencies:

Agency aName, InsuranceCompany icName → WorksFor (aName, icName)

Evaluates Table

```
CREATE TABLE Evaluates (
        claimID                 char(12) references Claim(claimID),
        SSN                     char(11) references Adjuster(SSN),
PRIMARY KEY(ClaimID, SSN)
);
```

Sample Data →



Functional Dependencies:

Claim (claimID), Adjuster (adjSSN) → claimID, SSN

OfferCom Table

```
CREATE TABLE OffersCom (
        cRate           float(8),
        aName           char(20) references Agency(aName),
        coverage        char(40) references InsuranceProducts(coverage),
        limits          char(40) references InsuranceProducts(limits),
        code            char(20) references InsuranceProducts(code),
        icName          char(20) references InsuranceProducts(icName),
primary key(cRate, aName, coverage, limits, code, icName)
);
```

Sample Data →

```
5  SELECT * FROM OffersCom;
```

Data Output | Explain | Messages | History

crate real	aname character	coverage character	limits character	code character	icname character
50	Pierson	cov-50	lim-20	c000	Alan Insurance
51	Mendleton	cov-51	lim-21	c001	Geico Insurance
52	Jennings	cov-52	lim-22	c002	RichIn Insurance

Functional Dependencies:

cRate, aName, coverage, limits, code, icName → itself (no functional dependency)

Sells Table

```
CREATE TABLE Sells (
        aName       char(20) references Agency(aName),
        coverage    char(40) references InsuranceProducts(coverage),
        limits      char(40) references InsuranceProducts(limits),
        code        char(20) references InsuranceProducts(code),
        icName      char(20) references InsuranceProducts(icName),
PRIMARY KEY(aName, coverage, limits, code, icName)
);
```

Sample Data →

```
5  SELECT * FROM Sells;
```

Data Output | Explain | Messages | History

aname character	coverage character	limits character	code character	icname character
Pierson	cov-50	lim-20	c000	Alan Insurance
Mendleton	cov-51	lim-21	c001	Geico Insurance
Jennings	cov-52	lim-22	c002	RichIn Insurance

Functional Dependencies:

aName, coverage, limits, code, icName →

Refund Table

```
CREATE TABLE Refund (
refundDate              char(20) not null unique,
amount                  float(8),
reportDate              char(20) not null uniquereferences Report(reportDate),
reportDescription       char(40) not null uniquereferences Report(reportDescription),
claimID                 char(12) not null uniquereferences Report(claimID),
reportNum               integer not null unique references Report(reportNum),
accountantSSN           char(7) references Accountant(accountantSSN),
primary key(refundDate, reportDate, reportDescription, claimID, reportNum)
);
```

Sample Data →

```
5  SELECT * FROM Refund;
```

Data Output | Explain | Messages | History

refundd... character	amount real	reportda... character	reportde... character	claimid character	reportnum integer	ssn character
10-20-20...	150	10-10-20...	descripti...	claim-100	100	20-2000-200
10-21-20...	200	10-11-20...	descripti...	claim-101	101	10-2000-200
10-22-20...	250	10-12-20...	descripti...	claim-102	102	20-2000-200
10-23-20...	225	10-13-20...	descripti...	claim-103	103	10-2000-200

Functional Dependencies:

refundDate, reportDate, reportDescription, claimID, reportNum → amountaccountantSSN

Generates Table

```
CREATE TABLE Generates (
        claimID                 char(12) references Report(claimID),
        adjSSN                  char(11) references Adjuster(adjSSN),
        reportNum               integer references Report(reportNum),
        reportDate              char references Report(reportDate),
        reportDescription       char(40) references Report(reportDescription),
primary key(claimID, adjSSN, reportNum, reportDate, reportDescription)
);
```

Sample Data →

```
5  SELECT * FROM generates;
```

Data Output | Explain | Messages | History

ssn character	reportnum integer	reportda... character	reportdescription character	claimid character
30-3000-300	100	10-10-20...	description-1	claim-100
40-4000-400	101	10-11-20...	description-2	claim-101
30-3000-300	102	10-12-20...	description-3	claim-102
40-4000-400	103	10-13-20...	description-4	claim-103

Functional Dependencies:

claimID, adjSSN, reportNum, reportDate, reportDescription → itself

RefundRelatedTo Table

```
CREATE TABLE RefundRelatedTo (
        claimID                 char(12) references Refund(claimID),
        phcID                   char(6) references InsuredPerson(cid),
        insuredPersonName       char(12) references InsuredPerson(name),
        refundDate              char(20) references Refund(refundDate),
        reportDate              char(20  references Refund(reportDate),
        reportDescription       char(40) references Refund(reportDescription),
        reportNum               integer references Refund(reportNum),
PRIMARY KEY(phcID, insuredPersonName, refundDate, claimID , reportNum, reportDate, reportDescription)
);
```

Sample Data →

```
5  SELECT * FROM RefundRelatedTo;
```

Data Output | Explain | Messages | History

claimid character	cid character	insuredpersonn... character	refundd... character	reportdate character	reportdescription character	reportnum integer
claim-100	c001	Alan Labouseur	10-20-20...	10-10-2016	description-1	100
claim-101	c019	Professor Alan	10-21-20...	10-11-2016	description-2	101
claim-102	c002	Alex Richin	10-22-20...	10-12-2016	description-3	103
claim-103	c017	Patrick Trainor	10-23-20...	10-13-2016	description-4	102

Functional Dependencies:

phcID, insuredPersonName, refundDate, claimID, reportNum, reportDate, reportDescription →

ClaimRelatedPersons Table

```
CREATE TABLE ClaimRelatedPersons (
        claimID                 char(12) references Claim(claimID),
        phcID                   char(6) references InsuredPerson(phcID),
        insuredPersonName       char(12) references InsuredPerson(name),
primary key(claimID, phcID, insuredPersonName)
);
```

Note:
phcID → cid

```
5   SELECT * FROM claimrelatedpersons;
```

Data Output | Explain | Messages | History

claimid character	cid character	insuredpersonname character
claim-100	c001	Alan Labouseur
claim-101	c002	Alex Richin
claim-103	c004	John Smith
claim-104	c010	George Wright

Sample Data →

Functional Dependencies:

claimID, phcID, insuredPersonName → itself (no functional dependencies)

StoreManager Table

```
CREATE TABLE StoreManager (
        aName                   char(20) NOT NULL REFERENCES Store(aName),
        storeNum                integer REFERENCES Store(storeNum),
        agentNum                integer REFERENCES Agent(agentNum),
        totalHoursWorked        numeric,
PRIMARY KEY(aName, storeNum, agentNum)
);
```

Sample Data →

```
5  SELECT * FROM storemanager;
```

Data Output | Explain | Messages | History

aname character	storenum integer	agentnum integer
Milford	100	101
Mendleton	102	104
Pierson	104	106
Richard Co.	103	102

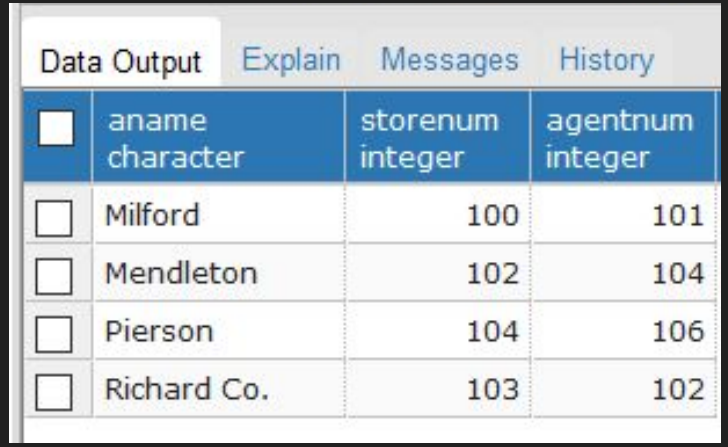
Functional Dependencies:

aName, storeNum, agentNum → totalHoursWorked

Views

Views

```
Create or replace view QuoteProducts as
select IP.coverage, IP.limits, IP.code, IP.insuranceType, IP.icName
from InsuranceProducts IP
inner join sells S
        on (IP.coverage = S.coverage
                and IP.limits = S.limits
                and IP.code = S.code
                and IP.icName = S.icName);
```



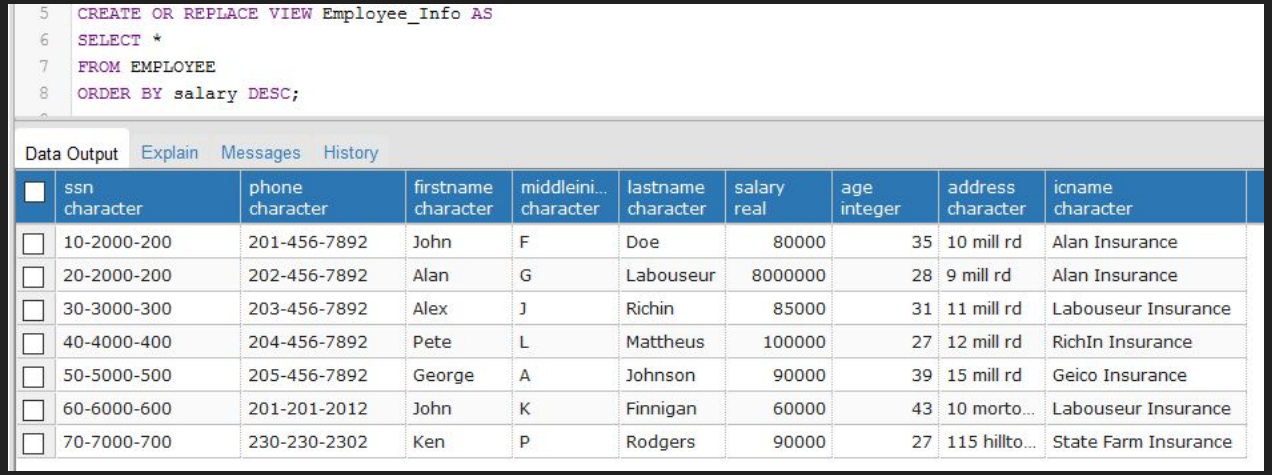
Data Output | Explain | Messages | History

aname character	storenum integer	agentnum integer
Milford	100	101
Mendleton	102	104
Pierson	104	106
Richard Co.	103	102

-- All products relative to their quotes will be displayed. --

Employee Information View

```
CREATE OR REPLACE VIEW Employee_Info AS
SELECT *
FROM EMPLOYEE
ORDER BY salary DESC;
```



```
5  CREATE OR REPLACE VIEW Employee_Info AS
6  SELECT *
7  FROM EMPLOYEE
8  ORDER BY salary DESC;
```

Data Output | Explain | Messages | History

ssn character	phone character	firstname character	middleini... character	lastname character	salary real	age integer	address character	icname character
10-2000-200	201-456-7892	John	F	Doe	80000	35	10 mill rd	Alan Insurance
20-2000-200	202-456-7892	Alan	G	Labouseur	8000000	28	9 mill rd	Alan Insurance
30-3000-300	203-456-7892	Alex	J	Richin	85000	31	11 mill rd	Labouseur Insurance
40-4000-400	204-456-7892	Pete	L	Mattheus	100000	27	12 mill rd	RichIn Insurance
50-5000-500	205-456-7892	George	A	Johnson	90000	39	15 mill rd	Geico Insurance
60-6000-600	201-201-2012	John	K	Finnigan	60000	43	10 morto...	Labouseur Insurance
70-7000-700	230-230-2302	Ken	P	Rodgers	90000	27	115 hillto...	State Farm Insurance

Views

```
CREATE OR REPLACE VIEW Large_premium AS
SELECT c.storeNum,c.storeName, c.lvl1, c.sPhone, c.aName, c.street, c.city, c.state, c.zip
FROM InsuranceProducts m, WorksFor n,
(SELECT storeNum, storeName, lvl1, sPhone, aName, street, city, state, b.zip
FROM Store a
INNER JOIN
StoreZip b on a.zip=b.zip) c, policy p
WHERE
m.icName=n.icName
AND c.aName=n.aName
AND p.coverage=m.coverage
AND p.code=m.code
AND p.limits=m.limits
AND p.icName=m.icName
AND p.finalPremium >= 200
```

Not enough data to have any result…. Query was successful in creating

Views

```
CREATE OR REPLACE VIEW Valid_Refunds as
SELECT p.policyNum,
            pDate,
            finalPremium,
            policyTerms,
            p.cid,
            coverage,
            limits,
            code,
            icName
FROM Refund r
        INNER JOIN Claim c
                ON c.claimID = r.claimID
                        INNER JOIN policy p on p.policyNum = c.policyNum
WHERE
        year IN (Select year FROM refundDate = '10-20-2016');
```

--Valid_Refunds: This view is designed to return all policies that are relative to any particular refunds issued to clients in the year of 2011.

Views

```
CREATE OR REPLACE VIEW Client_Online_Quotes AS
SELECT c.cid,  clientName, age, address,  cp.phone,  email
FROM Clients c
        INNER JOIN Gets g ON g.cid = c.cid
                INNER JOIN Quote q ON q.qid = g.qid
                        INNER JOIN clientPhone cp ON cp.cid = c.cid
WHERE q.qid IN ( SELECT qid
                            FROM OnlineQuote);
```

Query created succesfully, not enough corresponding data to output*

Stored Procedures

Stored Procedure

```
create or replace function  Client_in_Policy(cid char(6), resultset refcursor) returns refcursor
as
$$
declare
  --
begin
  open resultset for select policyNum
                     from PolicyHolder
                     Where Clients.cid = PolicyHolder.cid
  return resultset;
end;
$$ language plpgsql
```

Reports

Who are all the clients who live at Marist?

Query:

```
SELECT cid, clientName

FROM Clients

WHERE upper(address) like upper('3399 N Rd');
```

Sample Data Output →

```
6  Select cid, clientName
7  FROM Clients
8  WHERE upper(address) like upper('3399 N Rd');
```

Data Output | Explain | Messages | History

cid character	clientname character
c001	Alan Labouseur

Retrieve the names and Tax IDs of all agencies who sell at least one type of product:

DATA OUTPUT →

```
4  Select a.aName, tid, count(insuranceType)
5  From agency a, sells w, InsuranceProducts p
6  Where a.aName = w.aName
7  And p.icName = w.icName
8  group by a.aName, tid
9  having count(insuranceType)=1;
```

Data Output | Explain | Messages | History

aname character	tid character	count bigint
Jennings	t008	1
Mendleton	t005	1
Pierson	t009	1

QUERY:

```
SELECT a.aName, tid, COUNT(insuranceType)
FROM agency a, sells w, InsuranceProducts p
WHERE a.aName = w.aName AND p.icName = w.icName
GROUP BY a.aName, tid
HAVING COUNT(insuranceType)=1;
```

Which PolicyHolders have filed a claim already?

Data Output →

```
4  select cl.clientName, cl.address, cp.phone
5  from Clients cl, clientPhone cp
6  where  cl.cid = cp.cid and
7  exists (select 1
8      from claimRelatedPersons cl inner join claimRelatedPersons cp
9          on cl.insuredPersonName = cp.insuredPersonName
10     where cl.cid <> cp.cid and cl.cid = cp.cid);
11
```

Data Output | Explain | Messages | History

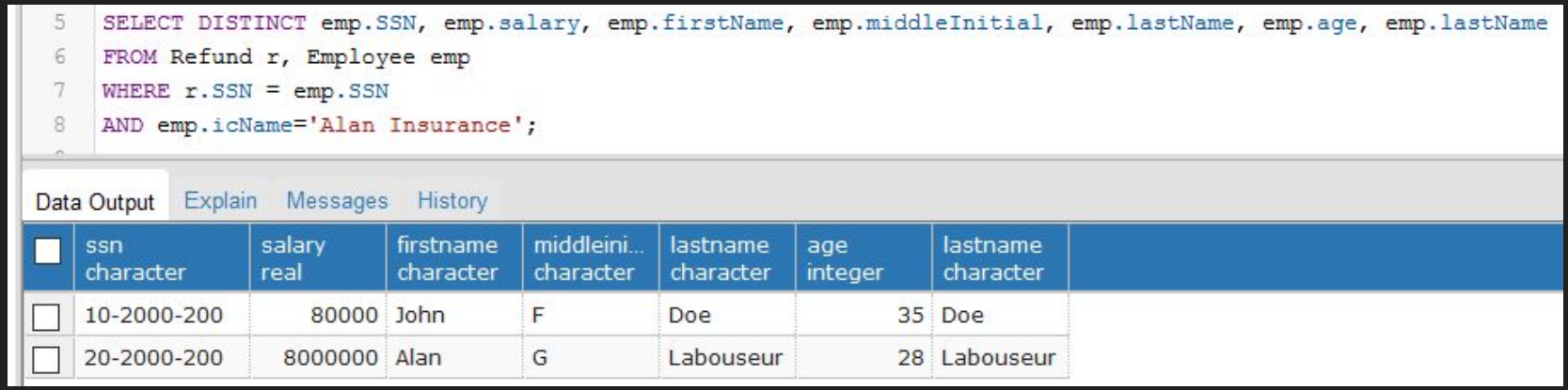
aname character	coverage character	limits character	code character	icname character
Pierson	cov-50	lim-20	c000	Alan Insurance
Mendleton	cov-51	lim-21	c001	Geico Insurance
Jennings	cov-52	lim-22	c002	RichIn Insurance

Query:

```
SELECT cl.clientName, cl.address, cp.phone
FROM Clients cl, clientPhone cp
WHERE  cl.cid = cp.cid and
EXISTS (SELECT 1
        FROM claimRelatedPersons cl INNER JOIN claimRelatedPersons cp
                ON cl.insuredPersonName = cp.insuredPersonName
        WHERE cl.cid <> cp.cid AND cl.cid = cp.cid);
```

Retrieve the name, salary, SSN, and age from employees who works on a claim as an accountant for a particular insurance company:

```
SELECT DISTINCT emp.SSN, emp.salary, emp.firstName, emp.middleInitial, emp.lastName, emp.age, emp.lastName
FROM Refund r, Employee emp
WHERE r.SSN = emp.SSN
AND emp.icName = 'Alan Insurance';
```



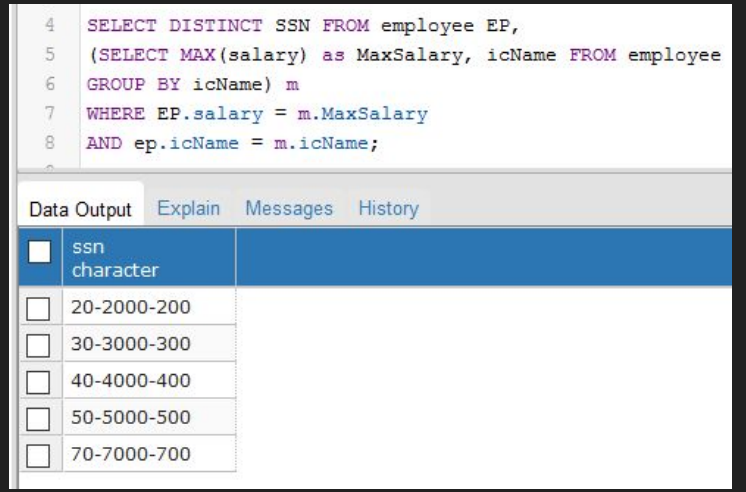
```
5  SELECT DISTINCT emp.SSN, emp.salary, emp.firstName, emp.middleInitial, emp.lastName, emp.age, emp.lastName
6  FROM Refund r, Employee emp
7  WHERE r.SSN = emp.SSN
8  AND emp.icName='Alan Insurance';
```

Data Output | Explain | Messages | History

ssn character	salary real	firstname character	middleini... character	lastname character	age integer	lastname character
10-2000-200	80000	John	F	Doe	35	Doe
20-2000-200	8000000	Alan	G	Labouseur	28	Labouseur

This query will return employees(distinct) with the largest salary.

```
SELECT DISTINCT SSN FROM employee EP,
(SELECT MAX(salary) as MaxSalary, icName FROM employee
GROUP BY icName) m
WHERE EP.salary = m.MaxSalary
AND ep.icName = m.icName;
```



This query will return clients who have a discounted rate higher than that of 5%.

```
select count(cid)
from discount d , gets g, policy p
where
p.phClientID = g.cid
and d.qid=g.qid
and rate > 5;
```

Security

Security(1)

CREATE ROLE Admin;

GRANT SELECT, INSERT, UPDATE ON Policy
TO Admin;

Admins can change the policy details only.

CREATE ROLE Actuator;

GRANT Select, Insert ON Actuary;

TO Actuator;

Security(2)

CREATE ROLE Client;

GRANT SELECT ON Policy

TO Client;

Clients can only view their policies, not change or delete them.

Security(3)

CREATE ROLE Adjuster;

GRANT Insert, Update ON Claim;

TO Adjuster;

CREATE ROLE Accountant;

GRANT INSERT, UPDATE ON Accountant(status);

TO Accountant;

Triggers

Trigger:

```
CREATE TRIGGER ValidInfo
BEFORE insert OR update ON Clients
EXECUTE PROCEDURE validInfo();
```

Conclusion

Implementation Notes / Conclusion

We have used specialization for Employee, Client and Quote below.
We adopted the option which works for any specialization (total or partial, disjoint or overlapping).
Since all employees can be divided into three parts: Accountant, Actuary, and Adjuster, we have made them as subclasses of Employee and have used the overlapping constraint as they may play more than one role at the same time.
We created separate relations for Accountant, Actuary, and Adjuster.
Since some clients may accept the quote and then become Policy holder so we have made Policy Holder as a sub class of the Client and have used subset notation to show this in the ER diagram.

Future Enhancements

Some possible future enhancements of this database are...

Clients who have more than 3 insurance are considered as the Priority customers.

Client can not have more than one claim for the same ailment.
Employees can not work as adjuster/accountant claims which belong to themselves or their dependent/s.
Employee’s manager’s approval is needed in case belongs to any of his/her employee or their dependent/s to avoid conflict of interest.
Policy can not have more than 10 payments.

Problems

It was difficult to decipher between SSN for some tables such as accountant, actuator, or adjuster. This is because the SSN originates from the employee table and there is no clear way to decipher which SSN is for which table.

Also had some difficulty getting everything to follow foriegn / primary key references. Had to go back and classify which table’s was unique or not. It was very time consuming to implement this database and at times it is hard to follow.

Conclusion

We have summarized all the necessary descriptions and solutions for Insurance management system database. This includes the process and result of the ER diagram, relational schemas in third normal form, SQL statements to create database, create view and solve corresponding queries. We have also tested each of the queries in a database environment to check for correctness. We also explained why we use superclass/subclass relationships to build the relational schema.