TABLE OF CONTENT

ABSTRACT

The purpose of the Vehicle Insurance Management System is to automate the existing manual system with the help of computerized equipment and full-fledged computer software, fulfilling their requirements so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same.

The required software and hardware are easy to work with Vehicle Insurance Management systems, as described above, can lead to error-free, secure, reliable and fast management systems. It can assist the user to concentrate on their activities rather to concentrate on the record keeping. Thus, it will help an organization in better utilization of resources. The organization can maintain computerized records without redundant entries. That means that one need not be distracted by information that is not relevant while being able to reach the information.

LIST OF TABLES

LIST OF ABBREVIATIONS

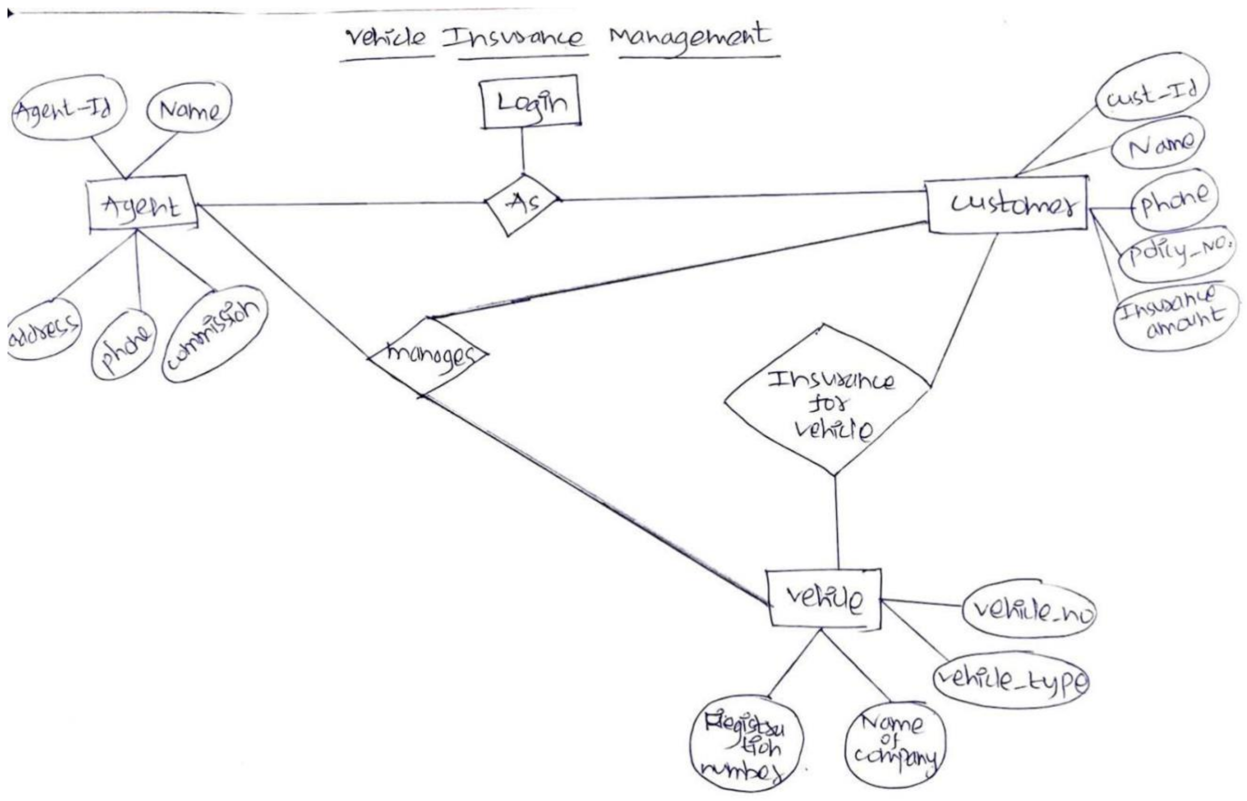
Abbreviations	**Description**
#	Primary Key of an entity
*	Normal Attribute
▭	Entity
◇	Relationship
———	Straight Relationship line
→	Relationship arrow with head
A:B	Cardinality between entities

INTRODUCTION

“The Vehicle Insurance Management System" has been developed to override the problems prevailing in the practicing manual system. This software is supported to eliminate and, in some cases, reduce the hardships faced by this existing system. Moreover, this system is designed for the particular need of the company to carry out operations in a smooth and effective manner. The application is reduced as much as possible to avoid errors while entering the data.

It also provides error message while entering invalid data. No formal knowledge is needed for the user to use this system. Thus, by this all it proves it is user-friendly. Vehicle Insurance Management System. It can assist the user to concentrate on their other activities rather concentrate on the record-keeping. Thus, it will help the organization in better utilization of resources.

ER- DIAGRAM:-



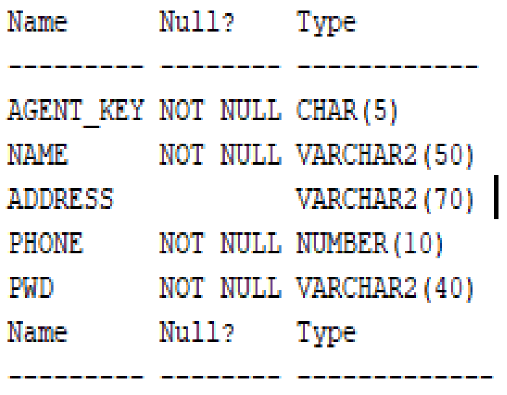
Vehicle Insurance Management
Login
Agent-Id
Name
Agent
As
address
phone
commission
manages
cust-Id
Name
customer
phone
policy_no.
Insurance amount
Insurance for vehicle
vehicle
vehicle_no
vehicle_type
Registration number
Name of company

NORMALIZED DATABASE TABLE

When the above ER diagram is converted to relational model as we move from conceptual design to logical design we get 3 relations or tables. All these tables will be in 1NF as belong to RDBMS. In all these relations only the unique attribute reserves the ability to uniquely determine all the other prime as well as non-prime attributes.

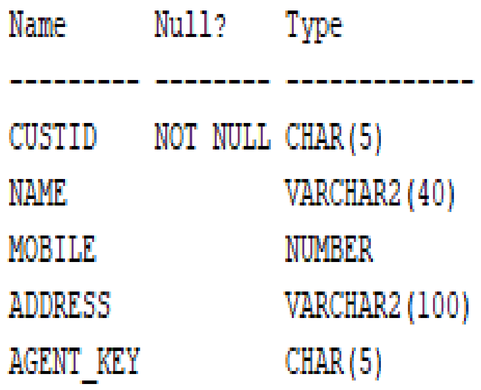
Due to this hypothesis and lack of the some theoretically backed FD's we can conclude that the relations will be in BCNF(no partial dependencies, RHS of the only FD will be candidate key, and the only RHS will also will be super key/candidate key). These relations also preserve join as they all are related with each other through one or the other common attributer.

AGENT



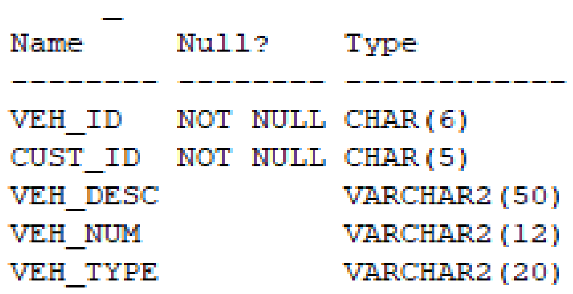
```
Name      Null?    Type
--------- -------- ------------
AGENT_KEY NOT NULL CHAR(5)
NAME      NOT NULL VARCHAR2(50)
ADDRESS            VARCHAR2(70)
PHONE     NOT NULL NUMBER(10)
PWD       NOT NULL VARCHAR2(40)
Name      Null?    Type
--------- -------- -------------
```

CUSTOMER



```
Name      Null?    Type
--------- -------- -------------
CUSTID    NOT NULL CHAR(5)
NAME               VARCHAR2(40)
MOBILE             NUMBER
ADDRESS            VARCHAR2(100)
AGENT_KEY          CHAR(5)
```

VEHICLE



```
Name      Null?    Type
--------- -------- ------------
VEH_ID    NOT NULL CHAR(6)
CUST_ID   NOT NULL CHAR(5)
VEH_DESC           VARCHAR2(50)
VEH_NUM            VARCHAR2(12)
VEH_TYPE           VARCHAR2(20)
```

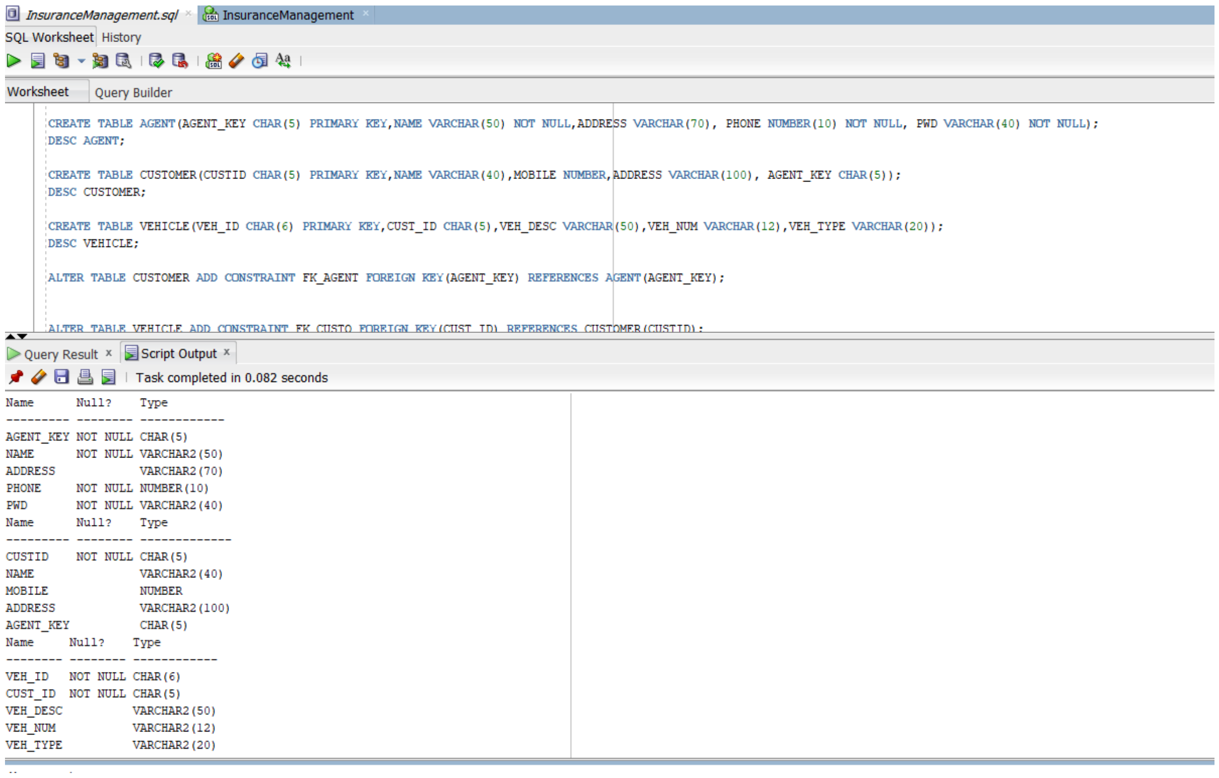
SQL QUERIES WITH RESULTS

```
select * from global_name;

select * from customers100;

alter table customers rename to customers100;

select * from customer;

select * from Agent;

select * from vehicle;

CREATE TABLE AGENT(AGENT_KEY CHAR(5) PRIMARY
KEY,NAME VARCHAR(50) NOT NULL,ADDRESS
VARCHAR(70), PHONE NUMBER(10) NOT NULL, PWD
VARCHAR(40) NOT NULL);
DESC AGENT;

CREATE TABLE CUSTOMER(CUSTID CHAR(5) PRIMARY
KEY,NAME VARCHAR(40),MOBILE NUMBER,ADDRESS
VARCHAR(100), AGENT_KEY CHAR(5));
DESC CUSTOMER;

CREATE TABLE VEHICLE(VEH_ID CHAR(6) PRIMARY
KEY,CUST_ID CHAR(5),VEH_DESC VARCHAR(50),VEH_NUM
VARCHAR(12),VEH_TYPE VARCHAR(20));
DESC VEHICLE;

ALTER TABLE CUSTOMER ADD CONSTRAINT FK_AGENT
FOREIGN KEY(AGENT_KEY) REFERENCES
AGENT(AGENT_KEY);

ALTER TABLE VEHICLE ADD CONSTRAINT FK_CUSTO
FOREIGN KEY(CUST_ID) REFERENCES CUSTOMER(CUSTID);

ALTER TABLE VEHICLE MODIFY (CUST_ID NOT NULL);
```

Insurancemanagement.ipynb

```
from tkinter import *
import os
import cx_Oracle
import random
from tkinter import messagebox
from tkinter import ttk
connectString = os.getenv('db_connect')
#con = cx_Oracle.connect('system/asdf@1251/InsuranceManagement')
#dsn_tns = cx_Oracle.makedsn('localhost', '1521',
service_name='SYS$USERS')
#dsn_tns = cx_Oracle.makedsn("oracle.sub.example.com", "1521", "xe")
#con = cx_Oracle.connect(user=r'system', password='asdf', dsn=dsn_tns)
#con='oracle://system:asdf@hostname:1521/?service_name=XE'

dsn_tns = cx_Oracle.makedsn('127.0.0.1',1521,'xe')
con = cx_Oracle.connect('system', 'asdf', dsn_tns)


def stop(root):
    root.destroy()

#Class for inserting new agent
class agent_insert:
    def __init__(self):
        top=self.top=Tk()
        top.geometry("360x360+0+0")
        top.title("Add agent")
        self.frame=Frame(top,bg='lightgreen',width=360,height=360).pack()
        self.nameins=StringVar()
        self.addrins=StringVar()
        self.passwordins=StringVar()
        self.phoneins=StringVar()

    def insert(self):
        if self.nameins.get()=='' or self.addrins.get()=='' or self.phoneins.get()==''
or self.passwordins.get()=='':
            a='Please enter correct details...'
            messagebox.showinfo("Success", a)
        else:
            self.agent_key = str(random.randint(10000, 99999))
            a = 'New agent added successfully with agent id =' + self.agent_key
            cur = con.cursor()
            statement = 'insert into agent (agent_key,name,address,phone,pwd)
values(:2,:3,:4,:5,:6)'
            cur.execute(statement, (self.agent_key, self.nameins.get(),
self.addrins.get(), self.phoneins.get(), self.passwordins.get()))
            messagebox.showinfo("Success", a)
            con.commit()
    def stop(self):
        self.top.destroy()
    def admin_page(self):
        self.top.destroy()
        Admin_Page()

class agent_login:
    def start(self,agent_key):
        top=self.top=Tk()
        self.agent_key=agent_key
        top.geometry("1280x720+0+0")
        top.title("Agent Login")
        self.frame=Frame(top,bg='lightblue',width=1280,height=720).pack()
        self.custid=StringVar()

        cur=con.cursor()
        statement = "select *  from agent where agent_key= '" + agent_key + "'  "
        cur.execute(statement)
        self.treeview.config(column=('CName', 'CMobile', 'CAddress',
'VId','VDesc','VNum','VType'))
        self.treeview.heading('CName', text='Customer Name')
        self.treeview.heading('CMobile', text='Mobile')
        self.treeview.heading('CAddress', text='Address')
        self.treeview.heading('VId', text='Vehicle ID')
        self.treeview.heading('VDesc', text='Vehicle Desc')
        self.treeview.heading('VNum', text='Vehicle Number')
        self.treeview.heading('VType', text='Vehicle Type')


    def stop(self):
        self.treeview.delete(*self.treeview.get_children())
        self.top.destroy()
    def edit_customer(self):
        self.top.destroy()
        a=edit_customer()
        a.start(self.agent_key)
        login()


#NEW-CUSTOMER
class new_customer:
    def start(self,agent_key):
        top = self.top = Tk()
        self.agent_key=agent_key
        top.geometry("1280x720+0+0")
        top.title("Add Customer")
        self.frame = Frame(top, bg='lightblue', width=1280, height=720).pack()

        self.name = StringVar()
        self.mobile = StringVar()
        self.address = StringVar()
        self.desc=StringVar()
        self.number=StringVar()
        self.type=StringVar()

        Label(self.frame, text='Name',bg='lightblue',font=('arial 10')).place(x=50,
y=150)
        Label(self.frame, text='Mobile_no',bg='lightblue',font=('arial
10')).place(x=50, y=200)
        Label(self.frame, text='Address',bg='lightblue',font=('arial
10')).place(x=50, y=250)


        Entry(self.frame, textvariable=self.name).place(x=155,y=150)
        Entry(self.frame,textvariable=self.mobile).place(x=150,y=200)
        Entry(self.frame, textvariable=self.address).place(x=150,y=250)

        Label(self.frame, text='Description',bg='lightblue',font=('arial
10')).place(x=650, y=150)
        Label(self.frame, text='Vehicle_no',bg='lightblue',font=('arial
10')).place(x=650, y=200)
        Label(self.frame, text='Type',bg='lightblue',font=('arial 10')).place(x=650,
y=250)

        Entry(self.frame, textvariable=self.desc).place(x=750, y=150)
        Entry(self.frame, textvariable=self.number).place(x=750, y=200)
        Entry(self.frame, textvariable=self.type).place(x=750, y=250)

        ttk.Button(self.frame, text="Insert",
command=self.insert).place(x=375,y=350)
        ttk.Button(self.frame, text="Back", command=self.back).place(x=600,
y=600)

    def back(self):
        self.top.destroy()
        a=agent_login()
        a.start(self.agent_key)



    def insert(self):
        if self.name.get()=='' or self.mobile.get()=='' or self.address.get()=='' or
self.desc.get()=='' or self.number.get()=='' or self.type.get()=='':
            a='Please enter correct details...'
            messagebox.showinfo("success",a)
class edit_customer:
    def start(self,agent_key):
        top=self.top=Tk()
        self.agent_key=agent_key
        top.geometry("1280x720+0+0")
        top.title("Edit Customer")
        self.frame = Frame(top, bg='#7B68EE', width=1280, height=720).pack()

        self.custid=StringVar()
        self.name = StringVar()
        self.mobile = StringVar()
        self.address = StringVar()
        self.desc = StringVar()
        self.number = StringVar()
        self.type = StringVar()

        style=ttk.Style()

style.configure("BW.TLabel",foreground="Black",background="#7B68EE")

        #EDIT FEATURES FOR CUSTOMER

        ttk.Entry(self.frame, textvariable=self.number).place(x=775, y=350)
        ttk.Button(self.frame, text="Update Vehicle
no.",command=self.edit_vehno).place(x=925, y=350)

        ttk.Entry(self.frame, textvariable=self.type).place(x=775, y=450)
        ttk.Button(self.frame, text="Update Vehicle
type",command=self.edit_type).place(x=925, y=450)

        ttk.Button(self.frame, text="Back",
command=self.back).place(x=600,y=600)

        top.mainloop()

    def back(self):
        self.top.destroy()
        a=agent_login()
        a.start(self.agent_key)

    def edit_name(self):
        a = 'Info edited successfully'
        cur = con.cursor()
        statement = "UPDATE customer SET NAME=:2  WHERE custid=:1"
        cur.execute(statement, (self.name.get(), self.custid.get()))
        messagebox.showinfo("Success", a)
        con.commit()
        con.commit()

#Class for new admin page
class Admin_Page:
    def __init__(self):
        top=self.top=Tk()
        top.geometry("1280x720+0+0")
        top.title("Admin Login")
        top.resizable(False,False)


self.left=Frame(top,width=800,height=720,bg="lightpink").pack(side=LEFT)
        self.right = Frame(top, width=480, height=720,
bg="lightpink").pack(side=LEFT)


        self.agent_key=StringVar()
        self.name=StringVar()
        self.phone=StringVar()
        for i in a:
            (key, name, add, no, pwd) = i;
            self.treeview.insert('', 'end', key, text=key)
            self.treeview.set(key, 'Name', name)
            self.treeview.set(key, 'Address', add)
            self.treeview.set(key, 'Phone', no)
            self.treeview.set(key, 'Password', pwd)

        top.mainloop()
    def new_agent(self):
        self.stop()
        agent_insert()
    def logout(self):
        self.stop()
        login()

    def delete(self):
        a = 'Record deleted successfully with agent id =' +
str(self.agent_key.get())+' Please Login again....!'
        cur = con.cursor()
        statement = "delete from agent where agent_key= '" + self.agent_key.get()
        cur.execute(statement)
        con.commit()
        messagebox.showinfo("Success", a)
        self.logout()
        con.commit()

class login:
    def __init__(self):
        top=self.top=Tk()
        top.title('LOGIN')
        top.geometry('480x360+0+0')
        top.resizable(False,False)


self.left=Frame(top,width=240,height=360,bg="lightpink").pack(side=LEFT)

self.right=Frame(top,width=240,height=360,bg="lightblue").pack(side=RIGH)

        ttk.Button(self.right, text="LOGIN",
command=self.agent_login).place(x=325, y=180)

        self.agent_key=StringVar()
        self.agentpwd=StringVar()
        top.mainloop()

    def admin_login(self):
        if self.adminID.get()=='' and self.adminpwd.get()=='':
            self.stop()
            Admin_Page()
        else:
            messagebox.showerror('Error','Invalid Credentials')
    def stop(self):
        self.top.destroy()

    def agent_login(self):
        dsn_tns = cx_Oracle.makedsn('127.0.0.1',1521,'xe')
        con = cx_Oracle.connect('system', 'asdf', dsn_tns)

        cur = con.cursor()
        statement = "select * from agent where agent_key=:1 and pwd=:2 "
        cur.execute(statement,(self.agent_key.get(),self.agentpwd.get()))
        a = cur.fetchall()
        if len(a)==0:
            messagebox.showerror('Error','Enter valid login credentials')
        else:
            self.stop()
            a=agent_login()
            a.start(self.agent_key.get())

        Button(root, text='click here to continue', background="#66c7df",
font=('arial 14'),command=self.login).place(x=500, y=10)
        root.mainloop()
    def login(self):
        self.root.destroy()
        login()

start()
```

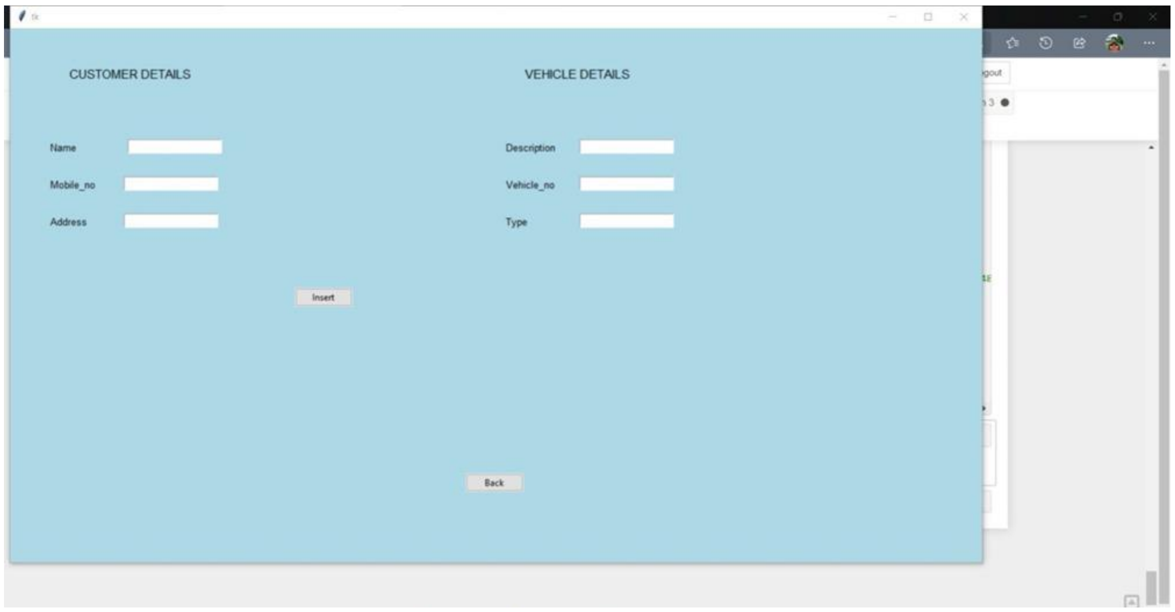
OUTPUT

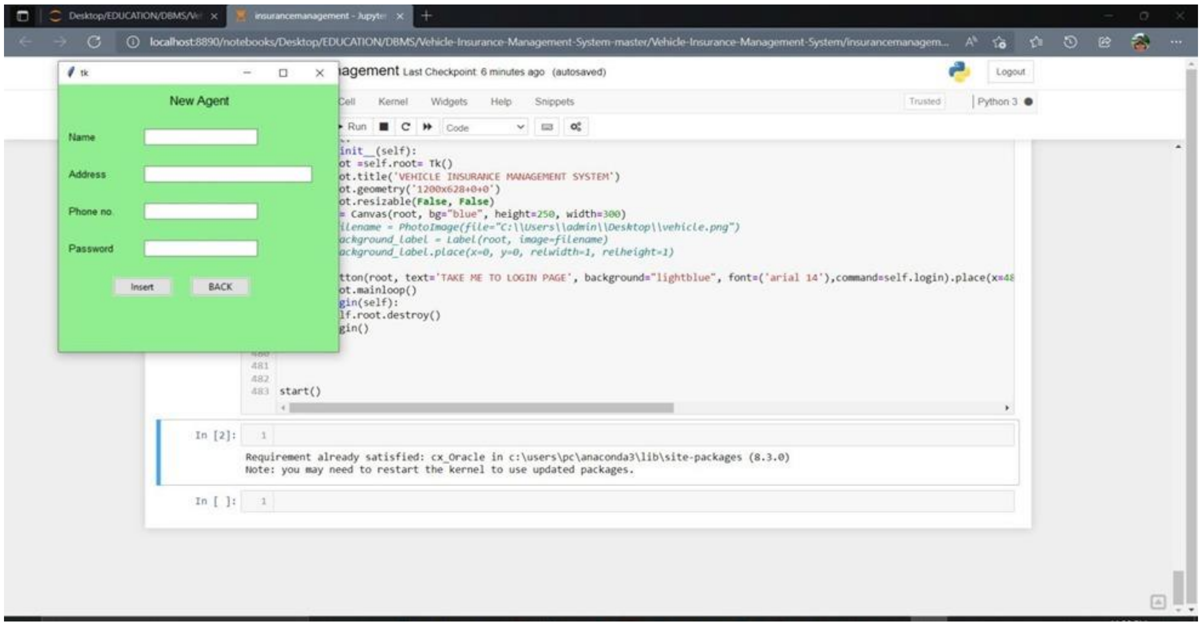


```
CREATE TABLE AGENT(AGENT_KEY CHAR(5) PRIMARY KEY,NAME VARCHAR(50) NOT NULL,ADDRESS VARCHAR(70), PHONE NUMBER(10) NOT NULL, PWD VARCHAR(40) NOT NULL);
DESC AGENT;

CREATE TABLE CUSTOMER(CUSTID CHAR(5) PRIMARY KEY,NAME VARCHAR(40),MOBILE NUMBER,ADDRESS VARCHAR(100), AGENT_KEY CHAR(5));
DESC CUSTOMER;

CREATE TABLE VEHICLE(VEH_ID CHAR(6) PRIMARY KEY,CUST_ID CHAR(5),VEH_DESC VARCHAR(50),VEH_NUM VARCHAR(12),VEH_TYPE VARCHAR(20));
DESC VEHICLE;

ALTER TABLE CUSTOMER ADD CONSTRAINT FK_AGENT FOREIGN KEY(AGENT_KEY) REFERENCES AGENT(AGENT_KEY);


ALTER TABLE VEHICLE ADD CONSTRAINT FK_CUSTO FOREIGN KEY(CUST_ID) REFERENCES CUSTOMER(CUSTID);
```

Task completed in 0.082 seconds

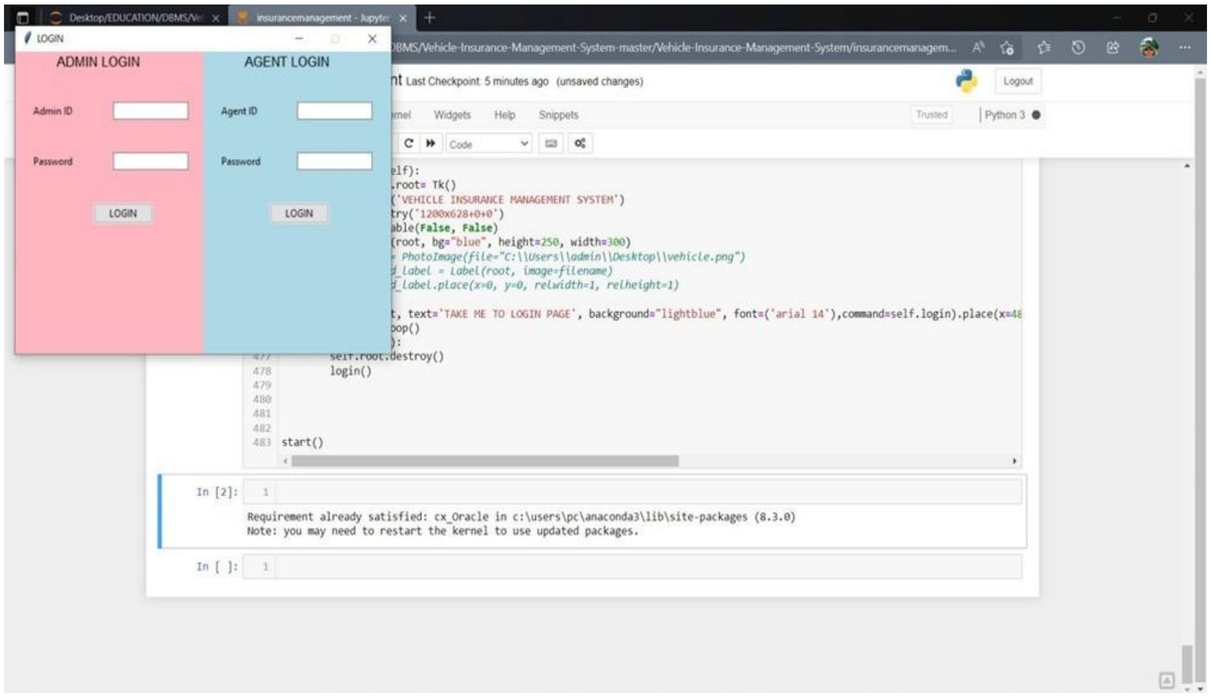
```
Name      Null?    Type
--------- -------- ------------
AGENT_KEY NOT NULL CHAR(5)
NAME      NOT NULL VARCHAR2(50)
ADDRESS            VARCHAR2(70)
PHONE     NOT NULL NUMBER(10)
PWD       NOT NULL VARCHAR2(40)
Name      Null?    Type
--------- -------- -------------
CUSTID    NOT NULL CHAR(5)
NAME               VARCHAR2(40)
MOBILE             NUMBER
ADDRESS            VARCHAR2(100)
AGENT_KEY          CHAR(5)
Name     Null?    Type
-------- -------- ------------
VEH_ID   NOT NULL CHAR(6)
CUST_ID  NOT NULL CHAR(5)
VEH_DESC          VARCHAR2(50)
VEH_NUM           VARCHAR2(12)
VEH_TYPE          VARCHAR2(20)
```





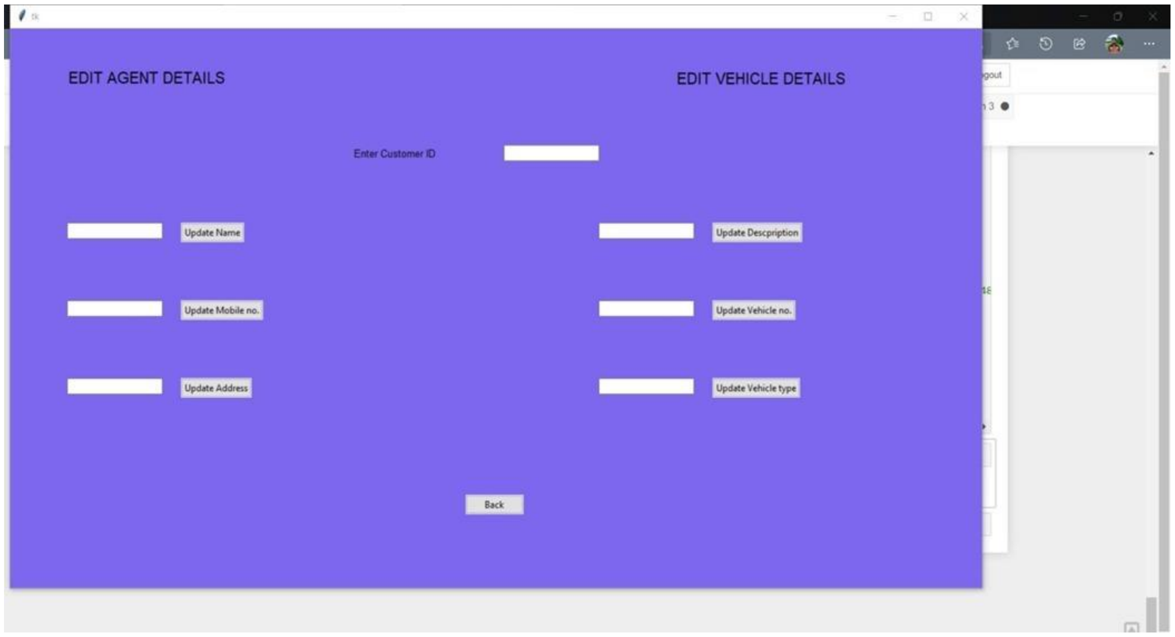
```
init__(self):
ot =self.root= Tk()
ot.title('VEHICLE INSURANCE MANAGEMENT SYSTEM')
ot.geometry('1200x628+0+0')
ot.resizable(False, False)
= Canvas(root, bg="blue", height=250, width=300)
ilename = PhotoImage(file="C:\\Users\\admin\\Desktop\\vehicle.png")
ackground_label = Label(root, image=filename)
ackground_label.place(x=0, y=0, relwidth=1, relheight=1)

tton(root, text='TAKE ME TO LOGIN PAGE', background="lightblue", font=('arial 14'),command=self.login).place(x=48
ot.mainloop()
gin(self):
lf.root.destroy()
gin()



start()
```

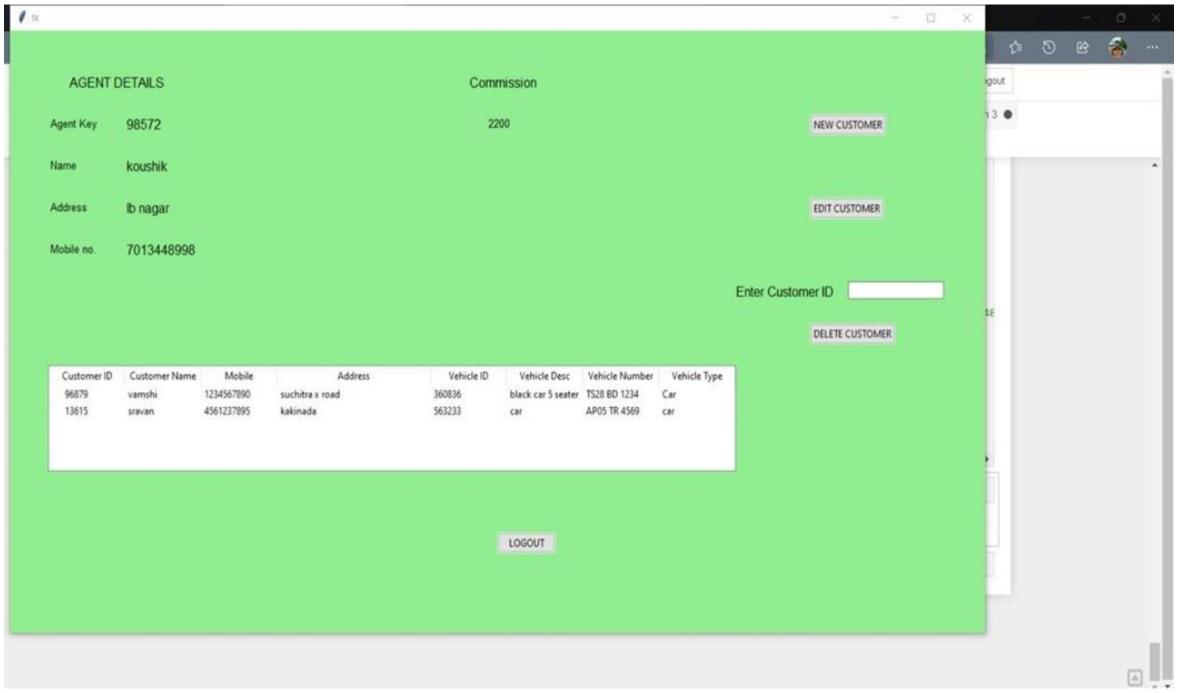
```
Requirement already satisfied: cx_Oracle in c:\users\pc\anaconda3\lib\site-packages (8.3.0)
Note: you may need to restart the kernel to use updated packages.
```



```
elf):
.root= Tk()
('VEHICLE INSURANCE MANAGEMENT SYSTEM')
try('1200x628+0+0')
able(False, False)
(root, bg="blue", height=250, width=300)
= PhotoImage(file="C:\\Users\\admin\\Desktop\\vehicle.png")
d_label = Label(root, image=filename)
d_label.place(x=0, y=0, relwidth=1, relheight=1)

t, text='TAKE ME TO LOGIN PAGE', background="lightblue", font=('arial 14'),command=self.login).place(x=48
oop()
):
        self.root.destroy()
        login()




start()
```

```
Requirement already satisfied: cx_Oracle in c:\users\pc\anaconda3\lib\site-packages (8.3.0)
Note: you may need to restart the kernel to use updated packages.
```





Agent Key 98572

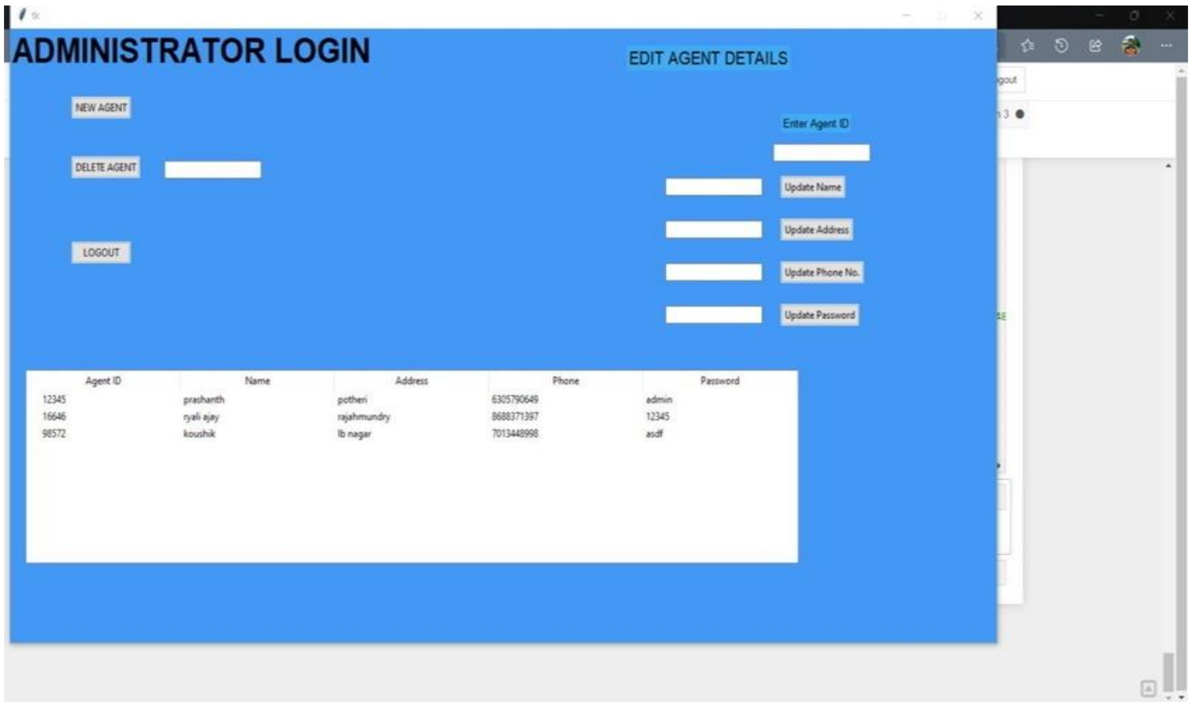
Name koushik

Address lb nagar

Mobile no. 7013448998

2200

Customer ID	Customer Name	Mobile	Address	Vehicle ID	Vehicle Desc	Vehicle Number	Vehicle Type
96879	vamshi	1234567890	suchitra x road	360836	black car 5 seater	TS28 BD 1234	Car
13615	sravan	4561237895	kakinada	563233	car	AP05 TR 4569	car



Agent ID	Name	Address	Phone	Password
12345	prashanth	potheri	6305790649	admin
16646	ryali ajay	rajahmundry	8688371397	12345
98572	koushik	lb nagar	7013448998	asdf

BENEFITS OF VEHICLE INSURANCE MANAGEMENT SYSTEM

A Vehicle Insurance Management System can offer several benefits, including:

1. Efficient Data Management: A Vehicle Insurance Management System can help manage large amounts of data related to vehicle insurance policies, claims, and other relevant information. This can help reduce errors and improve efficiency in managing insurance-related information.

2. Streamlined Claims Processing: With a Vehicle Insurance Management System, insurers can process claims more quickly and efficiently, reducing the time it takes to settle claims. This can help improve customer satisfaction and loyalty.

3. Improved Customer Service: Vehicle Insurance Management Systems can provide customers with real-time access to their insurance policy details, claim status, and other relevant information. This can help improve customer satisfaction by providing timely and accurate information.

4. Risk Management: A Vehicle Insurance Management System can help insurers manage risks associated with insuring vehicles. By analyzing data related to driving behavior, accident statistics, and other factors, insurers can better understand the risks associated with insuring a particular vehicle or driver.

5. Cost Savings: By automating and streamlining insurance-related processes, a Vehicle Insurance Management System can help reduce costs associated with claims processing, customer service, and other activities.

Overall, a Vehicle Insurance Management System can help insurers improve efficiency, reduce costs, and provide better customer service.

CONCLUSION

An Vehicle insurance management system has been developed and it was tested with sample data. The system results in regular timely preparations of required outputs. In comparison with manual system the benefits under a computer system are considerable in the saving of man power working hours and Effort. Provision for addition, updating and deletion of customers is there in the system .It is observed that proper filing system has been adopted for future reference. The system can be used to make better management described at appropriate time. The user gets timely information.

FUTURE SCOPE

The system may be further updated or modified at will owing to its simple structure. We can further add a transaction entity which will look after the payments made by the customer towards their policy. Depending on future requirements more changes can be made owing to the organization’s need.

REFERNCES:-

Technology Indusrty Outlook Survey, "https://www.kpmg.com/US/en/IssuesAndInsights/ArticlesPublications/Documents/insurance- industry-2013- outlook-survey.pdf"

2010 Online Auto Insurance report,

"http://www.kpmg.com/ZA/en/IssuesAndInsights/Article sPublications/Financial-Services/Documents/InsuranceIndustry-Survey-2013.pdf"

www.comscore.com/content/.../Auto+Insurance+Report

+Abstract.pdf

A New Beginning: Online Insurance Trend, Asha Bhandarkar, "www.insuringindia.com/documents/InsuringIndiaOnli neTrends.pdf"