

# **Web Based Application for Insurance Services**

## **A PROJECT REVIEW FINAL REPORT**

*Submitted by*

**19BCE2599 Tenzin Khorlo**

**19BCE2600 Lhendup Dorji**

**19BCE2607 Pritam Chaurasiya**

**19BCE2609 Rishi Srikaanth**

*CSE 2004 Database Management Systems*

*Guided by*

**Professor Pradeepa M**

*Assistant Professor Sr. Grade 1*

*School of Information Technology and Engineering*



**VIT<sup>®</sup>**

---

**Vellore Institute of Technology**

(Deemed to be University under section 3 of UGC Act, 1956)

**Fall Semester 2020-21**

## **Abstract**

The present system of the insurance companies is characterized by the manual method as a result serious threat has been posed to the operation of the service and too much workload on the staffs. The manual method involves the marketing staffs moving from one location to the other to meet up with the requirement of their broker and also the files and data of their broker are stored in cabinet which are easily destroyed by rodents.

With regards to this method the insurance computer application for insurance company would be developed this would have the ability to remotely connect insurance brokers in any location for them to carry out their insurance services and also their data would be stored in a secured database. In the software design, to achieve this task we intend to use the Oracle SQL as database and use java IDE Eclipse to make the app.

**LIST OF TABLES:****Page no:**

▪ <b>Table 2.1 Review of Literature.....</b>	<b>7-8</b>
▪ <b>Table 6.1: Agent1 .....</b>	<b>25</b>
▪ <b>Table 6.2 : Agent2.....</b>	<b>25</b>
▪ <b>Table 6.3 : Policy.....</b>	<b>26</b>
▪ <b>Table 6.4 : Policy_Holder1 .....</b>	<b>27</b>
▪ <b>Table 6.5 : Policy_Holder2.....</b>	<b>27</b>
▪ <b>Table 6.6 : Claimant1.....</b>	<b>28</b>
▪ <b>Table 6.7 : Claimant2.....</b>	<b>28</b>
▪ <b>Table 6.8 : Sales.....</b>	<b>29</b>
▪ <b>Table 6.9 : Insured_by1.....</b>	<b>30</b>
▪ <b>Table 6.10 : Insured_by2A.....</b>	<b>30</b>
▪ <b>Table 6.11 : Insured_by2B.....</b>	<b>30</b>
▪ <b>Table 6.12 : Claimed_by.....</b>	<b>31</b>
▪ <b>Table 6.13 : UserLogin.....</b>	<b>32</b>
▪ <b>Table 6.14 : AdminLogin.....</b>	<b>32</b>

## LIST OF FIGURES

Page no:

Figure 5.1 : ER Diagram.....	13
Figure 5.2 : Entities Relations Model.....	14
Figure 5.3a : Normalization part 1.....	15
Figure 5.3b : Normalization part 2.....	16
Figure 5.3c : Normalization part 3.....	17
Figure 5.3d : Normalization part 4.....	18
Figure 5.3e : Normalization part 5.....	19
Figure 5.3f : Normalization part 6.....	20
Figure 5.3g : Normalization part 7.....	21
Figure 5.3h : Normalization part 8.....	22
Figure 6.3i : Normalization part 9.....	23
Figure 6.3j : Normalization part 10.....	24
Figure 6.1 : Creation of table Agent1 and Agent2.....	25
Figure 6.2 : Creation of table Policy.....	26
Figure 6.3 : Creation of table Policy_Holder1 and Policy_Holder2.....	27
Figure 6.4 : Creation of table Claimant1 and Claimant2.....	28
Figure 6.5 : Creation of table Sales.....	29
Figure 6.6 : Creation of table Insured_by1, Insured_by2A and Insured_by2B.....	30
Figure 6.7 : Creation of table Claimed_by.....	31
Figure 6.8 : Creation of table UserLogin and AdminLogin.....	32
Figure 6.9 : Updated values in table Claimant1 and Claimant2.....	33
Figure 6.10 : Updated values in table Agent1, Agent2 and Policy.....	33
Figure 6.11 : Updated values in tables Policy_holder1, Policy_Holder2 and Sales.....	34
Figure 6.12 : Updated values in table Insured_by1, Insured_by2A and Insured_by2B.....	34
Figure 6.13 : Updated values in tables insured_by2B and Claimed_by....	35
Figure 6.14 : Updated values in tables UserLogin and AdminLogin.....	35
Figure 6.15 : Home Page.....	36
Figure 6.16 : Admin Login Form.....	36
Figure 6.17 : Admin Page.....	37
Figure 6.18 : User Details.....	37
Figure 6.19 : All AGENTS DETAILS.....	38
Figure 6.20 : All Sales Info.....	38
Figure 6.21 : All Claimed Insurance Details.....	39
Figure 6.22 : All Policy Details.....	39
Figure 6.23 : Back to Home Page after Sign out from Admin.....	40
Figure 6.24 : User Login Form.....	40
Figure 6.25 : User Page.....	41
Figure 6.26 : User Personal Info.....	41
Figure 6.27 : User's Agent Info.....	42
Figure 6.28 : User's Insurance Details.....	42
Figure 6.29 : Back to Home Page after signing out from User page.....	43
Figure 6.30 : Thank You Page and Our Contributions.....	43

## TABLE OF CONTENTS

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
	<b>ABSTRACT</b>	<b>2</b>
	<b>LIST OF TABLES</b>	<b>3</b>
	<b>LIST OF FIGURES</b>	<b>4</b>
<b>1</b>	<b>1.1 INTRODUCTION</b>	<b>6</b>
<b>2</b>	<b>2.1 REVIEW OF LITERATURE</b>	<b>7-8</b>
<b>3</b>	<b>PROBLEM STATEMENT AND OBJECTIVES</b>	
	<b>3.1 PROBLEM STATEMENT.....</b>	<b>9</b>
	<b>3.2 OBJECTIVES.....</b>	<b>9</b>
<b>4</b>	<b>HARDWARE AND SOFTWARE REQUIREMENTS</b>	
	<b>4.1 HARDWARE REQUIREMENTS.....</b>	<b>10</b>
	<b>4.2 SOFTWARE REQUIREMENTS.....</b>	<b>10</b>
	<b>IMPLEMENTATION</b>	
	<b>5.0 PROPOSED METHODOLOGY.....</b>	<b>11</b>
	<b>5.1 REQUIREMENT ANALYSIS</b>	
	<b>    5.1.1 FOR ACCESSING</b>	<b>11</b>
	<b>        5.1.1.A USER.....</b>	<b>11</b>
	<b>        5.1.1.B ADMIN.....</b>	
	<b>    5.1.2 FOR GETTING POLICY KEY</b>	<b>11</b>
	<b>        5.1.2.A CAR INSURANCE.....</b>	<b>12</b>
	<b>        5.1.2.B HEALTH INSURANCE .....</b>	
	<b>    5.2 ER DIAGRAM</b>	<b>13</b>
	<b>        5.2.1 ER-DIAGRAM.....</b>	<b>14</b>
	<b>        5.2.2 ENTITIES RELATIONS MODEL.....</b>	
	<b>    5.3 NORMALIZATION</b>	<b>15-24</b>
	<b>        5.3.1 NORMALIZATION OF ALL TABLES.....</b>	
	<b>RESULTS AND DISCUSSION</b>	
<b>6</b>	<b>6.1 CREATION OF TABLES IN MY SQL PLUS.....</b>	<b>25-32</b>
	<b>6.2 UPDATED VALUES IN DATABASE.....</b>	<b>33-35</b>
	<b>6.3 IMPLEMENTATION IN ECLIPSE(APP CREATION).....</b>	<b>36-43</b>
	<b>CONCLUSION</b>	
	<b>7.1 CONCLUSION.....</b>	<b>44</b>
	<b>7.2 REFERENCES.....</b>	<b>45</b>
	<b>7.3 APPENDICES.....</b>	<b>46-48</b>

# **CHAPTER 1**

## **1.1 Introduction**

Insurance is a means of protection from financial loss. It is a form of risk management, primarily used to hedge against the risk of a contingent or uncertain loss. The insured receives a contract, called the insurance policy, which details the conditions and circumstances under which the insurer will compensate the insured. The amount of money charged by the insurer to the policyholder for the coverage set forth in the insurance policy is called the premium. If the insured experiences a loss which is potentially covered by the insurance policy, the insured submits a claim to the insurer for processing by a claims adjuster. The insurer may hedge its own risk by taking out reinsurance, whereby another insurance company agrees to carry some of the risks, especially if the primary insurer deems the risk too large for it to carry.

Insurance is an important area of the business service industry. The project is based on implementing a web-based application for insurance services that shows the rates offered by different insurance agencies. The main types of insurance dealt in this project are home insurance, auto insurance, farm insurance, and health insurance. Depending upon the user information, real time quotes are generated from different companies. This project is intended to provide and manage a good customer relationship.

## Chapter 2

### 2.1 Review of literature

In this section, literature survey is given. Accordingly, research papers are reviewed and analyzed based on the prediction methods used:

**Table 2.1 Review of Literature**

Title, Name of Author And Date of Publication, Publisher info	Problem statement and objectives	Their Proposed Methodology	Outcome and Limitations
1) "Implementation of Futuristic Web Based Application for insurance Services" by Ele, Sylvester, <i>Department of Computer Science, University of Calabar, Nigeria.</i> Published in 2018 in American Journal of Engineering Research.	Their project was designed as a means to avert the low insurance penetration in Nigeria by promoting and managing a good customer relationship/experience using our modern day web technology.	The development methodology adopted for the proposed system is a hybrid of Agile development methodology (for development process management) and object oriented analysis and design- OOAD (for analysis and design).	1) He successfully concluded that E-insurance system was efficient, reliable and usable. 2) It was well explained with practical examples. However, explanation for practical output was insufficient and the images of their experiment result was not clear.
2) Web Based Application for Insurance Services Case Study of the Insurance Company by Igwe E.kelvin, who is a software developer, blogger and the CEO of K-Tech System. He is graduate of Computer Science and Engineering. Published date is not mentioned. Both journals can be found this journal in this link: <a href="http://www.academia.edu">www.academia.edu</a> or from Academia App.	1) To design web based application for insurance service. 2) To provide logical programming which is capable of providing insurance policy holders easy access to any kind of service provided by insurance service.	He intends to achieve the task using Microsoft Visual Studio platform and Microsoft Access 2007 as database to store information.	1) He only explained why E-insurance system is needed and scope of it. 2) It provides essential knowledge to beginner. 3) However, it does not provide practical examples and project was incomplete.

<p>3) “Web Based Application For Insurance Services Case Study Of The Insurance Company”, by Esedebe Fidelia Ogechukwu, in July, 2013, a graduate of Computer Science of Information and Technology, Caritas University Amorji – Nike Enugu, Nigeria.</p>	<p>1) To ensure effective insurance service communication around the globe from a remote location using the web application.      2) To promote growth and financial stability of insurance companies and effectively enable policy holders monitor their service around the globe.      3) To professionalize insurance services and develop insurance consciousness among the general populace.</p>	<p>They proposed to use Visual studio 2008 platform and Microsoft access as database.</p>	<p>1) His project detailed and was well explained with practical example.      2) His project was successfully done and output was also displayed.      3) But, there was no ER diagram.</p>
<p>4) “Teach yourself SQL in 21 days, second edition ,” ,by Auwal Gene , available at link : <a href="http://www.auwalgene.com/@mystudents/lecturenotes/teach_urself_sql.pdf">http://www.auwalgene.com/@mystudents/lecturenotes/teach_urself_sql.pdf</a></p>	<p>To Help reader to learn:      1. The process to create table .      2. The various constraints and its implementation.      3.</p>	<p>1. He explains the concepts of classes, objects, properties, methods, and events.      2. He teaches the process of visual program design and development.</p>	<p>1) He explains the process in detail with example making it easier for learners.      2) Recommended for Beginners.      3) However, he uses programming as an example in between which only the ones who know programming can understand.</p>
<p>5) “Web Based Project Ideas &amp; Topics” by nevon Projects, electronics  Software and mechanical kits.      Available in link below:  <a href="https://nevonprojects.com/web-based-project-ideas-topics/">https://nevonprojects.com/web-based-project-ideas-topics/</a></p>	<p>1) To help user learn more about Java Projects using bootstrap,html and c      2) To help learner gain basic knowledge to create web pages and forms.</p>	<p>1) They teach the basics of html, bootstrap.      2) They also teach how to connect database to server.</p>	<p>1) They have given various source codes of different projects with comment line explaining some lines of codes.      2) Not recommended for beginners.</p>

## **Chapter 3**

### **3.1 Problem Statement**

Insurance Companies have been experiencing difficulties trying to attend to their customers quickly and efficiently when required due to lack of adequate documentation of data. Web based application for insurance services has been acknowledged as the fourth site technology that could foster communication of insurance service very quickly and efficiently reaching people around the world irrespective of the location. So we look forward to developing a simple app that can store information of insurance, clients, admin and agent which can benefit the insurance company for keeping records in the long run.

### **3.2 Objectives**

- The aim of this project is based on implementing a web based application for insurance services in other to help insurance companies keep records of every details effectively and thus, have better interaction with clients.
- It is also aimed to promote growth and financial stability of insurance companies and effectively enable policy holders monitor their service with transparency.
- It is aimed to ensure effective insurance service communication around the globe.
- To establish a sound national insurance market; and also add speed to their data processing and retrieving.
- To reduce Data Redundancy and facilitate faster Searching of information.

# **Chapter 4**

## **HARDWARE AND SOFTWARE REQUIREMENTS:**

### **4.1. Hardware Requirements**

- Computer system
- Hard disk size 40 GB and above
- Processor speed of 1.6 GHz
- Ram size of 512 MB and above
- Network connection is required to connect Database with IDE

### **4.2. Software Requirements**

- Windows operating system
- Sql plus oracle(as Database)
- Eclipse as java IDE(Contains JavaScript Source files and classes)
- Ojdbc.jar (Jar File required to connect Database with java IDE)
- Oracle Data Source ( Used to connect to Database) found in ojdbc file.

## **CHAPTER 5**

### **IMPLEMENTATION:**

#### **5.0 Proposed Methodology:**

The proposed system is for making easier to manage policy holder details, agent details, policy details, claimant details and payment details. So this will be developed for managing the insurance management system. The overall system is control through the main menu. The main menu contains 4 parts: “User Login”, “Administrator Login”, “About us”, “Exit”

#### **5.1 Requirement Analysis**

##### **5.1.1 FOR ACCESSING:**

###### **5.1.1.A USER:**

User needs to Log in using their Claimant\_id as their User\_id and enter password which will be stored in database. And they can view their own info, their Insurance details and their agent details

###### **5.1.1.B ADMIN:**

For Admin they login using Pholder\_id as their user\_id and enter password. They can view their own information as well as overall data of users and agents and their contracts of Sales.

##### **5.1.2 TO GET POLICY KEYS AND INSURE**

###### **5.1.2.A CAR INSURANCE:-**

Customer Visits an Agent and asks him to insure his vehicle. Agent takes the customer details and the vehicle details (i.e. RC Book, and vehicle registration papers). Agent forwards the Details to the Company. Company gives the Quotation to the agent for different coverage's. Agent gives the quotation amount to the customer. Customer as per his convenience asks to issue the policy under the coverage of his choice. Agent takes the cheque or cash for the policy from the customer. Agent gives the cheque to the company and request them to issue the policy for the vehicle. Company arranges a check for the vehicle to be insured. If the vehicle is in Condition and passes the test set by the insurance company the vehicle is then insured by the company. The policy copy is then issued to the customer and sent to the mail address. The agent receives the commission as set by the company after the issuance of the policy.

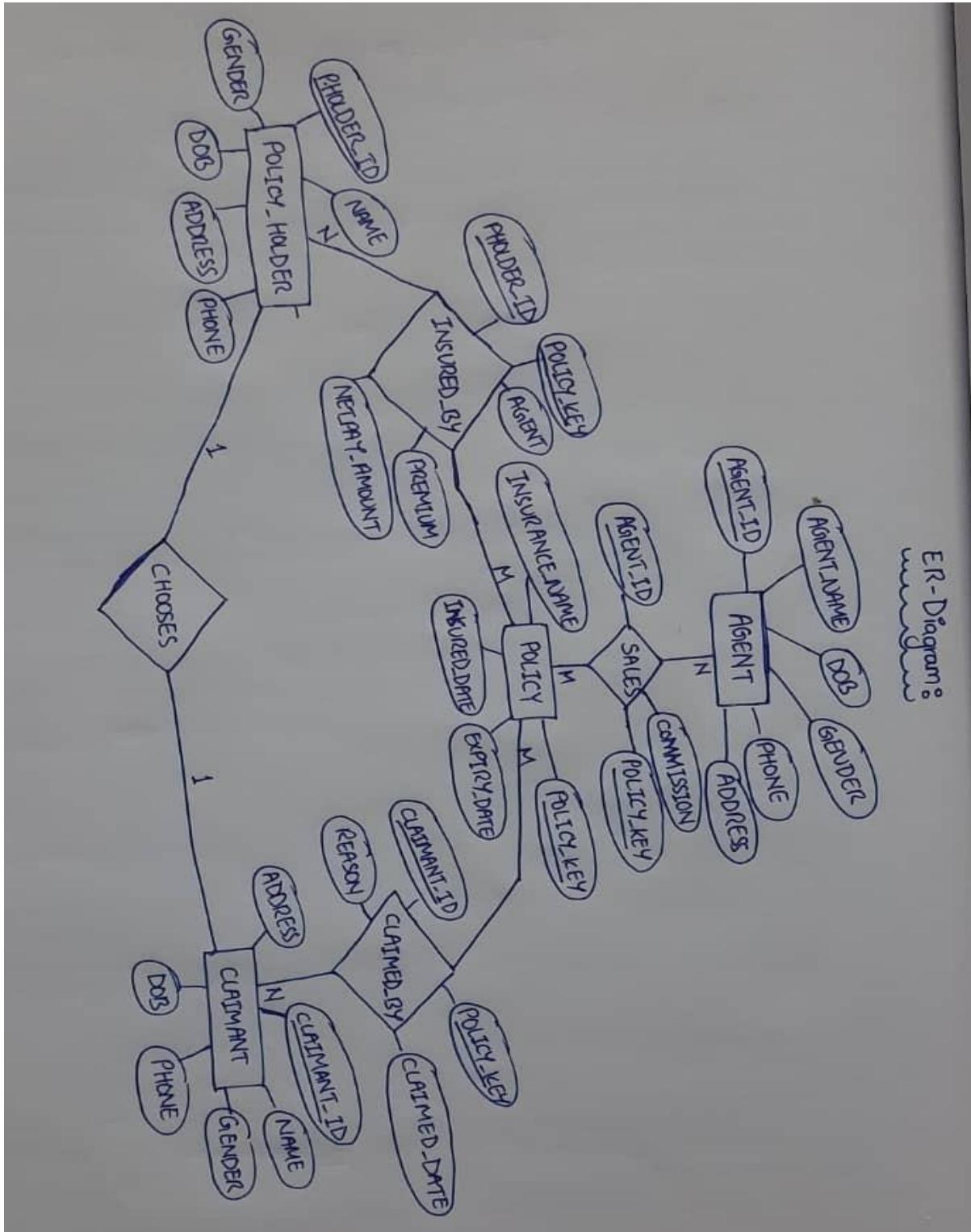
### **5.1.2.B HEALTH INSURANCE:-**

Customer Visits an Agent and asks for a Health Insurance. Agent Proposes the best suited health insurance to the customer as per his/her need. Customer decides whether he is interested in that policy if yes he asks for the quotation. Agent takes the customer details and gives it to the company. Company then according to the customer details specified by the agent depending on age of the customer gives different quotations on the basis of sum insured by the customer. Agent asks the customer the amount he/she wants to get insured for. Agent then gives the quotation to the customer. Customer then informs the agent whether he\she wants to get insured or not. If yes agent informs the company. If the customer is of age less than 45 he is issued the policy without any medical test. If the age of the customer is Greater than 45 then the company arranges the medical tests for the customer. Customer has to pay for the medical tests from his own pocket. If the customer passes the tests and the policy is issued then the amount of tests paid by the customer is then refunded. Once the customer is eligible for the policy after the test he/she has to give the premium amount to the agent. Agent gives the amount to the company. Company then issues a policy and the person is insured. Company sends the customer copy to the customer on the postal address specified by the customer.

## 5.2 ER Diagram:

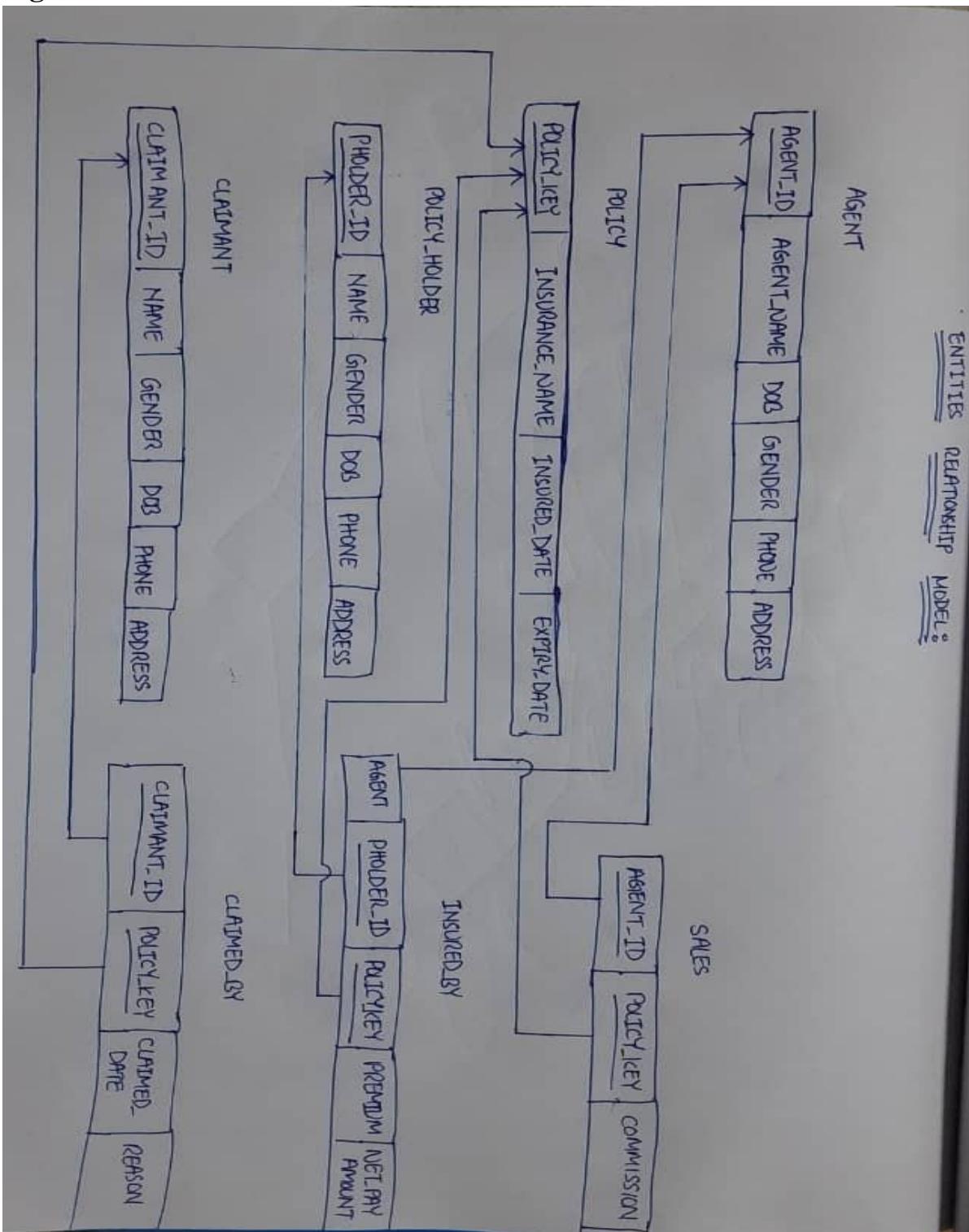
### 5.2.1 ER Diagram

Figure 5.1 : ER Diagram



## 5.2.2 Entities Relations Model:

Figure 5.2 : Entities Relations Model



## 5.3 Normalization:

### 5.3.1 NORMALIZATION OF ALL TABLES

Figure 5.3a : Normalization part 1

Normalization :

(1) Table AGENT

AGENT

AGENT-ID	AGENT-NAME	GENDER	DOB	PHONE	ADDRESS
----------	------------	--------	-----	-------	---------

(a) First Normal Form:

→ In a given table 'AGENT', 'PHONE' is a multivalued attribute as one agent can have two or more phone numbers, thus causing update anomaly. So, we normalize the table AGENT into:

AGENT 1

AGENT-ID	AGENT-NAME	GENDER	DOB	ADDRESS
----------	------------	--------	-----	---------

AGENT 2

AGENTID	PHONE
---------	-------

(b) Second Normal Form

AGENT 1

AGENT-ID	AGENT-NAME	GENDER	DOB	ADDRESS
FD1				

→ The table has only one candidate key 'AGENT-ID' so, there will be only one functional dependencies. Thus, conditions for 2NF is satisfied.

Figure 5.3b : Normalization part 2

(c) Third Normal Form

→ The table does not contain any transitive dependencies. So, the conditions for 3NF is also satisfied.

(d) Boyce-Codd Normal Form

→ As per the condition for BCNF, for each non-trivial functional dependency,  $X \rightarrow Y$  where 'X' is a superkey and ~~the~~ the table should be in third Normal Form.

In table 'AGENT,' there is only one functional dependency, i.e,

AGENT\_ID → AGENT\_NAME, GENDER, DOB, ADDRESS  
 ↑  
 Superkey

∴ Condition for BCNF is already satisfied, where BCNF only allows prime attributes → non-prime attributes, not other way round.

(2) Table Policy

POLICY

POLICY_KEY	INSURANCE_NAME	INSURED_DATE	EXPIRY_DATE
------------	----------------	--------------	-------------

(a) First Normal Form

→ In Table POLICY, there are no multivalued attributes nor nested relations. Therefore, condition for 1NF is satisfied.

(b) Second Normal Form

POLICY

POLICY_KEY	INSURANCE_NAME	INSURED_DATE	EXPIRY_DATE
FD1			

→ The table has only one candidate key 'POLICY\_KEY' with only one functional dependency 'FD1'. Therefore, conditions for 2NF is satisfied.

Figure 5.3c : Normalization part 3

(c) Third Normal Form

→ The table does not contain any transitive dependencies, satisfying the conditions for 3NF.

(d) Boyce - Codd Normal Form

→ In this table 'POLICY', there is only one functional dependency

$$\begin{array}{l} \text{POLICY\_KEY} \rightarrow \text{INSURANCE\_NAME}, \text{INSURED\_DATE}, \text{EXPIRY\_DATE} \\ \text{super key} \end{array}$$

The conditions for BCNF is already satisfied as  $X \rightarrow Y$  where 'X' is super key as explained earlier, and none of the non-prime attributes derives prime attributes.

(3) For Table POLICY\_HOLDER

POLICY\_HOLDER

PHOLDER_ID	NAME	GENDER	DOB	PHONE	ADDRESS

(a) First Normal Form

→ In this table 'POLICY\_HOLDER', there is one multi-valued attribute 'PHONE' as one person can have two or more phone numbers. Thus, it is normalized into 1NF as:

POLICY\_HOLDER 1

PHOLDER_ID	NAME	GENDER	DOB	ADDRESS

POLICY\_HOLDER 2

PHOLDER_ID	PHONE

↑  
Foreign Key

Figure 5.3d : Normalization part 4

b) Second Normal Form

POLICYHOLDER 1

PHOLDER_ID	NAME	GENDER	DOB	ADDRESS
FD1				

→ There is only one candidate key 'PHOLDER\_ID' with one functional dependency 'FD1'. So, the conditions for 2NF is already satisfied.

(c) Third Normal form

→ There are no functional or transitive dependencies among non-prime attributes. Thus, conditions for 3NF is satisfied.

(d) Boyce Codd Normal form

→ In this table, the functional dependency is

$$\begin{array}{c} \text{PHOLDER\_ID} \longrightarrow \text{NAME, GENDER, DOB, ADDRESS} \\ \uparrow \text{super key} \end{array}$$

Since, for functional dependency dependency  $X \rightarrow Y$  where ' $X$ ' is super key, the conditions for BCNF is already satisfied, where none of the non-prime attributes is deriving prime attributes.

Figure 5.3e : Normalization part 5

(a) Table CLAIMANT

CLAIMANT

CLAIMANT-ID	NAME	GENDER	DOB	PHONE	ADDRESS
-------------	------	--------	-----	-------	---------

(a) First Normal Form

→ The table contains multi-valued attribute 'PHONE' as one can have more phone numbers. So, we can normalize the table as :

CLAIMANT1

CLAIMANT-ID	NAME	GENDER	DOB	ADDRESS
-------------	------	--------	-----	---------

CLAIMANT2

CLAIMANT-ID	PHONE
-------------	-------

(b) Second Normal Form

CLAIMANT1

CLAIMANT-ID	NAME	GENDER	DOB	ADDRESS
FD1	↑	↑	↑	↑

→ The table has only one candidate key 'CLAIMANT-ID' with one functional dependency 'FD1'. So, the conditions for 2NF is satisfied.

(c) Third Normal Form

→ The table does not have any transitive dependency among the attributes. Hence, condition for 3NF is also satisfied.

Figure 5.3f : Normalization part 6

(d) Boyce Codd Normal Form

→ In this table, the functional dependency FD1 is

$$\begin{array}{c} \text{CLAIMANT-ID} \\ \uparrow \\ \text{superkey} \end{array} \rightarrow \text{NAME, GENDER, DOB, ADDRESS}$$

Since super key CLAIMANT ID determines all other attributes, it satisfies a condition  $X \rightarrow Y$  where 'X' is super key.

Hence, conditions for BCNF is also already satisfied, as no non-prime attributes derive prime attributes.

(5) Table Sales

SALES

<u>AGENT-ID</u>	<u>POLICY_KEY</u>	COMMISSION
-----------------	-------------------	------------

(a) First Normal Form

→ The table does not contain any multivalued attributes or relations. Hence, conditions for 1NF is satisfied.

(b) Second Normal Form

SALES

<u>AGENT-ID</u>	<u>POLICY_KEY</u>	COMMISSION
FD1		

→ There is only one functional dependency in this table as both AGENT-ID and POLICY-KEY derives COMMISSION together. Hence, conditions for 2NF is satisfied.

(c) Third Normal Form

→ The table does not contain any transitive dependencies, satisfying the conditions for 3NF.

Figure 5.3g : Normalization part 7

(d) Boyce Codd Normal Form

→ In the table, there is one functional dependency

$$\underbrace{\text{AGENT-ID}, \text{POLICY\_KEY}}_{\text{Prime attributes}} \rightarrow \underbrace{\text{COMMISSION}}_{\text{non-prime attributes}}$$

Since, only prime attributes are deriving non-prime attributes there is no update anomaly. Hence, conditions for BCNF is satisfied as BCNF does not allow non-prime  $\rightarrow$  prime attributes attributes

(e) Table INSURED\_BY

INSURED\_BY

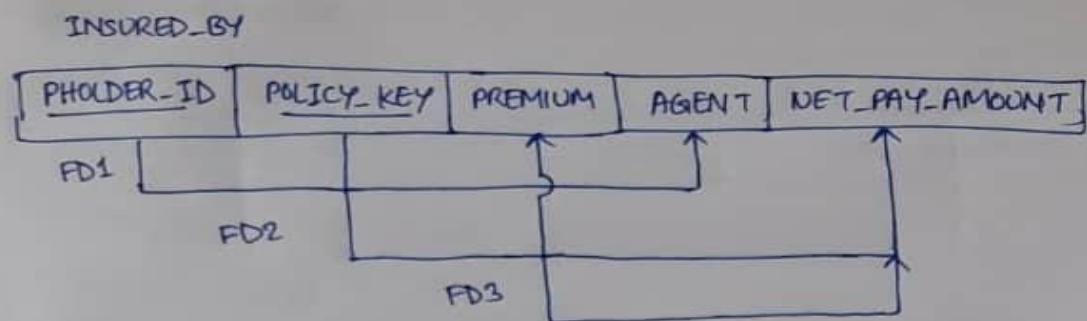
HOLDER_ID	POLICY_KEY	PREMIUM	AGENT	NET_PAY_AMOUNT
-----------	------------	---------	-------	----------------

(a) First Normal Form

→ In this table, there is no multi-valued attributes or nested relations, satisfying the conditions for 1NF.

Figure 5.3h : Normalization part 8

(b) Second Normal Form



Therefore, the table INSURED\_BY is normalized into 2NF as:

INSURED\_BY1

PHOLDER_ID	POLICY_KEY	AGENT
FD1		

INSURED\_BY2

POLICY_KEY	PREMIUM	NET_PAY_AMOUNT
FD2		
	FD3	

(c) Third Normal Form

Since, there is transitive dependencies:

$\text{POLICY\_KEY} \rightarrow \text{PREMIUM} \rightarrow \text{NET\_PAY\_AMOUNT}$

we further normalize the table as:

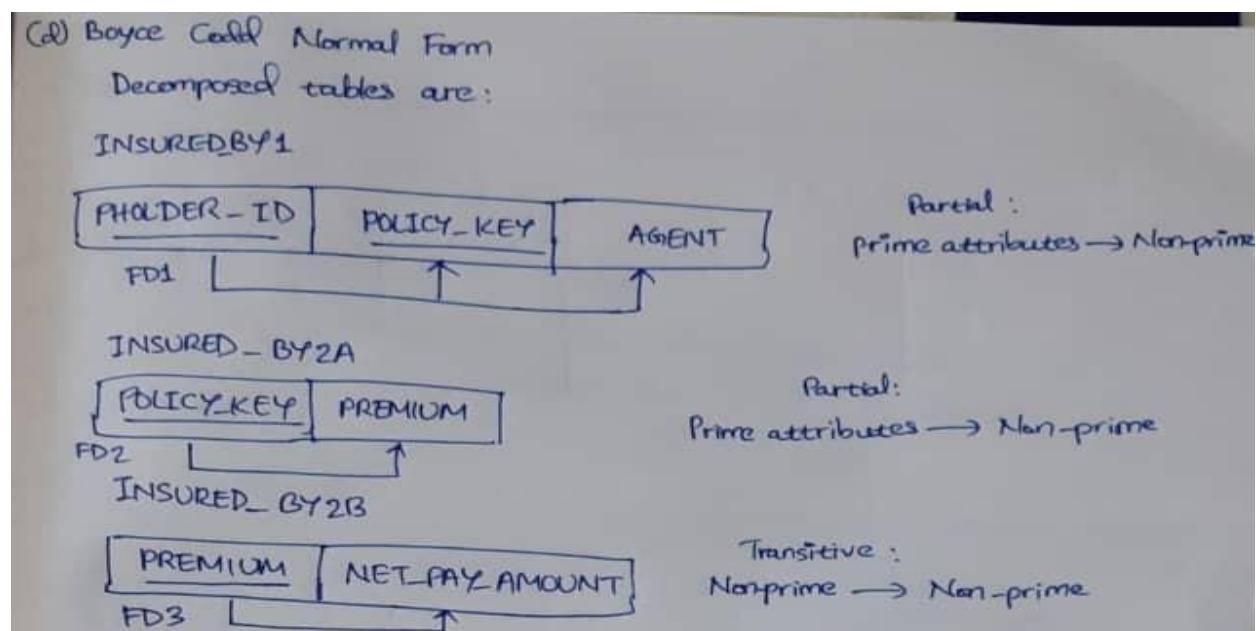
INSURED\_BY2A

POLICY_KEY	PREMIUM

INSURED\_BY2B

PREMIUM	NET_PAY_AMOUNT

Figure 5.3i : Normalization part 9



As you can see, there is no tables where non prime attributes derive prime attributes, therefore, conditions for BCNF is already satisfied.

(e) Table CLAIMED\_BY

<u>CLAIMANT_ID</u>	<u>POLICY_KEY</u>	CLAIMED_DATE	REASON
--------------------	-------------------	--------------	--------

(a) First Normal Form

→ There is no multivalued attributes or nested relations, satisfying conditions for 1NF.

(b) Second Normal Form

<u>CLAIMANT_ID</u>	<u>POLICY_KEY</u>	CLAIMED_DATE	REASON
FD1			

Here, there is only one partial dependency as both CLAIMANT\_ID and POLICY\_KEY Jointly determines CLAIMED\_DATE and REASON. Hence, the condition for 2NF is satisfied.

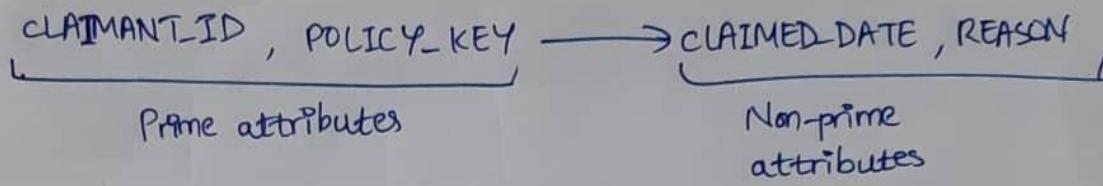
Figure 5.3j : Normalization part 10

(c) Third Normal Form

→ There is no transitive dependencies in a table, satisfying conditions for 3NF.

(d) Boyce Codd Normal Form

→ Since none of the non-prime attributes derive prime attributes, the conditions for BCNF is already satisfied where in FD1:



# Chapter 6

## RESULTS AND DISCUSSION

### 6.1 Creation of tables using DML commands:

#### 1) Table Agent:

After Normalizing, we get table Agent1 and Agent2. Creating table Agent1 and Agent2:

**Table 6.1: Agent1**

Attributes	Data types	Constraints
Agent_id	Varchar2(5)	Primary key
Agent_name	Varchar2(30)	
DOB	Date	
Gender	Char(1);	Check (Gender in ('M','F'))
Address	Varchar2(30)	

**Table 6.2 : Agent2**

Attributes	Data types	Constraints
Agent_id	Varchar2(5)	Foreign key
Phone	Number(10)	Unique

**Figure 6.1 : Creation of table Agent1 and Agent2**

The screenshot shows the SQL Plus interface with the following session history:

```
SQL> create table Agent1(
  2 AgentID varchar2(5),
  3 Agent_name varchar2(30),
  4 DOB date,
  5 Gender char(1) check(Gender in ('M','F')),
  6 Address varchar2(30),
  7 constraint agent_pri_key primary key(AgentID));
Table created.

SQL> create table Agent2(
  2 AgentID varchar2(5),
  3 Phone number(10),
  4 constraint agent_fk_key foreign key(AgentID) references Agent1,
  5 constraint unique_pno1 unique(Phone));
Table created.

SQL> desc Agent1;
Name          Null?    Type
-----        -----
AGENTID      NOT NULL VARCHAR2(5)
AGENT_NAME    VARCHAR2(30)
DOB           DATE
GENDER        CHAR(1)
ADDRESS       VARCHAR2(30)

SQL> desc Agent2;
Name          Null?    Type
-----        -----
AGENTID      NOT NULL VARCHAR2(5)
PHONE         NUMBER(10)

SQL>
```

## 2) Table policy

**Table 6.3 : Policy**

Attributes	Data types	Constraints
Policy_no	Varchar2(10)	Primary key
Insurance_Name	Number(10)	Check(in ('Health', 'Car', 'Life', 'Travel'))
Insured_date	Date	Check if it is before Expiry date
Expiry_date	Date	

**Figure 6.2 : Creation of table Policy**

The screenshot shows a Windows desktop environment with a SQL Plus window open. The window title is "SQL Plus". Inside, the following SQL code is executed:

```
SQL> create table Policy(
  2  policy_no varchar2(10) primary key,
  3  Insurance_name varchar2(10) constraint check_ins_name check(Insurance_name in ('Health','Life','Car','Travel')),
  4  Insured_date date,
  5  Expiry_date date,
  6  constraint checkInsdDate check(
  7  to_char(Insured_date,'YYYY-MM-DD')<to_char(Expiry_date,'YYYY-MM-DD'))
  8 );

Table created.

SQL> desc policy;
Name          Null?    Type
-----        -----   -----
POLICY_NO      NOT NULL VARCHAR2(10)
INSURANCE_NAME           VARCHAR2(10)
INSURED_DATE        DATE
EXPIRY_DATE         DATE
```

The command `desc policy;` is run to display the table structure. The table has four columns: `POLICY_NO` (primary key, not null, type `VARCHAR2(10)`), `INSURANCE_NAME` (type `VARCHAR2(10)`), `INSURED_DATE` (type `DATE`), and `EXPIRY_DATE` (type `DATE`). The window also shows the Windows taskbar at the bottom.

### 3) Table Policy\_Holder:

After Normalizing, we get table Policy\_Holder1 and Policy\_Holder2. Creating table Policy\_Holder1 and Policy\_Holder2:

**Table 6.4 : Policy\_Holder1**

Attributes	Data types	Constraints
PHolder_id	Varchar2(6)	Primary key
Name	Varchar2(30)	
DOB	Date	
Gender	Char(1);	Check (Gender in ('M','F'))
Address	Varchar2(30)	

**Table 6.5 : Policy\_Holder2**

Attributes	Data types	Constraints
PHolder_id	Varchar2(6)	Foreign key
Phone	Number(10)	Unique

**Figure 6.3 : Creation of table Policy\_Holder1 and Policy\_Holder2**

```
SQL> create table Policy_Holder1(
  2 PHolder_ID varchar2(6) primary key,
  3 name varchar2(30),
  4 Gender char(1) constraint checkgender2 check(Gender in ('M','F')),
  5 Address varchar2(30));
Table created.

SQL> create table Policy_Holder2(
  2 PHolder_ID varchar2(6),
  3 Phone number(10),
  4 constraint pholder_fk foreign key(PHolder_ID) references Policy_Holder1);
Table created.

SQL> desc policy_holder1;
Name          Null?    Type
-----        -----
PHOLDER_ID      NOT NULL VARCHAR2(6)
NAME           VARCHAR2(30)
GENDER          CHAR(1)
ADDRESS          VARCHAR2(30)

SQL> desc policy_holder2;
Name          Null?    Type
-----        -----
PHOLDER_ID      NOT NULL VARCHAR2(6)
PHONE           NUMBER(10)

SQL> -
SQL> alter table Policy_holder1 add DOB date;
Table altered.

SQL> desc policy_holder1;
Name          Null?    Type
-----        -----
PHOLDER_ID      NOT NULL VARCHAR2(6)
NAME           VARCHAR2(30)
GENDER          CHAR(1)
ADDRESS          VARCHAR2(30)
DOB             DATE
```

## 4) Claimant

After Normalizing, we get table Claimant1 and Claimant2. Creating table Claimant1 and Claimant2:

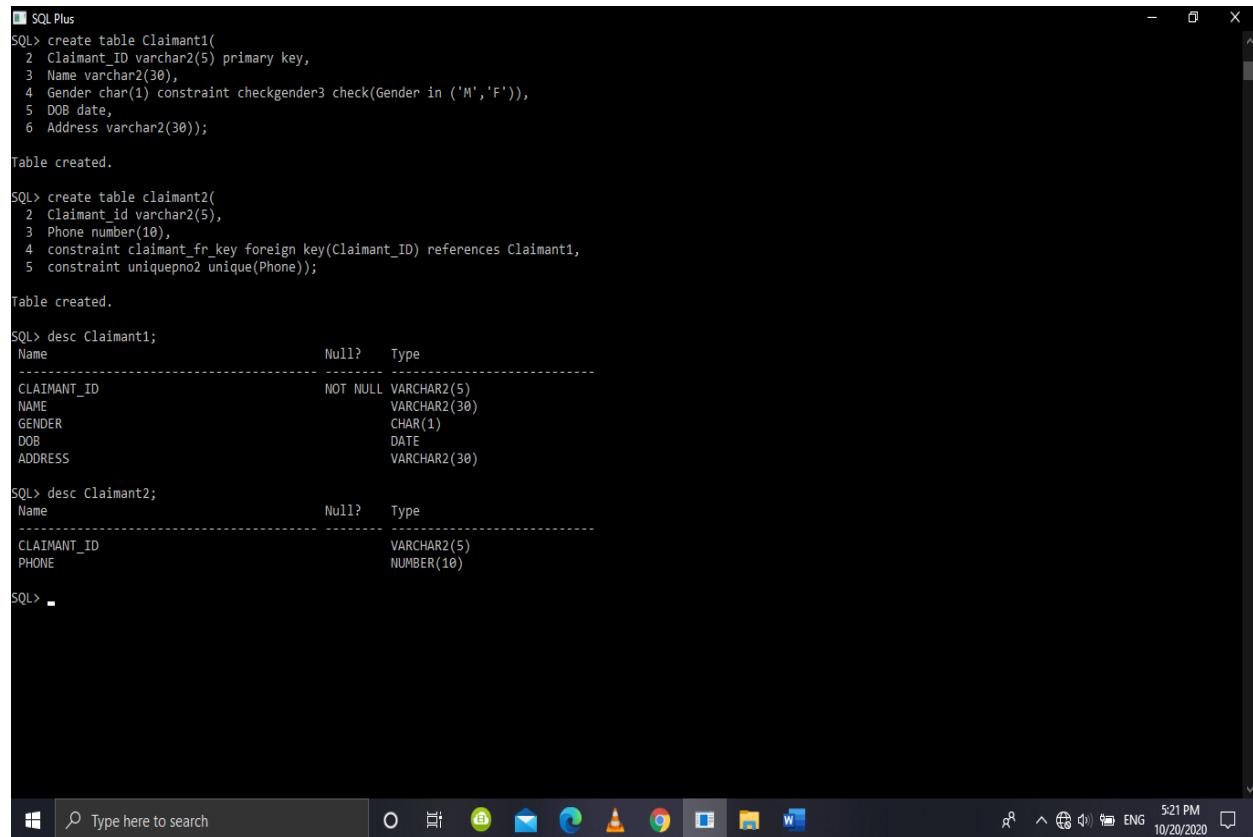
**Table 6.6 : Claimant1**

Attributes	Data types	Constraints
Claimant_id	Varchar2(5)	Primary key
Name	Varchar2(30)	
DOB	Date	
Gender	Char(1);	Check (Gender in ('M','F'))
Address	Varchar2(30)	

**Table 6.7 : Claimant2**

Attributes	Data types	Constraints
Claimant_id	Varchar2(5)	Foreign key
Phone	Number(10)	Unique

**Figure 6.4 : Creation of table Claimant1 and Claimant2**



```
SQL> create table Claimant1(
  2 Claimant_ID varchar2(5) primary key,
  3 Name varchar2(30),
  4 Gender char(1) constraint checkgender3 check(Gender in ('M','F')),
  5 DOB date,
  6 Address varchar2(30));

Table created.

SQL> create table claimant2(
  2 Claimant_id varchar2(5),
  3 Phone number(10),
  4 constraint claimant_fk foreign key(Claimant_ID) references Claimant1,
  5 constraint uniquepno2 unique(Phone));

Table created.

SQL> desc Claimant1;
Name          Null?    Type
-----        -----
CLAIMANT_ID      NOT NULL VARCHAR2(5)
NAME            VARCHAR2(30)
GENDER           CHAR(1)
DOB              DATE
ADDRESS          VARCHAR2(30)

SQL> desc Claimant2;
Name          Null?    Type
-----        -----
CLAIMANT_ID      VARCHAR2(5)
PHONE           NUMBER(10)

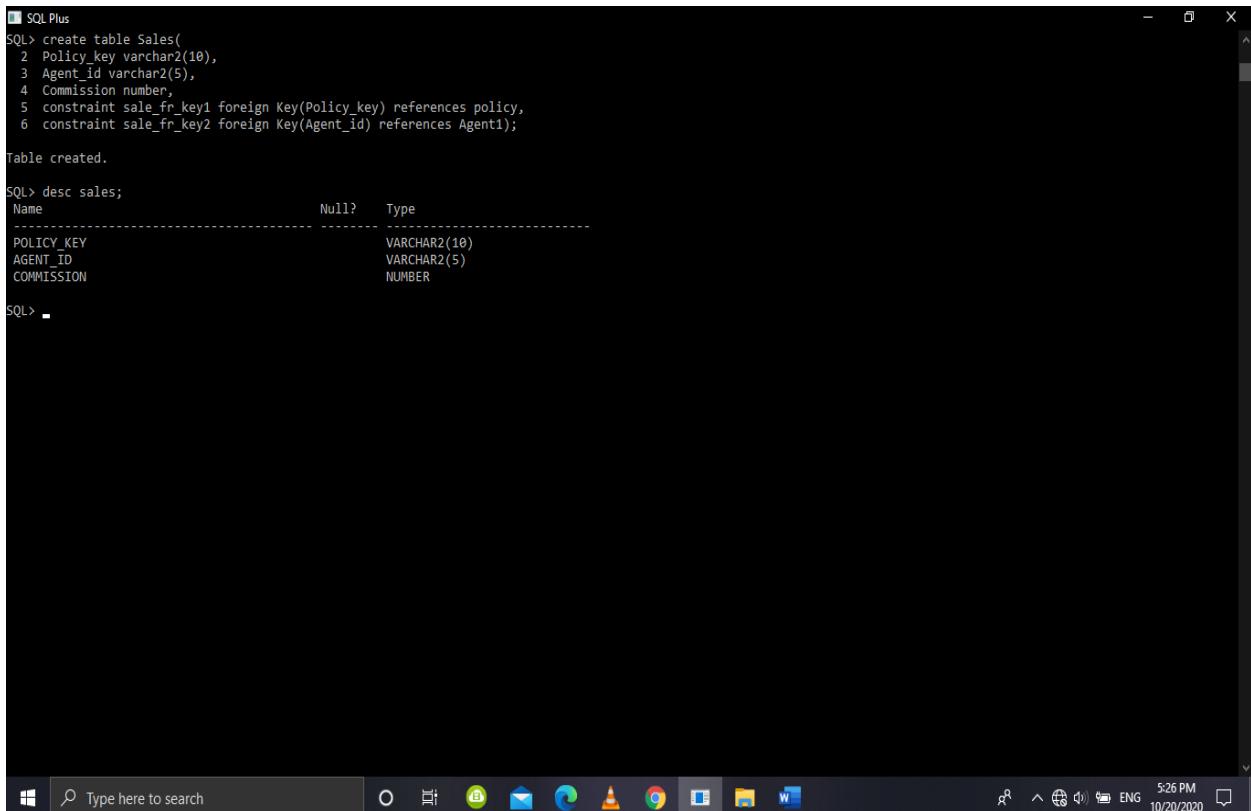
SQL>
```

## 5) Table Sales

**Table 6.8 : Sales**

Attributes	Data types	Constraints
Policy_key	Varchar2(10)	Primary key, Foreign key
Agent_id	Varchar2(5)	Foreign key
Commission	number	
Claimant_id	Varchar2(5)	Foreign key

**Figure 6.5 : Creation of table Sales**



The screenshot shows a Windows desktop with a dark theme. A SQL Plus window is open, displaying the creation of the Sales table and its description. The table has four columns: POLICY\_KEY (VARCHAR2(10)), AGENT\_ID (VARCHAR2(5)), COMMISSION (NUMBER), and CLAIMANT\_ID (VARCHAR2(5)). The CLAIMANT\_ID column is defined as a foreign key referencing the CLAIMANT\_ID column in the CLAIMANT table.

```
SQL> create table Sales(
  2 Policy_key varchar2(10),
  3 Agent_id varchar2(5),
  4 Commission number,
  5 constraint sale_fk1 foreign Key(Policy_key) references policy,
  6 constraint sale_fk2 foreign Key(Agent_id) references Agent1;

Table created.

SQL> desc sales;
   Name          Null?    Type
-----  -----
POLICY_KEY           VARCHAR2(10)
AGENT_ID            VARCHAR2(5)
COMMISSION          NUMBER

SQL>
```

SQL> alter table sales add Claimant\_id varchar2(5) add constraint claimfrkey foreign key(Claimant\_id) references claimant1;

Table altered.

## 6) Table Insured\_by

After Normalizing, we get table Insured\_by1A and Insured\_by1B and Insured\_by2. Creating table Insured\_by1A and Insured\_by1B and Insured\_by2:

**Table 6.9 : Insured\_by1**

Attributes	Data types	Constraints
Policy_key	Varchar2(10)	Foreign key, Primary key
Pholder_id	Varchar2(6)	Foreign key
Agent	Varchar2(5)	Foreign key

**Table 6.10 : Insured\_by2A**

Attributes	Data types	Constraints
Policy_key	Varchar2(10)	Primary key
Premium	Number	Foreign key

**Table 6.11 : Insured\_by2B**

Attributes	Data types	Constraints
Premium	Number	Primary key
Net_pay_Amount	Number	

**Figure 6.6 : Creation of table Insured\_by1, Insured\_by2A and Insured\_by2B**

```
SQL> create table Insured_by1(
  2 policy_key varchar2(10) primary key,
  3 Pholder_id varchar2(6),
  4 agent_id varchar2(5),
  5 constraint ins_by_fr_key1 foreign key(policy_key) references policy,
  6 constraint ins_by_fr_key2 foreign key(pholder_id) references Policy_holder,
  7 constraint ins_by_fr_key3 foreign key(agent_id) references Agent1);
Table created.

SQL> create table Insured_by2A(
  2 Policy_key varchar2(10) primary key,
  3 Premium number,
  4 constraint ins_by_fr_key4 foreign key(policy_key) references policy);
Table created.

SQL> create table Insured_by2B(
  2 Premium number primary key,
  3 Net_pay_amount number);
Table created.

SQL> alter table insured_by2A add constraint ins_by_fr_key5 foreign key(Premium) references Insured_by_2B;
alter table insured_by2A add constraint ins_by_fr_key5 foreign key(Premium) references Insured_by_2B
*
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> alter table insured_by2A add constraint ins_by_fr_key5 foreign key(Premium) references Insured_by2B;
Table altered.

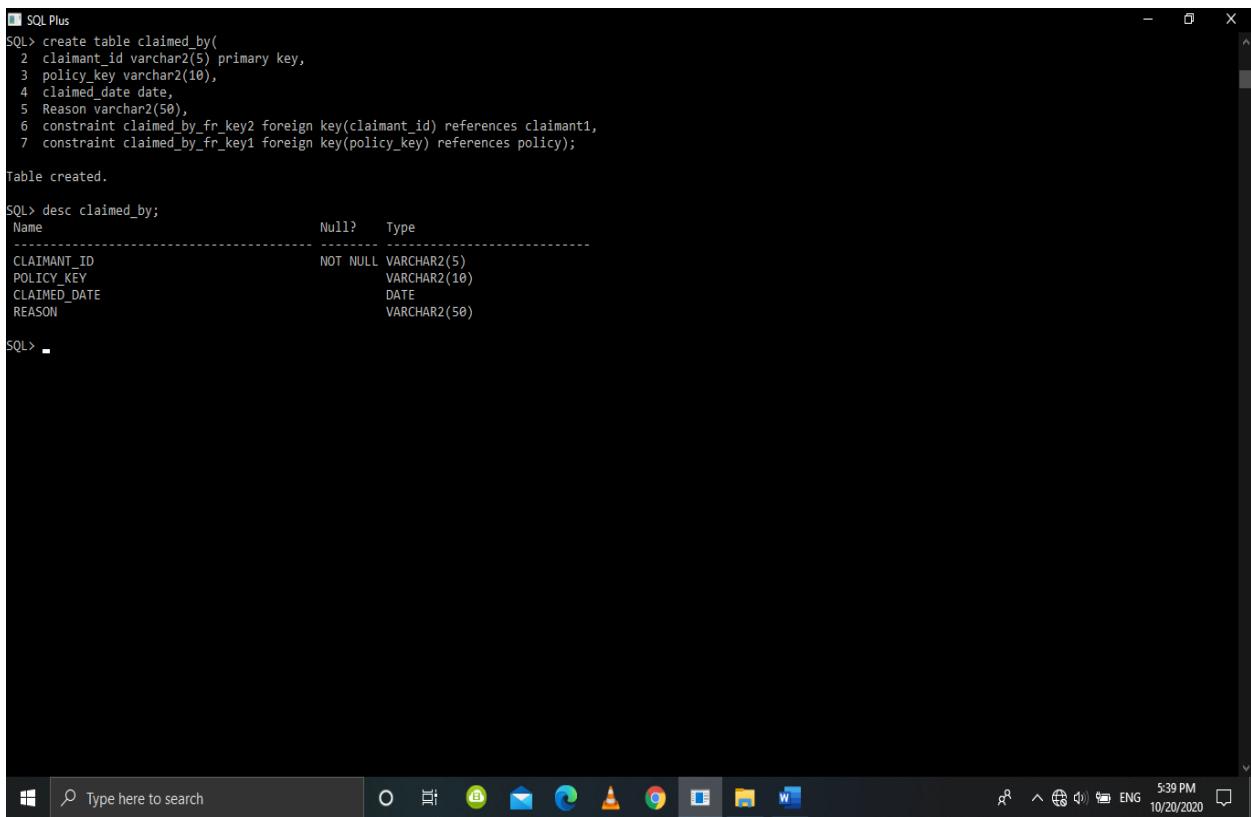
SQL>
```

## 7) Table Claimed\_by

**Table 6.12 : Claimed\_by**

Attributes	Data types	Constraints
Claimant_id	Varchar2(5)	Foreign key
Policy_key	Number(11)	Foreign Key
Claimed_Date	Date	
Reason	Varchar2(50)	

**Figure 6.7 : Creation of table Claimed\_by**



The screenshot shows a Windows desktop environment with a SQL Plus window open. The window title is "SQL Plus". Inside, the following SQL commands are run:

```
SQL> create table claimed_by(
  2 claimant_id varchar2(5) primary key,
  3 policy_key varchar2(10),
  4 claimed_date date,
  5 Reason varchar2(50),
  6 constraint claimed_by_fk2 foreign key(claimant_id) references claimant1,
  7 constraint claimed_by_fk1 foreign key(policy_key) references policy);

Table created.

SQL> desc claimed_by;
Name          Null?    Type
-----        -----
CLAIMANT_ID      NOT NULL  VARCHAR2(5)
POLICY_KEY           VARCHAR2(10)
CLAIMED_DATE        DATE
REASON             VARCHAR2(50)

SQL>
```

The system tray at the bottom shows icons for File Explorer, Mail, and other applications, along with the date and time (10/20/2020, 5:39 PM).

**8) Log in table:** These extra tables are created to store password of User and admin for login purpose:

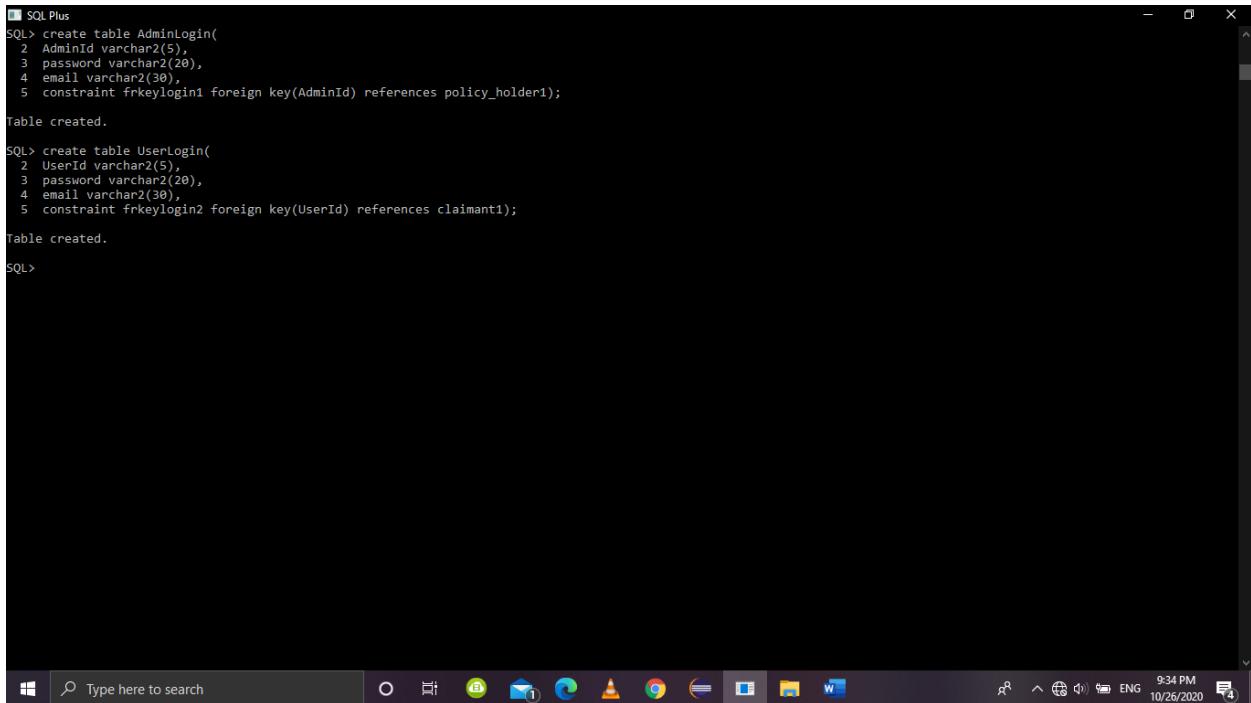
**a) Table 6.13 : UserLogin:**

Attributes	Data types	Constraints
User_id	Varchar2(5)	Primary, Foreign key
Password	Varchar2(20)	
Email	Varchar2(30)	

**b) Table 6.14 : AdminLogin:**

Attributes	Data types	Constraints
Admin_id	Varchar2(5)	Primary, Foreign key
Password	Varchar2(20)	
Email	Varchar2(30)	

**Figure 6.8 : Creation of table UserLogin and AdminLogin**



```
SQL> create table AdminLogin(
  2 AdminId varchar2(5),
  3 password varchar2(20),
  4 email varchar2(30),
  5 constraint frkeylogin1 foreign key(AdminId) references policy_holder1);
Table created.

SQL> create table UserLogin(
  2 UserId varchar2(5),
  3 password varchar2(20),
  4 email varchar2(30),
  5 constraint frkeylogin2 foreign key(UserId) references claimant1);
Table created.

SQL>
```

## 6.2 Updated values in database:

Figure 6.9 : Updated values in table Claimant1 and Claimant2

SQL> select \* from Claimant1;

CLAIM NAME	G DOB	ADDRESS
C0001 Lhendup Dorji	M 16-DEC-96	West Gate Park, Ikebukuro
C0002 Tenzin Khorlo	M 18-MAR-87	North Valley 36, Hyderabad
C0003 Preetam Singh	M 10-JUN-88	Upper Street 31, Kolkata
C0004 Rishi Gaughal	M 17-AUG-92	Down Street 31, Hyderabad
C0005 Neha Maya Rai	F 31-OCT-84	Flower Street 7, New Delhi
C0006 Ohani Bahadur	M 19-SEP-88	New Upper Street 34, Punjab
C0007 Priti Shekhar	F 20-JUL-89	Old Lower Market 1, Punjab
C0008 Dorji Phutsho	M 08-JAN-79	Eastern Road City, Mumbai
C0009 Tenzin Namgay	M 28-FEB-83	Western Road City, Mumbai
C0010 Sangay Mayuri	F 27-MAY-74	Down Street 34, New Delhi

10 rows selected.

SQL> select \* from Claimant2;

CLAIM	PHONE
C0001	9718261021
C0002	9727361720
C0003	8192121720
C0004	8177711720
C0005	7889211720
C0006	9082121720
C0007	7833521720
C0008	8835213492
C0009	9835218714
C0010	5685178714

10 rows selected.

SQL> select \* from Agent1;

AGENT	AGENT_NAME	DOB	G ADDRESS
A0001	Tenzin Lhendup	16-MAR-79	M East Gate Park, Ikebukuro
A0002	Lhendup Khorlo	19-APR-83	F Upper Street 31, Hyderabad
A0003	Rishi Chaurasia	28-JUN-76	M NRDwn Street 54, Kolkata
A0004	Pritam Singhal	13-OCT-81	M New South City, New Delhi
A0005	Seneha Kapoori	13-OCT-81	F Flower city 13, Mumbai

6 rows selected.

SQL> select \* from Agent2;

AGENT	PHONE
A0001	9789102827
A0002	9782803452
A0003	9728913487
A0004	9782618721
A0005	9786241502
A0006	9378171298

6 rows selected.

SQL> select \* from Policy;

POLICY_NO	INSURANCE_	INSURED_D	EXPIRY_DA
100A673312	Health	18-JAN-16	18-JAN-21
120A189312	Travel	16-FEB-15	16-FEB-18
131A189001	Life	18-FEB-15	18-FEB-21
151A189431	Car	21-FEB-17	21-FEB-22
151A134461	Car	21-MAR-18	21-FEB-24
171A134461	Health	16-APR-17	16-APR-23
171A001481	Car	18-JUN-15	18-JUN-20
171A0064534	Travel	24-JUL-15	24-JUL-20
171A192812	Life	16-JUL-15	16-JUL-21
191A001722	Health	18-AUG-14	16-JUL-22

10 rows selected.

SQL>

Figure 6.10 : Updated values in table Agent1, Agent2 and Policy

SQL> select \* from Agent1;

AGENT	AGENT_NAME	DOB	G ADDRESS
A0001	Tenzin Lhendup	16-MAR-79	M East Gate Park, Ikebukuro
A0002	Lhendup Khorlo	19-APR-83	F Upper Street 31, Hyderabad
A0003	Rishi Chaurasia	28-JUN-76	M NRDwn Street 54, Kolkata
A0004	Pritam Singhal	13-OCT-81	M New South City, New Delhi
A0005	Seneha Kapoori	13-OCT-81	F Flower city 13, Mumbai
A0006	Liwangi Lepcha	27-DEC-70	M Flower FC City 78, Mumbai

6 rows selected.

SQL> select \* from Agent2;

AGENT	PHONE
A0001	9789102827
A0002	9782803452
A0003	9728913487
A0004	9782618721
A0005	9786241502
A0006	9378171298

6 rows selected.

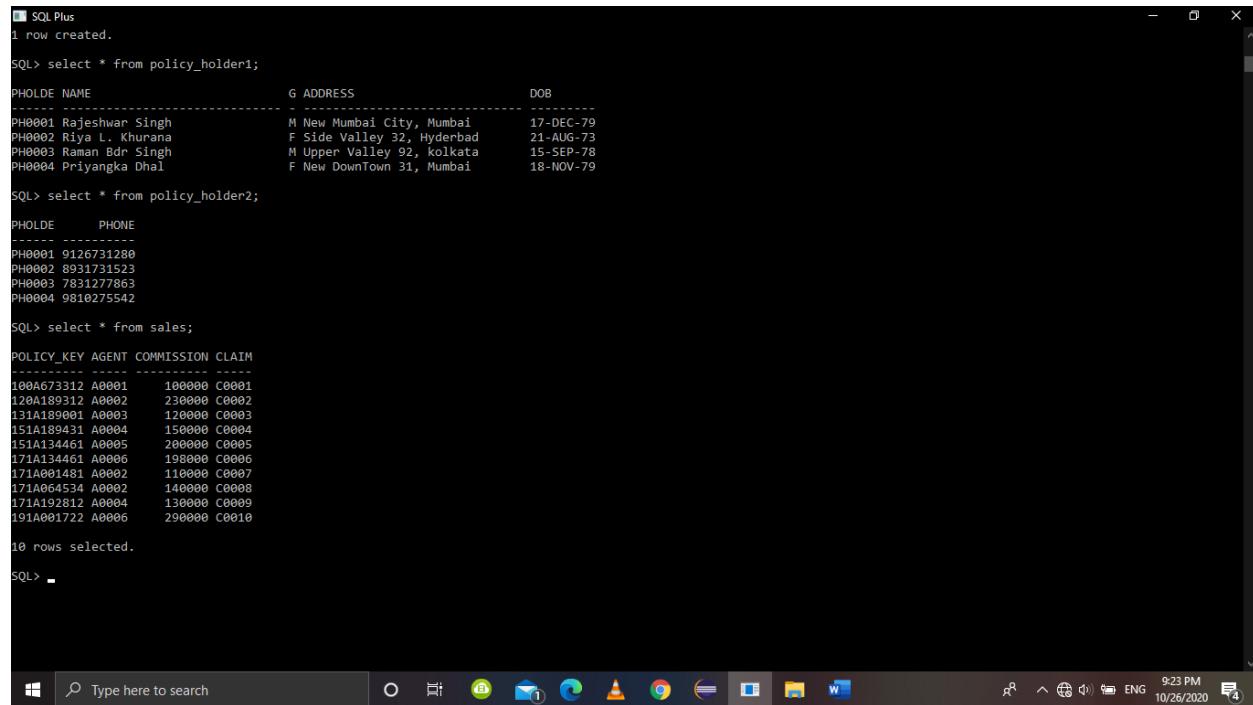
SQL> select \* from Policy;

POLICY_NO	INSURANCE_	INSURED_D	EXPIRY_DA
100A673312	Health	18-JAN-16	18-JAN-21
120A189312	Travel	16-FEB-15	16-FEB-18
131A189001	Life	18-FEB-15	18-FEB-21
151A189431	Car	21-FEB-17	21-FEB-22
151A134461	Car	21-MAR-18	21-FEB-24
171A134461	Health	16-APR-17	16-APR-23
171A001481	Car	18-JUN-15	18-JUN-20
171A0064534	Travel	24-JUL-15	24-JUL-20
171A192812	Life	16-JUL-15	16-JUL-21
191A001722	Health	18-AUG-14	16-JUL-22

10 rows selected.

SQL>

**Figure 6.11 : Updated values in tables Policy\_holder1, Policy\_Holder2 and Sales**



```

SQL> select * from policy_holder1;
PHOLDE NAME          G ADDRESS           DOB
----- -----
PH0001 Rajeshwar Singh   M New Mumbai City, Mumbai    17-DEC-79
PH0002 Riya L. Khurana   F Side Valley 32, Hyderabad 21-AUG-73
PH0003 Raman Bdr Singh   M Upper Valley 92, kolkata   15-SEP-78
PH0004 Priyangka Dhal    F New DownTown 31, Mumbai    18-NOV-79

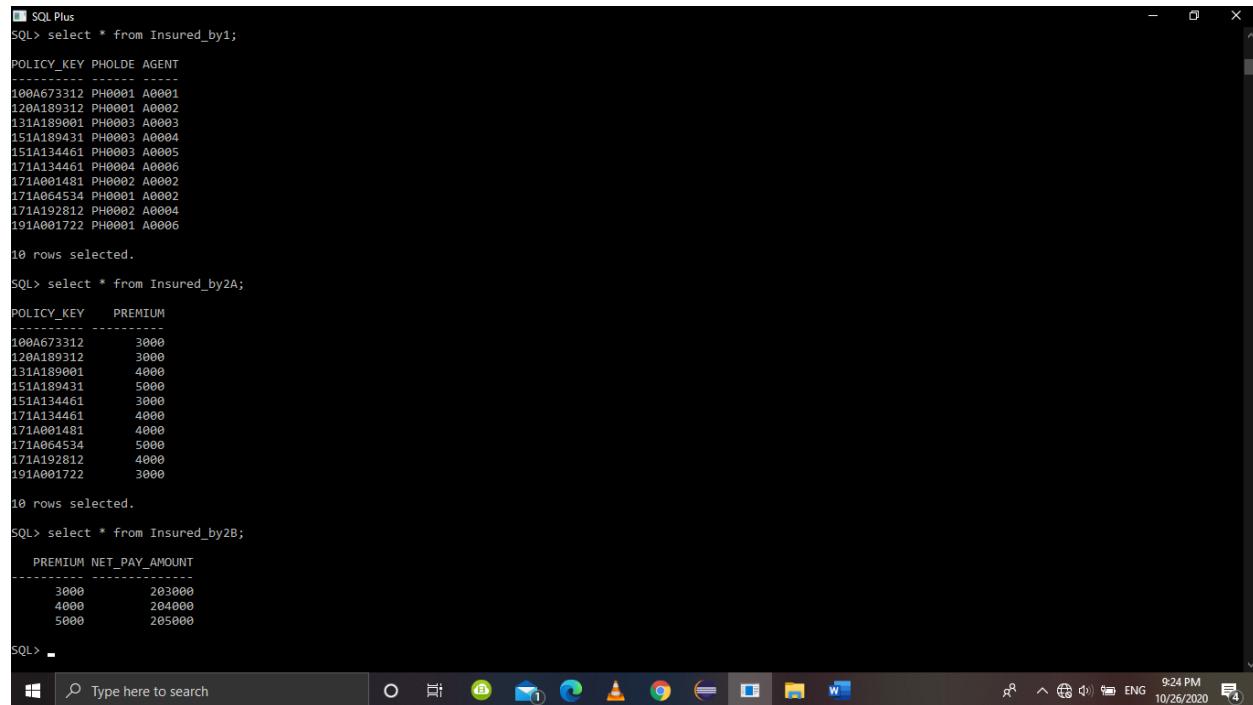
SQL> select * from policy_holder2;
PHOLDE PHONE
-----
PH0001 9126731280
PH0002 8931731523
PH0003 7831277863
PH0004 9810275542

SQL> select * from sales;
POLICY_KEY AGENT COMMISSION CLAIM
-----
100A673312 A0001     100000 C0001
120A189312 PH0001 A0002
131A189001 PH0003 A0003
151A189431 PH0003 A0004
151A134461 PH0003 A0005
171A134461 PH0004 A0006
171A001481 PH0002 A0002
171A064534 PH0001 A0002
171A192812 PH0002 A0004
191A001722 PH0001 A0006
10 rows selected.

SQL> -

```

**Figure 6.12 : Updated values in table Insured\_by1, Insured\_by2A and Insured\_by2B**



```

SQL> select * from Insured_by1;
POLICY_KEY PHOLDE AGENT
-----
100A673312 PH0001 A0001
120A189312 PH0001 A0002
131A189001 PH0003 A0003
151A189431 PH0003 A0004
151A134461 PH0003 A0005
171A134461 PH0004 A0006
171A001481 PH0002 A0002
171A064534 PH0001 A0002
171A192812 PH0002 A0004
191A001722 PH0001 A0006
10 rows selected.

SQL> select * from Insured_by2A;
POLICY_KEY PREMIUM
-----
100A673312      3000
120A189312      3000
131A189001      4000
151A189431      5000
151A134461      3000
171A134461      4000
171A001481      4000
171A064534      5000
171A192812      4000
191A001722      3000
10 rows selected.

SQL> select * from Insured_by2B;
PREMIUM NET_PAY_AMOUNT
-----
3000      203000
4000      204000
5000      205000
SQL> -

```

**Figure 6.13 : Updated values in tables insured\_by2B and Claimed\_by**

SQL> select \* from Insured\_by2B;

PREMIUM	NET_PAY_AMOUNT
3000	203000
4000	204000
5000	205000

SQL> select \* from claimed\_by;

CLAIM	POLICY_KEY	CLAIMED_D	REASON
C0003	131A189001	04-MAY-20	Death of his parents due to COVID 19
C0007	171A001481	21-JUN-18	Car Washed away by Flood
C0008	171A064534	09-AUG-16	Travelling to USA for Work

SQL>

**Figure 6.14 : Updated values in tables UserLogin and AdminLogin**

SQL> select \* from adminLogin;

ADMINI	PASSWORD	EMAIL
PH0001	admin1	rajeshwari@gmail.com
PH0002	admin2	riya@gmail.com
PH0003	admin3	raman@gmail.com
PH0004	admin4	priyangka@gmail.com

SQL> select \* from UserLogin;

USERI	PASSWORD	EMAIL
C0001	user1	lhendup@gmail.com
C0002	user2	tenzing@gmail.com
C0003	user3	preetam@gmail.com
C0004	user4	rishi@gmail.com
C0005	user5	neha@gmail.com
C0006	user6	Dani@gmail.com
C0007	user7	Priti@gmail.com
C0008	user8	dorji@gmail.com
C0009	user9	maya@gmail.com
C0010	user10	sangay@gmail.com

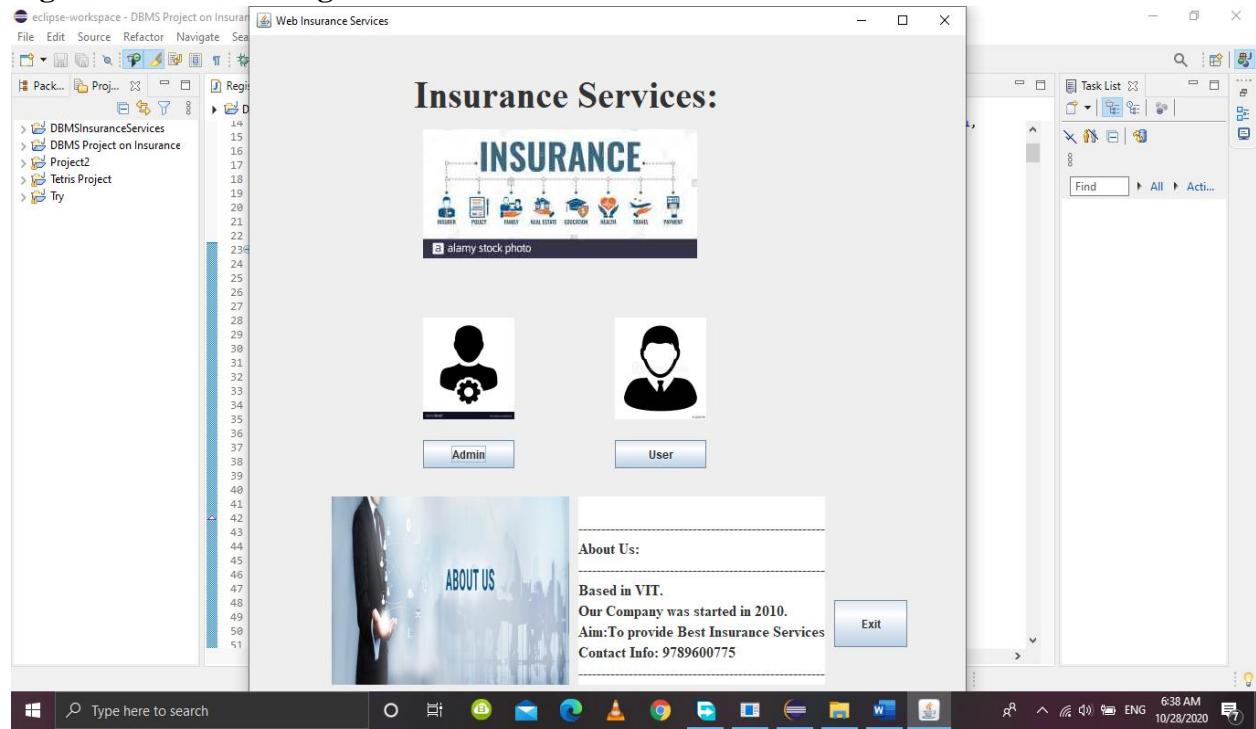
10 rows selected.

SQL>

### 6.3 Implementation in Eclipse(App Creation):

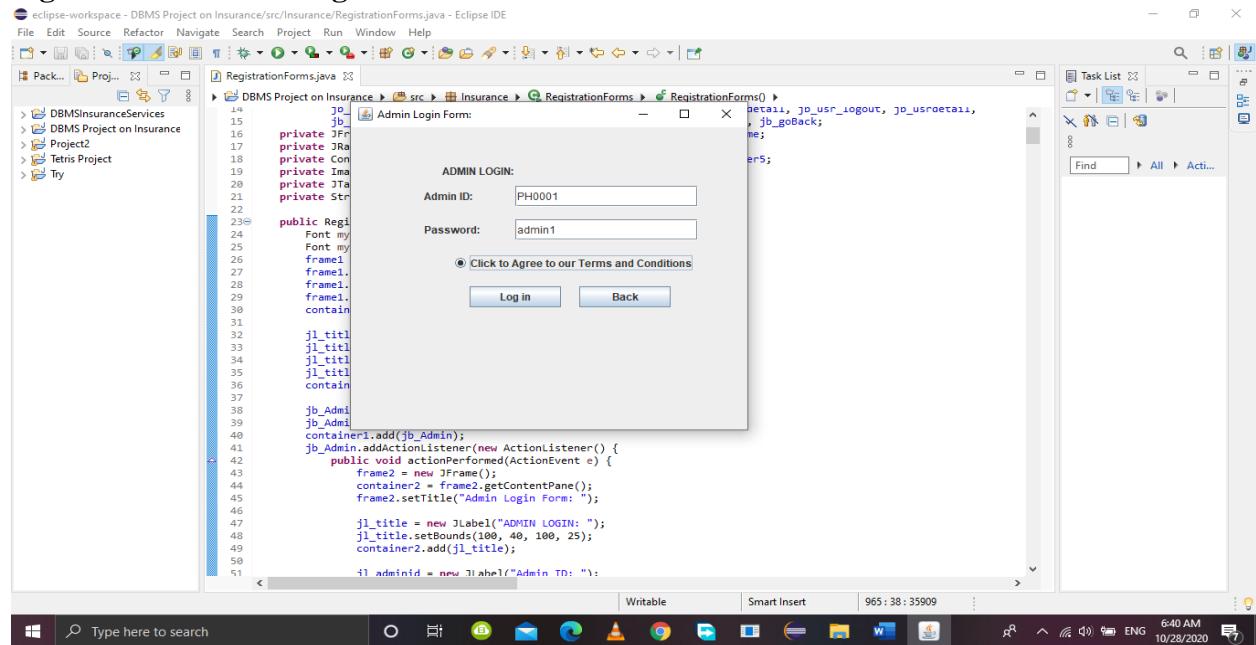
1) Home Page:

Figure 6.15 : Home Page



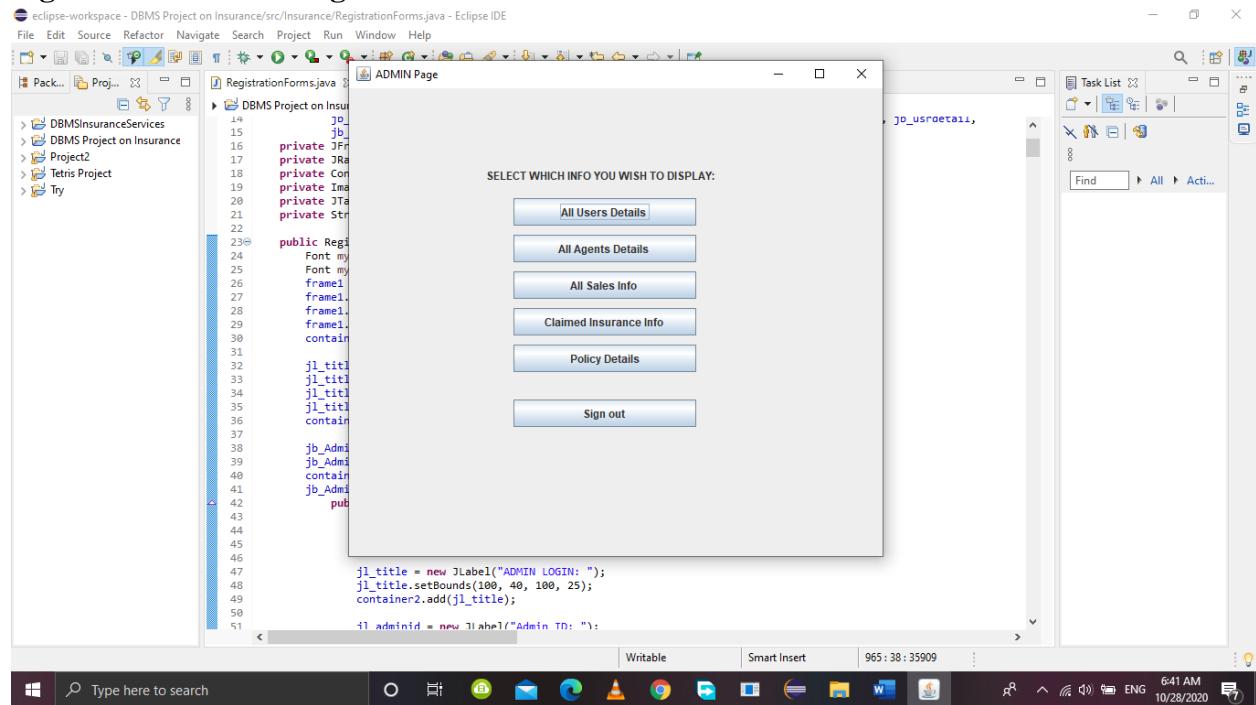
2) After Clicking on Admin:

Figure 6.16 : Admin Login Form



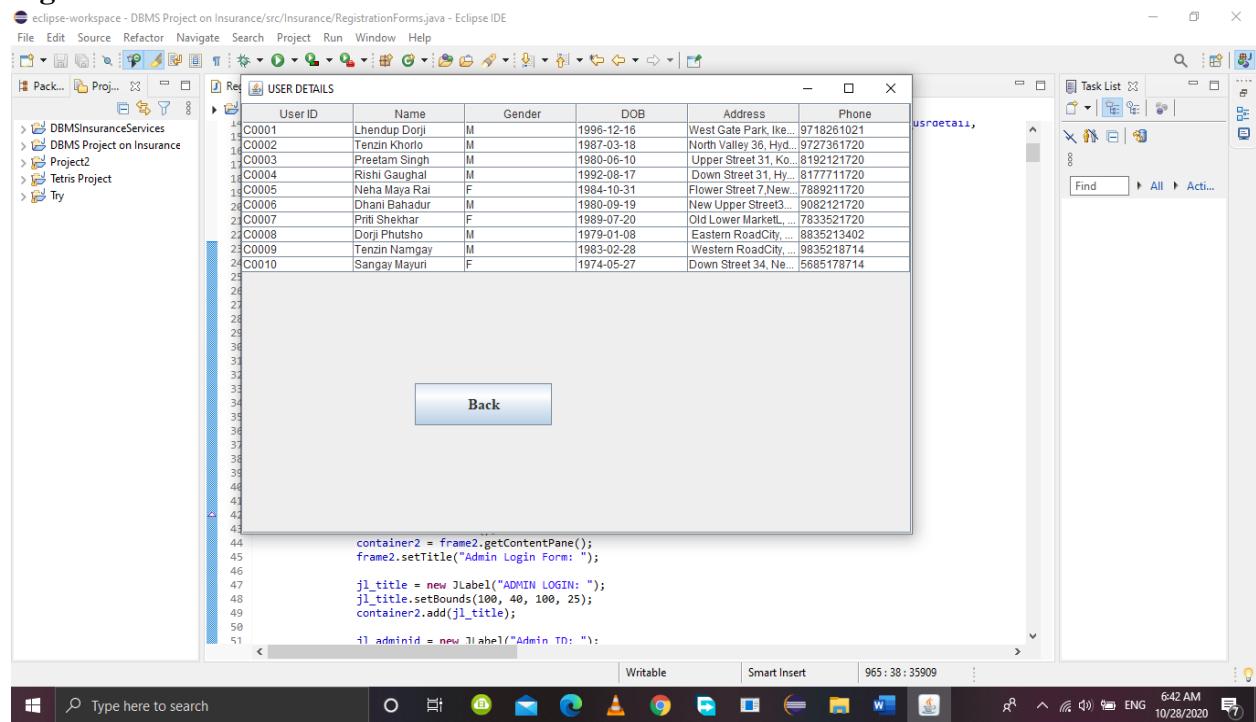
3) We get admin page

**Figure 6.17 : Admin Page**



4) If we select All User Details and Back button can be used to go back to previous page:

**Figure 6.18 : User Details**



5) If we select All Agents Details:

**Figure 6.19 : All AGENTS DETAILS**

The screenshot shows a Microsoft Word document window titled "DBMS project - Word". The ribbon menu includes File, Home, Insert, Design, Layout, References, Mailings, Review, View, Help, and a search bar. A toolbar with various icons is visible above the ribbon. The main content area displays a table titled "ALL AGENTS DETAILS" with the following data:

Agent ID	Name	Gender	DOB	Address	Phone
A0001	Tenzin Lhendup	M	1979-03-16	East Gate Park, Ikebukuro	9789102827
A0002	Lhendup Khorlo	F	1983-04-19	Upper Street 31, Hyderabad	9782803452
A0003	Rishi Chaurasia	M	1976-06-28	NRDown Street 54, Kolkata	9728913487
A0004	Pritam Singh	M	1981-10-13	New South City, New Delhi	9782618721
A0005	Seneha Kapoori	F	1981-10-13	Flower city 13, Mumbai	9789241502
A0006	Liwangi Lepcha	M	1970-12-27	Flower FC City 78, Mumbai	9378171298

Below the table is a "Back" button.

8) If we select Policy Details:

6) If we select All Sales Info:

**Figure 6.20 : All Sales Info**

The screenshot shows the Eclipse IDE interface with a Java code editor at the bottom. The code editor contains the following Java code:

```

43     frame2 = new JFrame();
44     container2 = frame2.getContentPane();
45     frame2.setTitle("Admin Login Form:");
46
47     jl_title = new JLabel("ADMIN LOGIN: ");
48     jl_title.setBounds(100, 40, 100, 25);
49     container2.add(jl_title);
50
51     jl_adminid = new JLabel("Admin ID: ");

```

On top of the code editor is a table titled "ALL SALES INFO" with the following data:

Policy Key	PHolder ID	Pholder Name	Agent ID	Agent Name	Claimant ID	Claimant Name	Commission
100A673312	PH0001	Rajeshwar Singh	A0001	Tenzin Lhendup	C0001	Lhendup Dorji	100000
120A189312	PH0001	Rajeshwar Singh	A0002	Lhendup Khorlo	C0002	Tenzin Khorlo	230000
131A189001	PH0003	Raman Bdr Singh	A0003	Rishi Chaurasia	C0003	Preetam Singh	120000
151A189431	PH0003	Raman Bdr Singh	A0004	Pritam Singh	C0004	Rishi Gaughal	150000
151A134461	PH0003	Raman Bdr Singh	A0005	Seneha Kapoori	C0005	Neha Maya Rai	200000
171A134461	PH0004	Priyanka Dhal	A0006	Liwangi Lepcha	C0006	Dhani Bahadur	198000
171A001481	PH0002	Riya L. Khurana	A0002	Lhendup Khorlo	C0007	Priti Shekhar	110000
171A064534	PH0001	Rajeshwar Singh	A0002	Lhendup Khorlo	C0008	Dorji Phutsho	140000
171A192812	PH0002	Riya L. Khurana	A0004	Pritam Singh	C0009	Tenzin Namgay	130000
191A001722	PH0001	Rajeshwar Singh	A0006	Liwangi Lepcha	C0010	Sangay Mayuri	290000

Below the table is a "Back" button.

7) If we select All Claimed Insurance Info:

**Figure 6.21 : All Claimed Insurance Details**

A screenshot of a Microsoft Word document titled "DBMS project - Word". The document contains a table with the following data:

Policy Key	Claimed by	Claimant Name	Claimed From	Policy holder Name	On Date	Reason
171A064534	C0008	Dorji Phutsho	PH0001	Rajeshwar Singh	2016-08-09	Travelling to USA for...
171A001481	C0007	Priti Shekhar	PH0002	Riya L. Khurana	2018-06-21	Car Washed away b...
131A189001	C0003	Preetam Singh	PH0003	Raman Bdr Singh	2020-05-04	Death of his parents...

The Word ribbon is visible at the top, and the Windows taskbar is at the bottom.

8) If we select Policy Details:

**Figure 6.22 : All Policy Details**

A screenshot of the Eclipse IDE interface. On the left, the Project Explorer shows several projects: DBMSInsuranceServices, DBMS Project on Insurance, Project2, Tetris Project, and Try. In the center, a table titled "ALL POLICY DETAILS" displays the following data:

Policy Key	Insurance Name	Insured Date	Expiry Date
100A673312	Health	2016-01-18	2021-01-18
120A189312	Travel	2015-02-16	2018-02-16
131A189001	Life	2015-02-18	2021-02-18
151A189431	Car	2017-02-21	2022-02-21
151A134461	Car	2018-03-21	2024-02-21
171A134461	Health	2017-04-16	2023-04-16
171A001481	Car	2015-06-18	2020-06-18
171A064534	Travel	2015-07-24	2020-07-24
171A192812	Life	2015-07-16	2021-07-16
191A001722	Health	2014-08-18	2022-07-16

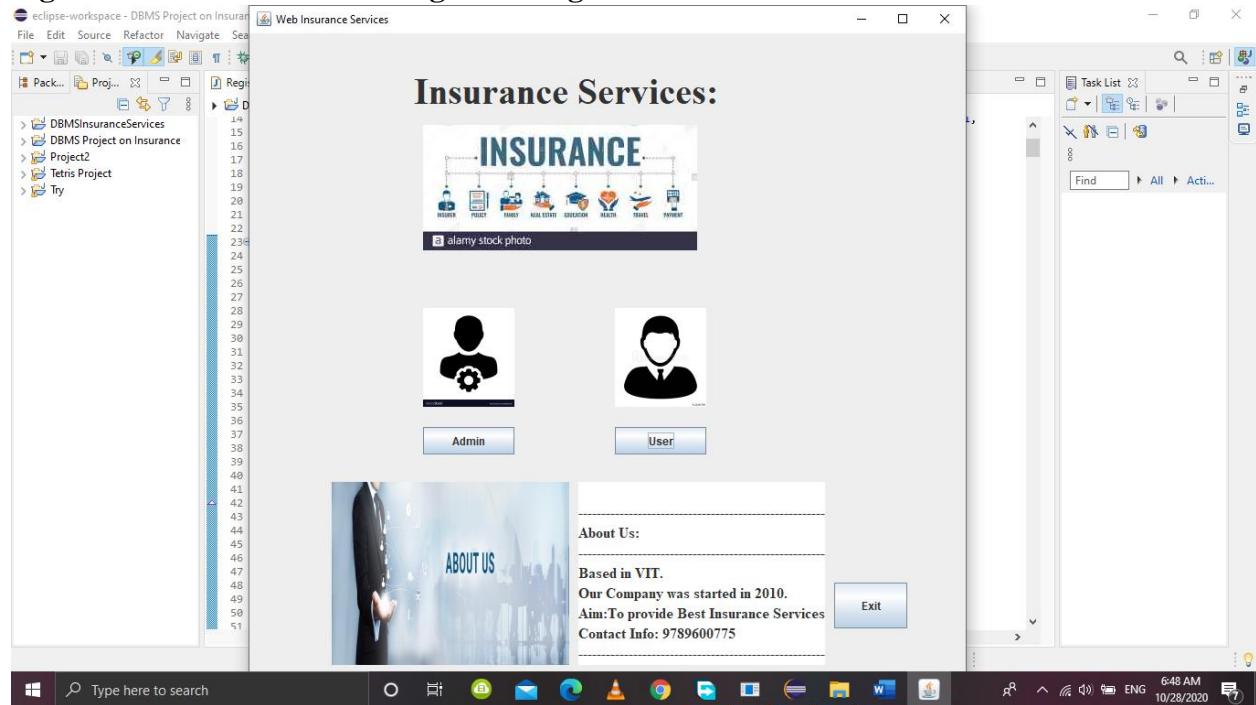
Below the table, a Java code editor shows the following code:

```
public void actionPerformed(ActionEvent e) {
    frame2 = new JFrame();
    container2 = frame2.getContentPane();
    frame2.setTitle("Admin Login Form:");
    jl_title = new JLabel("ADMIN LOGIN:");
    jl_title.setBounds(100, 40, 100, 25);
    container2.add(jl_title);
    jl_adminid = new JTextField("Admin ID: ");
    jl_adminid.setBounds(100, 60, 100, 25);
    container2.add(jl_adminid);
    jl_adminpass = new JPasswordField("Admin Pass: ");
    jl_adminpass.setBounds(100, 80, 100, 25);
    container2.add(jl_adminpass);
    jl_login = new JButton("Login");
    jl_login.setBounds(100, 100, 100, 25);
    container2.add(jl_login);
}
```

The Eclipse toolbar, Task List, and Windows taskbar are visible at the bottom.

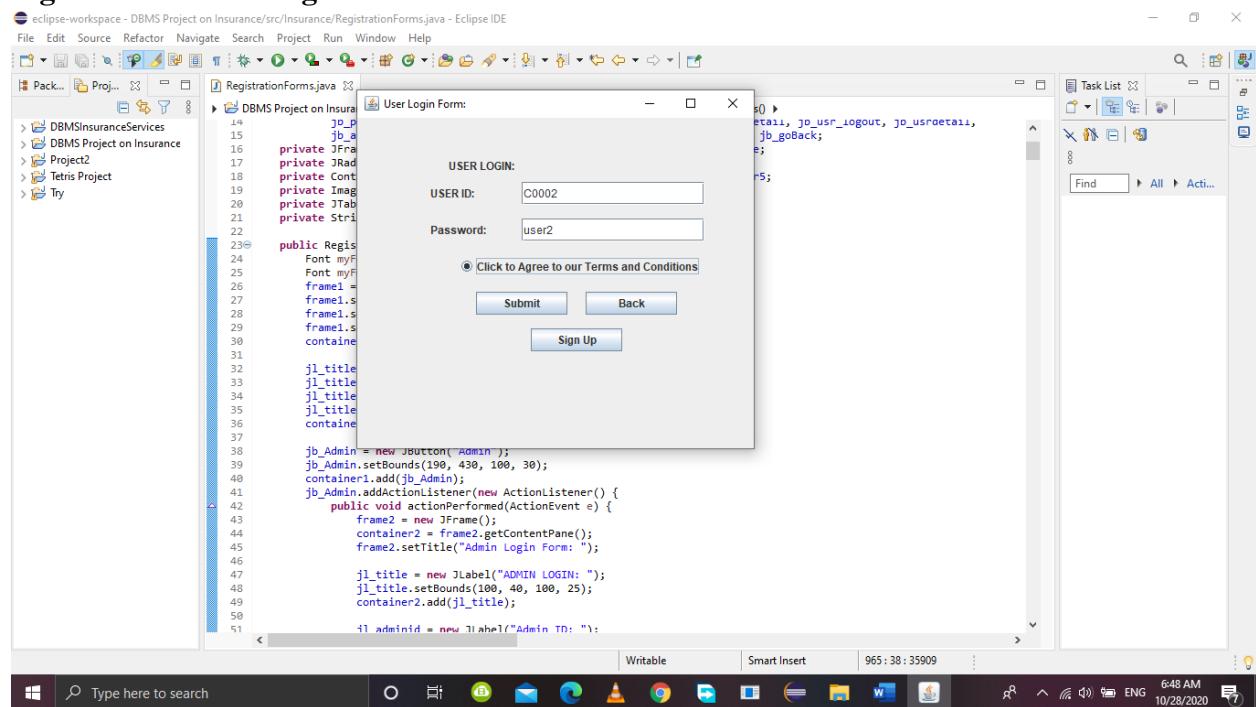
9) If we select Log Out we reach Home Page:

**Figure 6.23 : Back to Home Page after Sign out from Admin**



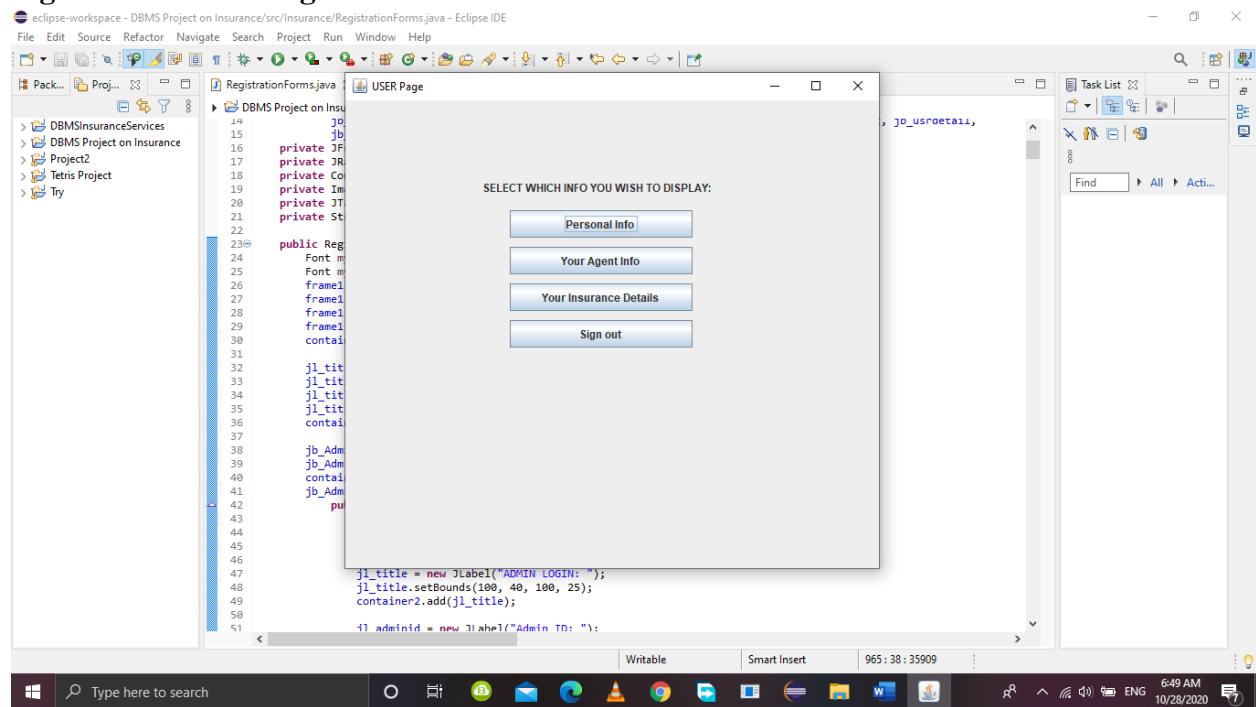
10) Selecting User :

**Figure 6.24 : User Login Form**



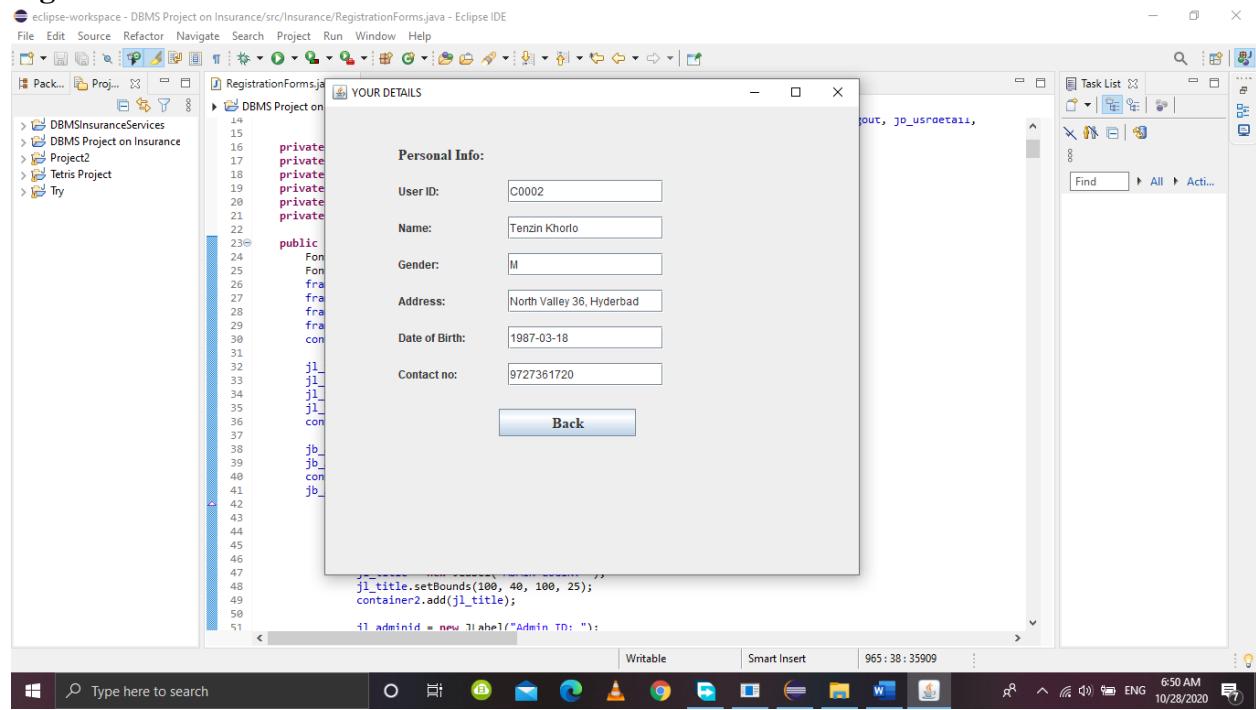
11) We get User Page after logging in:

**Figure 6.25 : User Page**



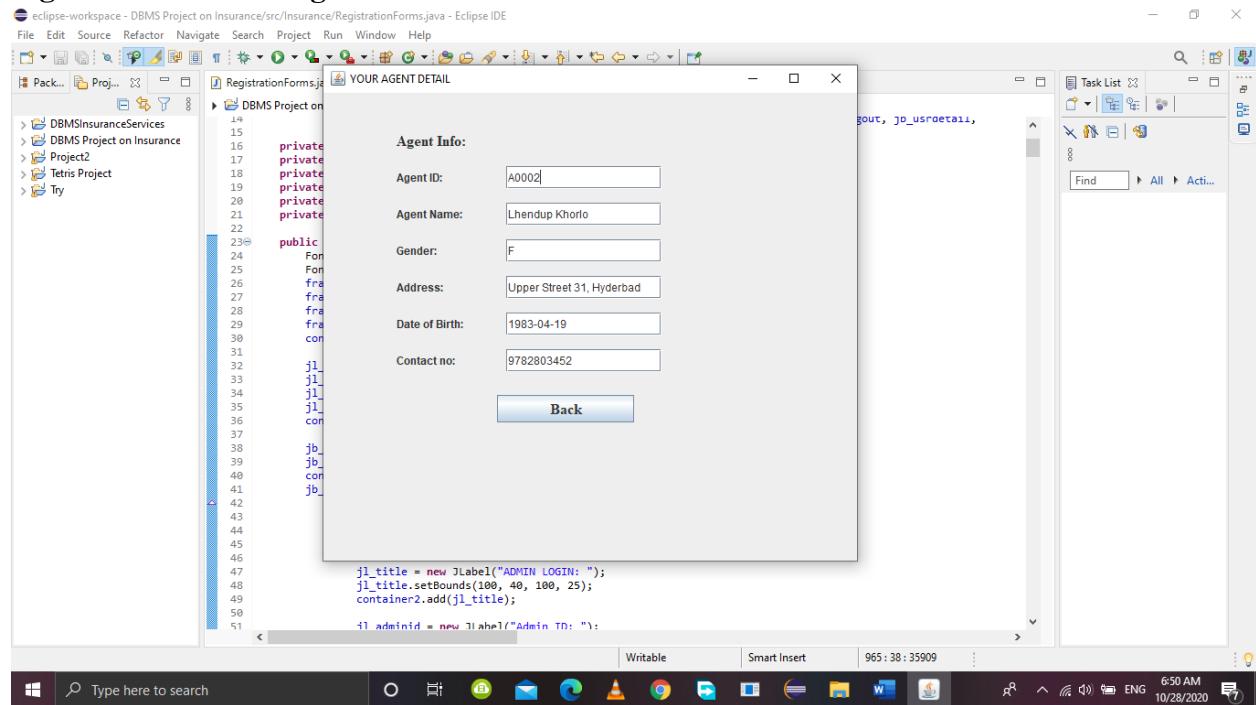
12) If we select Personal Info:

**Figure 6.26 : User Personal Info**



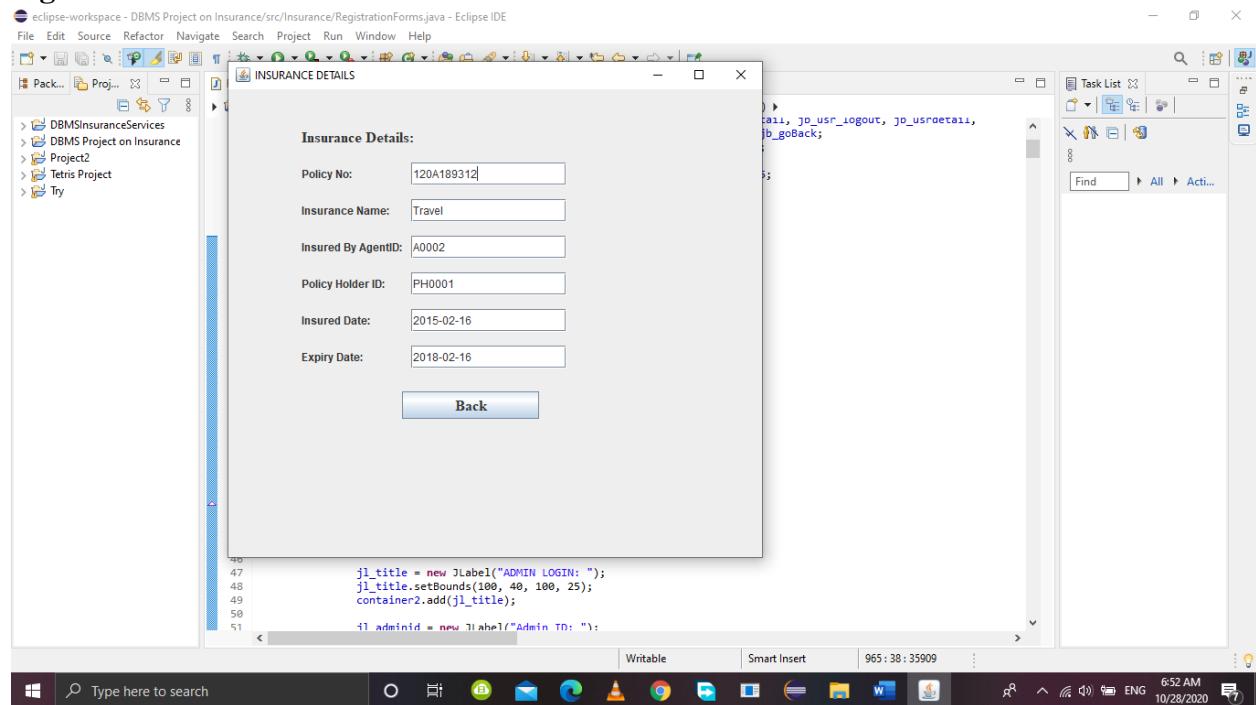
13) If we select Your Agent info:

**Figure 6.27 : User's Agent Info**



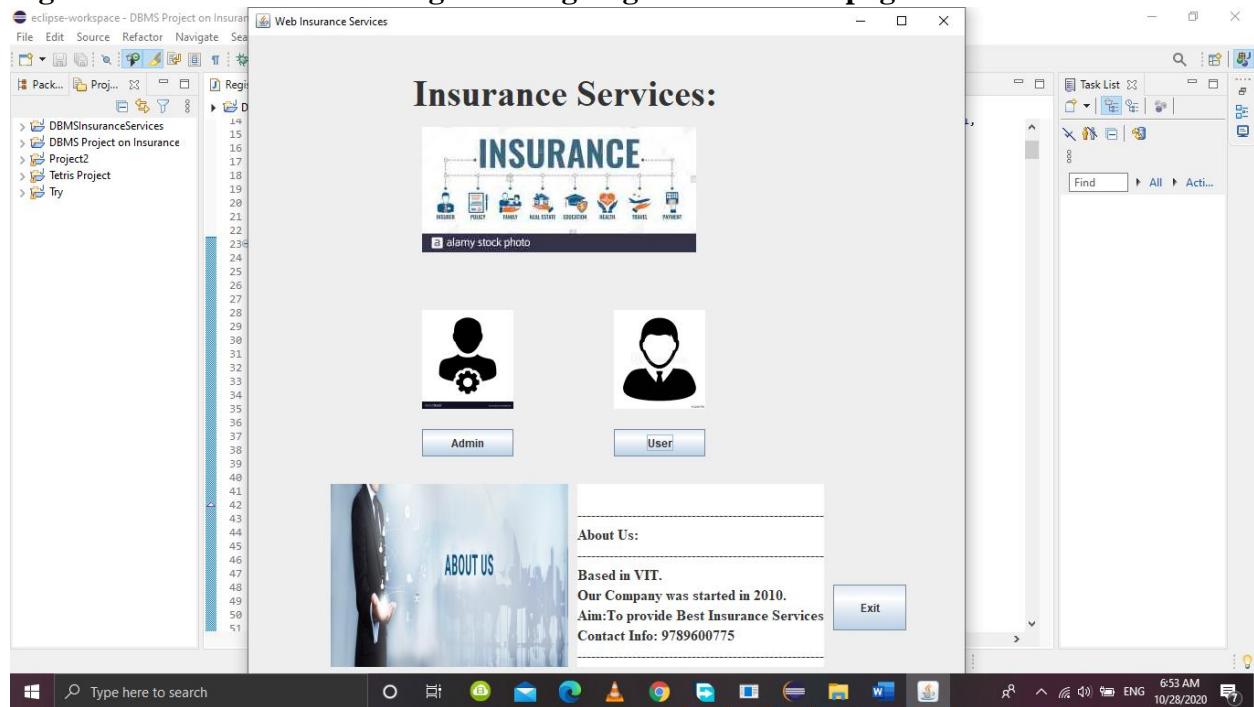
14) If we select Your Insurance Details:

**Figure 6.28 : User's Insurance Details**



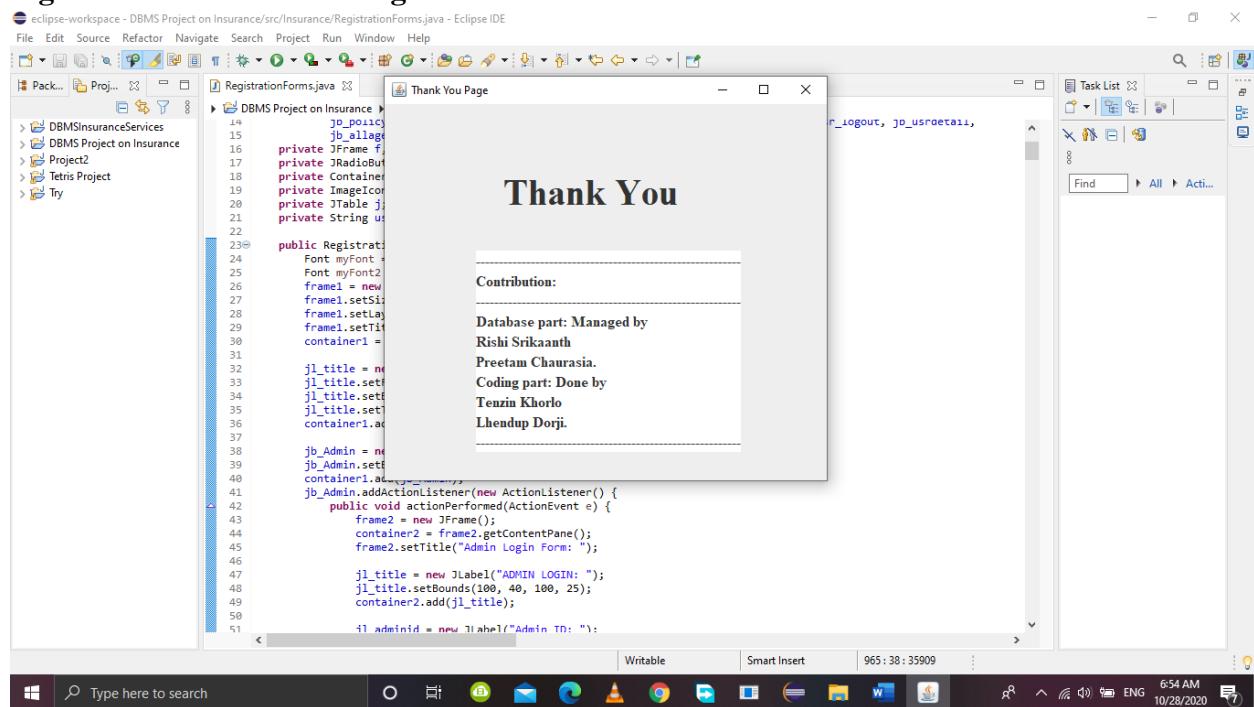
15) If we select Sign in button, we reach back to Home Page:

**Figure 6.29 : Back to Home Page after signing out from User page**



16) If we select Exit button, we get Thank you Page:

**Figure 6.30 : Thank You Page and Our Contributions**



## **CHAPTER 7**

### **7.1 CONCLUSION**

We have successfully done our project and our main objectives of keeping records of all admin, clients , agents and insurance details have been accomplished. The problems of insurance companies while trying to attend to their customers quickly and efficiently due to lack of adequate documentation of data can be easily solved by the web-based application of insurance management as we can see how efficiently we can store all the data. This application definitely ensures the insurance service communication around the globe as the customers and companies can interact virtually. On the top of that it also reduces data redundancy and facilitates faster searching of information without waste of people's time.

## 7.2 REFERENCES:

➤ From App Academia:

- 1) Ele, Sylvester I., O. A. Ofem And Obono I. Ofem. *Department of Computer Science, University of Calabar, Nigeria.* “Implementation of Futuristic Web-Based Application for Insurance Services”, 2018.
- 2) Igwe E. Kelvin. Department of Computer Science, CEO of K-tech system, Ishiagu, Nigeria. “WEB BASED APPLICATION FOR INSURANCE SERVICES CASE STUDY OF THE INSURANCE COMPANY”.

➤ From Google Chrome:

- 3) Uploaded by Nnamdi Chimaobi. Department of science. “Project on Web Based Application for Insurance Services”, on Nov 22, 2012.

Available in: <https://www.scribd.com/doc/114107737/Project-on-Web-Based-Application-for-Insurance-Services>

- 4) “Web Based Application For Insurance Services Case Study Of The Insurance Company”, by Esedebe Fidelia Ogechukwu, in July, 2013, a graduate of Computer Science of Information and Technology, Caritas University Amorji – Nike Enugu, Nigeria.

Available in: [https://www.sau.edu.ng/materials/WEB\\_BASED\\_APPLICATION\\_FOR\\_INSURANCE\\_SERVICES%20\(1\).pdf](https://www.sau.edu.ng/materials/WEB_BASED_APPLICATION_FOR_INSURANCE_SERVICES%20(1).pdf)

- 5 ) “Teach yourself SQL in 21 days, second edition ,” ,by Auwal Gene

Available in:

[http://www.auwalgene.com/@mystudents/lecturenotes/teach\\_urself\\_sql.pdf](http://www.auwalgene.com/@mystudents/lecturenotes/teach_urself_sql.pdf)

- 6) “Web Based Project Ideas & Topics” by nevon Projects, electronics |Software and mechanical kits.

Available in: <https://nevonprojects.com/web-based-project-ideas-topics/>

- 7) To create JTable in java:

<https://docs.oracle.com/javase/tutorial/uiswing/components/table.html>

- 8) to connect database with Java IDEs:

<https://www.infoworld.com/article/3388036/what-is-jdbc-introduction-to-java-database-connectivity.html>

- 9) To create JFrame, JTextArea, JTextField and all basics of java:

<https://www.javatpoint.com/java-jframe>

- 10) Java Tutorial:

<https://www.youtube.com/watch?v=r59xYe3Vyks&t=2s>

## **7.3 Appendices:**

### **Appendix I: About Eclipse IDE**

Eclipse is an integrated development environment (IDE) used in computer programming.[6] It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins, including Ada, ABAP, C, C++, C#, Clojure, COBOL, D, Erlang, Fortran, Groovy, Haskell, JavaScript, Julia, Lasso, Lua, NATURAL, Perl, PHP, Prolog, Python, R, Ruby (including Ruby on Rails framework), Rust, Scala, and Scheme. It can also be used to develop documents with LaTeX (via a TeXlipse plug-in) and packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++, and Eclipse PDT for PHP, among others.

The initial codebase originated from IBM VisualAge. The Eclipse software development kit (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Since the introduction of the OSGi implementation (Equinox) in version 3 of Eclipse, plug-ins can be plugged-stopped dynamically and are termed (OSGI) bundles.

Eclipse software development kit (SDK) is free and open-source software, released under the terms of the Eclipse Public License, although it is incompatible with the GNU General Public License. It was one of the first IDEs to run under GNU Classpath and it runs without problems under IcedTea.

### **Appendix II: About SQL**

SQL -Structured Query Language) is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS). It is particularly useful in handling structured data, i.e. data incorporating relations among entities and variables.

SQL offers two main advantages over older read–write APIs such as ISAM or VSAM. Firstly, it introduced the concept of accessing many records with one single command. Secondly, it eliminates the need to specify how to reach a record, e.g. with or without an index.

Originally based upon relational algebra and tuple relational calculus, SQL consists of many types of statements, which may be informally classed as sublanguages, commonly: a data query language (DQL), a data definition language (DDL),[b] a data control language (DCL), and a data manipulation language (DML). The scope of SQL includes data query, data manipulation (insert, update and delete), data definition (schema creation and modification), and data access control. Although SQL is essentially a declarative language (4GL), it also includes procedural elements.

SQL was one of the first commercial languages to utilize Edgar F. Codd's relational model. The model was described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks". Despite not entirely adhering to the relational model as described by Codd, it became the most widely used database language.

SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987. Since then the standard has been revised to include a larger set of features. Despite the existence of standards, most SQL code requires at least some changes before being ported to different database systems.

### **Appendix III: Codes**

One can find the full codes for this application in Git HUB in the link given Below:

<https://github.com/preetam077/DBMS-J-component.git>

Here are some figures of implemented codes:

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - DBMS Project on Insurance/src/Insurance/RegistrationForms.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbars:** Standard, Java, Database, Java Persistence API, Java Swing, Java Swing Table, Java SQL, Oracle JDBC Pool, and Java Persistence API.
- Left Sidebar (Project Explorer):** Shows the project structure: DBMSInsurance (JRE System Library), src (containing Insurance and RegistrationForms), and various files like dbdbs8.jar, Project2, Tetris Project, and Try.
- Central Editor:** Displays the Java code for `RegistrationForms.java`. The code implements `ActionListener` and contains numerous private member variables and methods, including `RegistrationForms()` and `jl_title`.
- Right Sidebar (Task List):** Shows a search bar with "Find" and "All" buttons, and a list of tasks.
- Bottom Status Bar:** Writable, Smart Insert, 21:27:1049, and system icons.

eclipse-workspace - DBMS Project on Insurance/src/Insurance/RegistrationForms.java - Eclipse IDE

```
File Edit Source Refactor Navigate Search Project Run Window Help
P RegistrationForms.java
DBMS Project on Insurance > src > Insurance > RegistrationForms > userid: String
46 jl_title = new JLabel("ADMIN LOGIN:");
47 jl_title.setBounds(100, 40, 100, 25);
48 container2.add(jl_title);
49
50 jl_adminid = new JLabel("Admin ID:");
51 jl_adminid.setBounds(80, 70, 100, 25);
52 container2.add(jl_adminid);
53
54 jl_admin_password = new JLabel("Password:");
55 jl_admin_password.setBounds(80, 110, 100, 25);
56 container2.add(jl_admin_password);
57
58 jt_adminid = new JTextField();
59 jt_adminid.setBounds(180, 70, 200, 25);
60 container2.add(jt_adminid);
61
62 jt_admin_password = new JTextField();
63 jt_admin_password.setBounds(180, 110, 200, 25);
64 container2.add(jt_admin_password);
65
66 jr_agreement_admin = new JRadioButton("Click to Agree to our Terms and Conditions");
67 jr_agreement_admin.setBounds(110, 150, 300, 25);
68 container2.add(jr_agreement_admin);
69
70 jb_admin_submit = new JButton("Log in");
71 jb_admin_submit.setBounds(130, 190, 100, 25);
72 container2.add(jb_admin_submit);
73 jb_admin_submit.addActionListener(new ActionListener() {
74     public void actionPerformed(ActionEvent e) {
75         if (jr_agreement_admin.isSelected() == false) {
76             jl_warning1 = new JLabel("Can't proceed without agreeing our conditions");
77             jl_warning1.setBounds(110, 230, 300, 25);
78             container2.add(jl_warning1);
79             frame2.repaint();
80             frame2.setVisible(true);
81         } else {
82             ...
83         }
84     }
85 })
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
```

Writable Smart Insert 21:27:1049

Type here to search 7:43 PM ENG 10/31/2020

eclipse-workspace - DBMS Project on Insurance/src/Insurance/RegistrationForms.java - Eclipse IDE

```
File Edit Source Refactor Navigate Search Project Run Window Help
P RegistrationForms.java
DBMS Project on Insurance > src > Insurance > RegistrationForms > userid: String
...
971 lastFrame.setSize(500,450);
972 lastFrame.setTitle("Thank You Page");
973 JLabel thankyou=new JLabel("Thank You");
974 thankyou.setFont(myFont);
975 thankyou.setBounds(130,70,200,50);
976 lastFrame.add(thankyou);
977 JTextArea jta=new JTextArea();
978 jta.setBounds(100, 160, 290, 220);
979 lastFrame.add(jta);
980 jta.setText("\nContribution:\n-----\n");
981 jta.setFont(myFont2);
982
983 lastFrame.setLayout(null);
984 frame1.dispose();
985 lastFrame.setVisible(true);
986
987
988 });
989
990 frame1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
991 frame1.setVisible(true);
992
993
994
995
996
997
998
999
1000 public static void main(String args[]) {
1001     EventQueue.invokeLater(() -> {
1002         new RegistrationForms();
1003     });
1004
1005
1006
1007 }
```

Writable Smart Insert 21:27:1049

Type here to search 7:44 PM ENG 10/31/2020

# THANK YOU