**DB DESIGN PROJECT**

**TOPIC:**

**INSURANCE MANAGEMENT SYSTEM**

**Course Instructor**

Dr. Soon M. Chung

**Submitted By**

Hima Sekhar Reddy Devarinti

UID: U00955448

**1. Problem Description**

Design, implement, and test ***Insurance Management System*** that provides services to clients (policy-holders) and insurance companies based on a relational database.

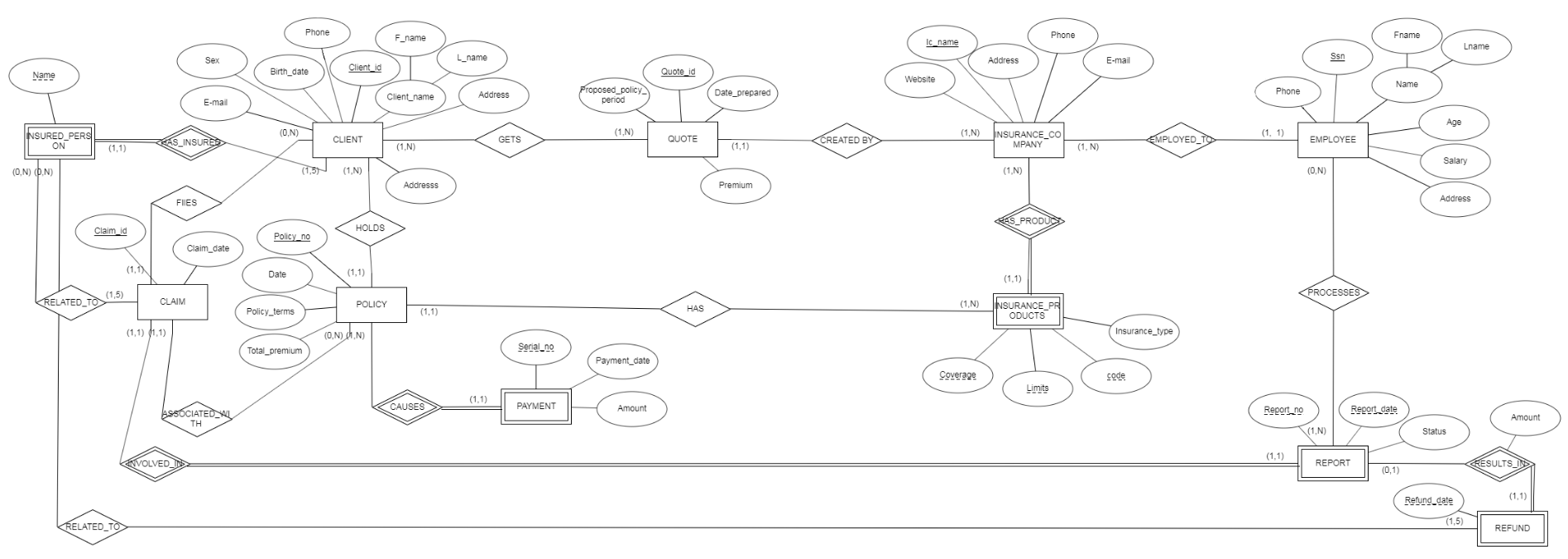
Data Requirements

1. Each client contains such information: Unique Client id, Name, Birthdate, Sex, Address, Phone Number (one or more phone numbers) and Email-id.
2. A client can get insurance quotes and file claims more than once.
3. Clients can get quotes on their insurance from the insurance companies.

Each quote, which has the attributes Unique Quote ID, Date Prepared, Proposed Policy Period and Premium, is issued for one particular client.

1. Some clients in the system may file claims including Unique Claim ID, Insured Person (one or more), and Claim Date. Each claim must associate with an exact one policy.
2. Once a client accepts a quote, he/she would hold a policy based on the accepted quote. One client may have at most five insured persons who only have Name (may not be unique) recorded in the system. Moreover, each policy has a Unique Number, Date, Final Premium and Policy Terms. One policy must have one or more payments. For each payment, Serial Number (may not be unique), Date, and Amount are recorded, whereas Serial Number, Payment Date, and the policy determine one particular payment.
3. An insurance company offers three types (auto, life, and home) of insurance products, and has the following information: Unique Company name, Website, Address, Email and Phone Number.
4. Each product contains Type, Code, Coverage and Limits; whereas Code, Coverage, Limits and the insurance company associated uniquely determine one specific product
5. Employees are hired by insurance companies with the following information: Unique SSN, Name (Fname and Lname), Age, Address, Phone Number, and Salary.
6. Employees who work for insurance companies evaluate the filed claims, and then process final reports respectively with the attributes Report Number, Report Date and Description. Report Number, Report Date and the claim\_id determine one unique report. Some reports may have a refund that is an aggregation of Refund-Date, Amount, and Name of Related Person. Each refund is uniquely decided by a specific report and Refund-Date.

**2. E-R DESIGN**



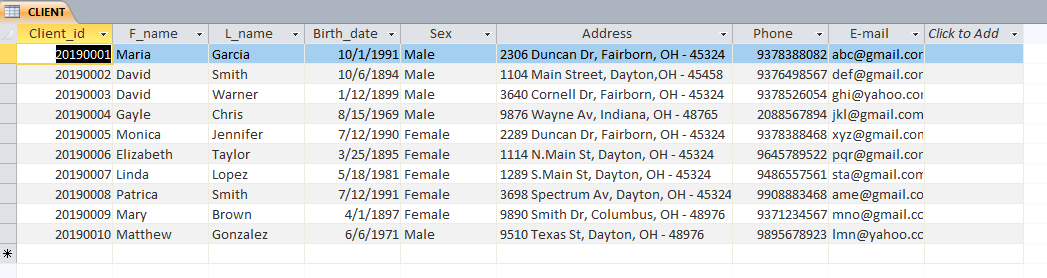
**3.**

We inserted some values into the database in order to create and test typical SQL query statements. All the relations are created using MS Access.

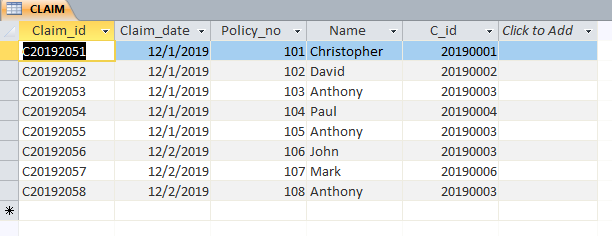
Here are the snapshots that shows the data in the tables.

**Below are the screenshots of the Database state:**

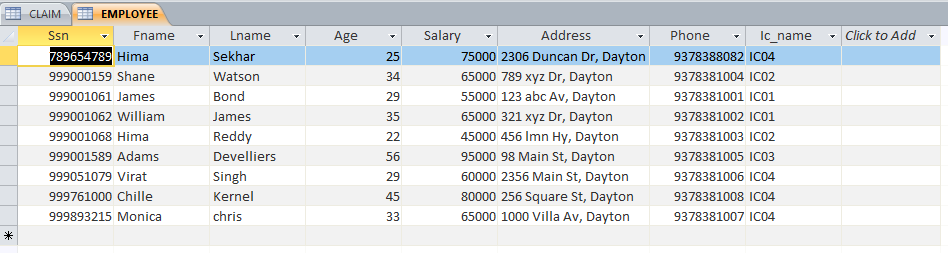
**CLIENT**



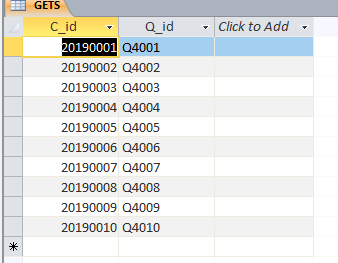
**CLAIM**



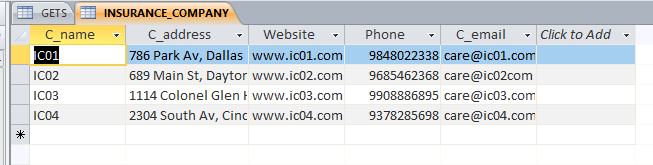
**EMPLOYEE**



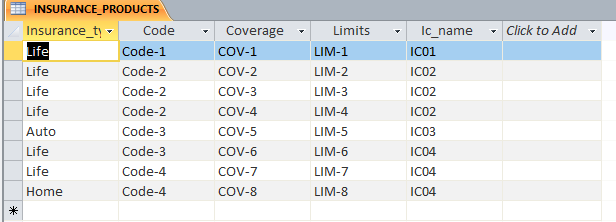
**GETS**



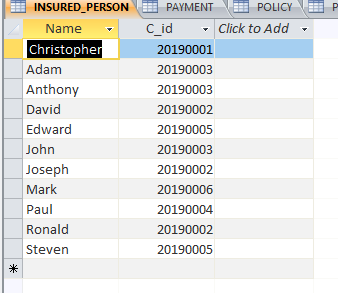
**INSURANCE\_COMPANY**



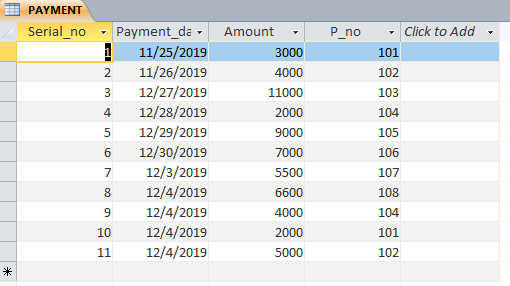
**INSURANCE\_PRODUCTS**



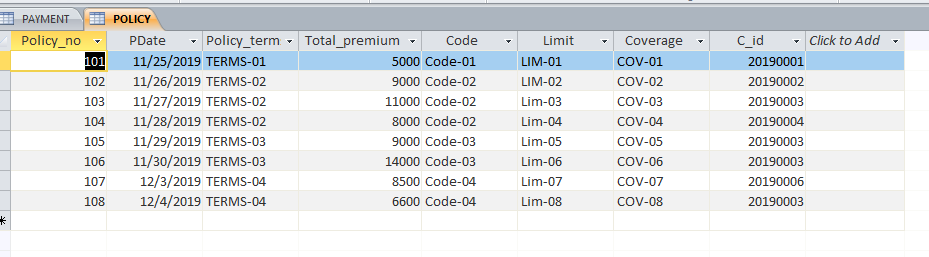
**INSURED\_PERSON**



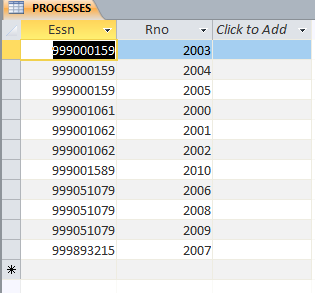
**PAYMENT**



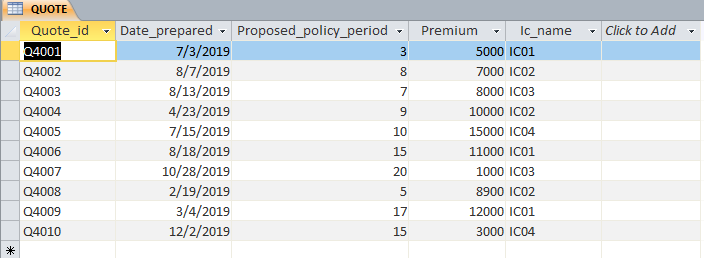
**POLICY**



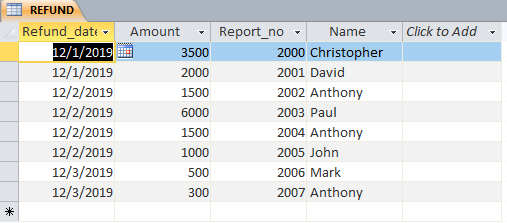
**PROCESSES**



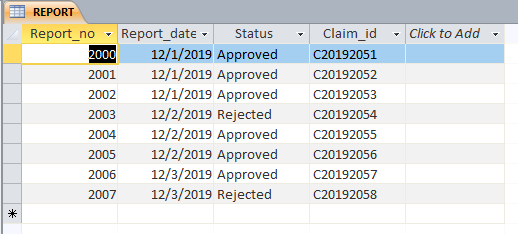
**QUOTE**



REFUND



**REPORT**



**4. SQL Queries and Results**

**1) Retrieve the IDs and names of all clients who live in Dayton**

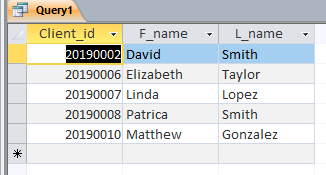
QUERY

SELECT Client\_id,F\_name,L\_name

FROM CLIENT

WHERE Address like '\*Dayton\*' ;

RESULT



**2) List the Name and address of the employee who works on the claim reports which are not approved.**

QUERY

SELECT E.Fname, E.Lname, E.Address

FROM EMPLOYEE AS E

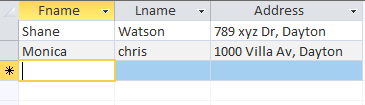
WHERE E.Ssn IN

(SELECT Essn

FROM (REPORT INNER JOIN PROCESSES ON REPORT.Report\_no = PROCESSES.Rno)

WHERE Status <> 'Approved' );

RESULT



**3) Retrieve the names and phone numbers of the clients who didn’t add Insured persons (Client who have no persons in their insurance policy)**

QUERY

SELECT F\_name,L\_name,Phone

FROM CLIENT

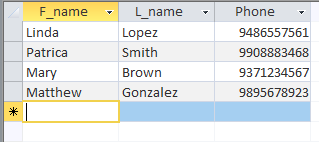
WHERE NOT EXISTS

(SELECT \*

FROM INSURED\_PERSON

WHERE Client\_id = C\_id) ;

RESULT



Here the correlated nested query retrieves all the INSURED\_PERSON tuples related to the CLIENT tuple (Client\_id) being evaluated. If NONE exists then CLIENT tuple is selected.

**4) List the name, address and website details of the insurance company along with the employees count for each company.**

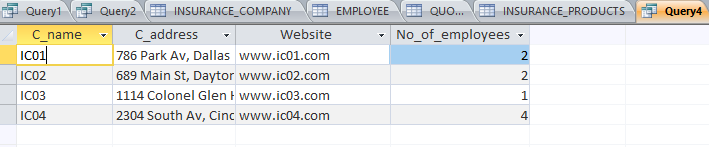
QUERY

SELECT C\_name, C\_address,Website, count(\*) as No\_of\_employees

FROM (EMPLOYEE INNER JOIN INSURANCE\_COMPANY ON EMPLOYEE.Ic\_name = INSURANCE\_COMPANY.C\_name)

GROUP BY C\_name, C\_address, Website ;

RESULT



The above result can also be obttained by using the following query

SELECT C\_name,C\_address,Website, count(\*) as No\_of\_employees

FROM EMPLOYEE, INSURANCE\_COMPANY

WHERE EMPLOYEE.Ic\_name = INSURANCE\_COMPANY.C\_name

GROUP BY C\_name, C\_address, Website ;

Here, the grouping and aggregate functions are applied after the two relations are joined.

C\_address and website are included in the GROUP BY clause, even though C\_name is enough to group the tuples, because the SELECT clause can include only grouping attributes and aggregate functions (when the GROUP BY clause is used).

**5) Retrieve the company names where it has provided employment to atleast 2 employees.**

QUERY

SELECT C\_name, count(\*) as No\_of\_employees

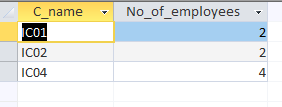
FROM EMPLOYEE AS E, INSURANCE\_COMPANY AS I

WHERE E.Ic\_name = I.C\_name

GROUP BY C\_name, C\_address, Website

HAVING Count(\*) >=2 ;

RESULT



The HAVING clause is used here for specifying a selection condition on Company name group (rather than on individual tuples).

6) **Retrieve the names and phone numbers of all the clients who have not fully paid for the policies held by them. (Hint: Total Premium > the sum of payments captured in Payment relation)**

QUERY

SELECT Cl.F\_name,Cl.L\_name,Cl.Phone

FROM CLIENT Cl, POLICY Py,

(SELECT P\_no, sum(Amount) as Total\_pay

FROM PAYMENT

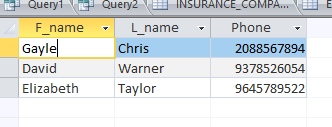
GROUP BY P\_no) Pa

WHERE (Py.Policy\_no = Pa.P\_no) AND

(Py.Total\_premium >Pa.Total\_pay) AND

(Py.C\_id = Cl.CLient\_id);

RESULT



**7) Retrieve the Average age of all the employees who work for the company ‘IC04’**

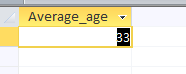
QUERY

SELECT DISTINCT AVG(Age) AS Average\_age

FROM EMPLOYEE AS E

WHERE Ic\_name = 'IC04' ;

OUTPUT



AVG aggregate function is used to obtain the average age of the employees

**8) Make a list of the client id’s whose quoted premium is greater than 10000 or whose policy total premium is greater than 10000**

QUERY

SELECT DISTINCT Client\_id

FROM CLIENT C

WHERE C.Client\_id IN

((SELECT Client\_id

FROM CLIENT,QUOTE,GETS G

WHERE Premium > 10000 AND Quote\_id = Q\_id AND C\_id = Client\_id)

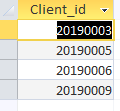
UNION

(SELECT Client\_id

FROM CLIENT,POLICY

WHERE Total\_premium > 10000 AND C\_id = Client\_id)) ;

RESULT



Here, OR operator can also be used in the place of UNION to obtain the same result.

**9) Retrieve the last name and first name of employees whose salary is greater than the salary of all the employees in the company IC01**

QUERY

SELECT Lname,Fname

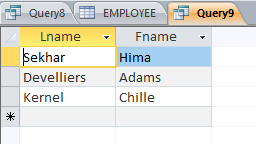
FROM EMPLOYEE

WHERE Salary > ALL ( SELECT Salary

FROM EMPLOYEE

WHERE Ic\_name = 'IC01') ;

RESULT



**10 ) Retrieve the names of clients, client Id’s and Quote id’s , ordered by the insurance company, and within each company order the first names of the clients in alphabetically reverse order.**

QUERY

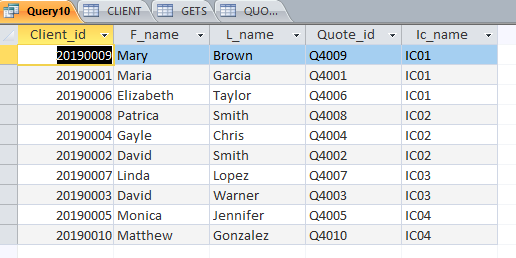
SELECT C.Client\_id,C.F\_name,C.L\_name,Q.Quote\_id,Q.Ic\_name

FROM CLIENT C, QUOTE Q, GETS G

WHERE Q.Quote\_id = G.Q\_id AND G.C\_id = C.Client\_id

ORDER BY Q.Ic\_name,C.F\_name DESC;

OUTPUT



The ORDER BY keyword is used to sort the F\_name in descending order

**11 ) Print the resulting salaries if every employee working for the 'IC04' company is given a 25% salary raise.**

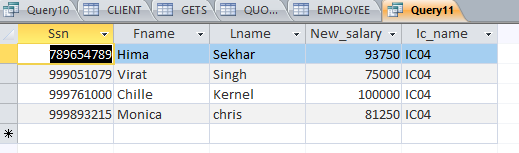
QUERY

SELECT Ssn, Fname, Lname, 1.25\*Salary AS New\_salary, Ic\_name

FROM EMPLOYEE

WHERE Ic\_name = 'IC04';

OUTPUT



FORMULAE:

Salary+(Salary\*25)/100 = salary+salary\*0.25 = 1.25 \* salary

**5. Discussion**

1. In this design, clients are getting their quotes on insurance from the insurance company websites only. Instead, we can also provide an offline option for the clients by integrating the agencies.
2. Agencies contain multiple stores, where clients can obtain their quotes. Each agency sells insurance products provided by one or more insurance companies.
3. Insurance Company employees authenticate the claims filed by the clients in the design discussed in this project. There is a possibility that employees can work on insurance claims belong to themselves or their dependents. So we can implement a manager for a group of people and the 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.
4. While implementing the project, I learned how to use wildcard operators effectively in the queries to search or compare the substrings.
5. Employees entity can be subdivided into two namely Evaluator who works on the claim reports and Accountant who works on the settlement of the claims once it is approved by the evaluator. In this way, we can maintain transparency by having managers who control these groups of employees.
6. Gained practical knowledge about Nesting of queries (both correlated and uncorrelated), Join functions, Aggregate functions, Exists and IN operators, Grouping, Order by, having, etc... I also implemented these functions in the sample database successfully in this project.
7. Tried to implement the discount option for the insurance policies but failed to do this, since I didn’t mention these criteria in the data requirements and there should be some parameter to evaluate whether the client is eligible for avail the discount or not. The criteria could be the client who is having more than 4 insurance policies with the same company and claimed amount should be less than 20% of his/her total premium.
8. There is scope for the implementation of Auto and House insurance by adding more features into the current E-R design.