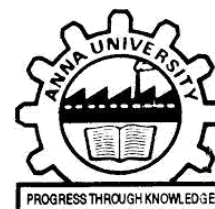


PAYROLL MANAGEMENT SYSTEM



A PROJECT REPORT

Submitted by

ABISHEAK .A	(202109004)
MOHAMED ASLAM.K	(202109034)
KARPAGA GANESH	(202109027)

In partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

**MEPCO SCHLENK COLLEGE OF ENGINEERING,
SIVAKASI-626123**

(An Autonomous Institution)

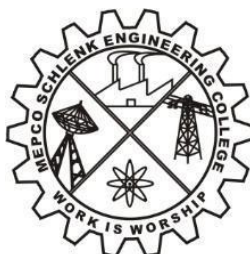
ANNA UNIVERSITY: CHENNAI 600 025

DECEMBER 2022

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

AUTONOMOUS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATASCIENCE



BONAFIDE CERTIFICATE

Certified that this project report “**PAYROLL MANAGEMENT SYSTEM**” is the bonafide work of **MOHAMED ASLAM.K (202109034)**, **ABISHEAK.A (202109004)**, **KARPAGA GANESH.B (202109027)** who carried out the project work under my supervision.

SIGNATURE

Dr.A.Shenbagarajan, M.E.,Ph.D
Mrs.L.Prasika. M.E.,(Ph.D)
Assistant Professor(SG)
Artificial Intelligence and Data Science
Mepco Schlenk Engineering College

SIGNATURE

Dr.J.AngelaJennifaSujana, M.E.,Ph.D
Associate Professor(SG) & Head
Artificial Intelligence and Data Science
Mepco Schlenk Engineering College,
Sivakasi- 626 005, Virudhunagar

The project report submitted for the viva voce held on

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First and foremost we **praise and thank “The Almighty”**, the lord of all creations, who by his abundant grace has sustained us and helped us to work on this project successfully.

We really find unique pleasure and immense gratitude in thanking our respected management members, who is the backbone of our college.

A deep bouquet of thanks to respected Principal **Dr.S.Arivazhagan M.E.,Ph.D.**, for having provided the facilities required for our mini project.

We sincerely thank our Head of the Department **Dr. J. Angela JennifaSujana M.E.,Ph.D.**, Associate Professor(SG) & Head, Department of Artificial Intelligence and Data Science, for her guidance and support throughout the mini project .

We also thank our guide **Dr.A.Shenbagarajan.,M.E.,Ph.D.**, Assistant Professor(SG), **Mrs.L.Prasika., M.E(Ph.D)**, Assistant Professor Department of Artificial Intelligence and Data Sciencefor their valuable guidance and it is great privilege to express our gratitude to them.

Weextremely thank our project coordinator **Dr.A.Shenbagarajan.,M.E.,Ph.D.**, Assistant Professor(SG),**Mrs.L.Prasika., M.E(Ph.D)**, Assistant Professor Department of Artificial Intelligence and Data Science, who inspired us and supported us throughout the mini project.

We extend our heartfelt thanks and profound gratitude to all the faculty members of Artificial Intelligence and Data Science department for their kind help during our mini project work.

We also thank our parents and our friends who had been providing us with constant support during the course of the mini project work.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	5
	LIST OF FIGURES	6
1	INTRODUCTION	7
	1.1 Scope of the project	7
	1.2 Objective of the project	8
	1.3 Review Summary	8
2	MODULES	9
	2.1 Introduction	9
	2.1.1 Connectivity	10
	2.1.1.1 Pymysql	11
	2.1.2 Graphical User Interface	11
	2.1.2.1 Tkinter	10
	2.1.2.2 Notebook	11
	2.1.2.3 Tree view	11
3	DATABASE DESIGN	12
	3.1 Introduction	12
	3.1.1 Data base connectivity	12
	3.1.1.1 Data insertion	13
	3.1.1.2 Data deletion	15
	3.1.1.3 Data updation	15
	3.2 Schema	16
	3.2.1 Table creation	16
	3.3 ER-Diagram	17
	3.4 Schema Diagram	18
	3.5 Flow Diagram	19
	3.6 Dependencies	19
	3.7 Normalization	20

4	SOURCE CODE	21
5	RESULTS	33
6	CONCLUSION	34

ABSTRACT

The payroll management process is one of the most repetitive and annoying processes to be done. Mistakes are very common, and if they aren't eliminated, the morale of the employees could be negatively affected. Using payroll management software minimizes this possibility. In case there's a sudden system failure or irreparable damage to the hardware, backups of the payroll records are imperative. Payroll management software can also provide the company with a graphical representation of the company's salary. This is very important when financial assessments need to be made. For example, when wages need to be increased or decreased, calculations can be performed to educate the company as to how this adjustment will impact the company's finances. A significant amount of time is spent managing staff and corporate income taxes. Payroll management software also helps in saving precious time. This time can then be better spent on other corporate activities instead. Over time, bonuses, vacation, and the varying deductions and taxes that need to be made on an individual basis can also be performed by the payroll management software. Security for the payroll database is ensured to be top-quality. It's important to keep the staff data secure and confidential. , various administrative rights and privileges thus exist to gain access to this payroll database. Employees are also allowed to access the breakdowns of their remunerations and may choose to debate on them as well if they wish to do so. Finally, modern payroll management systems are very cost-effective. Many make use of cloud technology as well. In such cases, the organization would only require an active subscription to continue accessing its database. This reduces the need to invest in expensive hardware to manage payrolls. Furthermore, the company can easily reduce or expand their requirements as and when they want.

LIST OF FIGURES

FIGURE NO	NAME OF THE FIGURES	PAGE NO
2.1.1.1	Import Pymysql module	10
2.1.2	Graphical User Interface	11
2.1.2.3	Tree view	12
3.1.1.1	Code for Data insertion	14
3.1.1.1(1)	Database	15
3.1.1.2	Code for Data deletion	16
3.1.1.3	Code for Data updation	16
3.2 .1	Table creation	17
3.3	ER Diagram	18
3.4	Schema Diagram	19
3.5	Flow Diagram	20
5.1	Result 1	34
5.2	Result 2	35

CHAPTER 1

INTRODUCTION

A payroll management system is the process by which employers pay wages to their employees. It's also how they demonstrate their commitment to their workers, fulfill their obligations to government agencies and keep financial records in order.

Payroll management is an important part of any business because it helps improve employee engagement and regulatory compliance. Without an efficient, accurate means of paying employees, depositing and filing taxes, and maintaining records, employers could face wage claims and expensive penalties.

Payroll management is the administrative task of compensating employees for services rendered. It also provides a financial record of employee gross earnings, payroll deductions and net pay, as well as the employer's related payroll tax liability.

Some benefits of payroll are Automation helps ensure that wage calculations, payroll deductions and tax payments are correct, Some providers of payroll software notify employers about changes in employment or tax regulations that might affect their business, In addition to traditional paychecks, payroll software can usually accommodate direct deposit and alternative payment methods, like pay cards, Many digital payroll systems create employee profiles with various payroll records that can be aggregated and downloaded to meet regulatory requirements.

Employers must calculate and withhold federal, state and local taxes from employee wages. The exact amounts are based on current tax rates and each employee's Form W-4, Employee's Withholding Certificate and state or local withholding certificates. Employers are also required to match employee contributions to Social Security and Medicare, otherwise known as Federal Insurance Contribution Act (FICA)

taxes, and may be solely responsible for federal and state unemployment taxes. Some states, however, require employees to contribute to state unemployment, as well as disability and paid family leave programs through payroll deductions. All payments must be sent to government agencies by the specified deadlines.

SCOPE OF THE PROJECT

The main scope of the project is to deal with the financial aspects of employee's Salary, Deductions, allowances, Net pay.

OBJECTIVE OF THE PROJECT

An equally important objective of any payroll system is to **generate paychecks and pay stubs**. Computerizing this task saves managers and business owners time and money; rather than reviewing time cards manually and calculating a check amount based on the hours worked, the payroll system should be designed to automatically calculate these figures.

REPORT SUMMARY

This project organized as follows:chapter-1 deals with introduction about the project scope of the project,main objective it deals with,chapter-2 deals with connectivity with database using pymysql and graphical user interface using tkinter and modules used,chapter-3 deals with backend of the project and table view,entity relationship diagram,schema diagram of the database and normalized table view and dependencies,Source code deals with actual working code of the project,Result delas with end result of the graphical user interface and connection with database.

CHAPTER 2

MODULES

INTRODUCTION

A Python module is a file containing Python definitions and statements. A module can define functions, classes, and variables. A module can also include runnable code. Grouping related code into a module makes the code easier to understand and use. It also makes the code logically organized.

CONNECTIVITY

PyMySQL is an interface for connecting to a MySQL database server from Python. It implements the Python Database API v2. 0 and contains a pure-Python MySQL client library. The goal of PyMySQL is to be a drop-in replacement for MySQLdb.

PYMYSQL

PyMySQL is an interface for connecting to a MySQL database server from Python. It implements the Python Database API v2. 0 and contains a pure-Python MySQL client library. The goal of PyMySQL is to be a drop-in replacement for MySQLdb.

```
from tkinter import*
from tkinter import ttk #tableview,notebook
import random
import tkinter.messagebox
import datetime
import time
import tkinter.ttk as tkrtk
import tkinter as tt
import pymysql
```

Fig. 2.1.1.1 Import PyMysql

Fig 2.1.1.1 shows the pyMysql module and some other modules that are used in GUI
GRAPICAL USER INTERFACE

A graphical user interface (GUI) is an interface that is drawn on the screen for the user to interact with. User interfaces have some common components: Main window. Menu.

Fig. 2.1.2 Graphical User Interface

Fig 2.1.2 shows the Graphical User Interface for the source code mentioned bellow

TKINTER

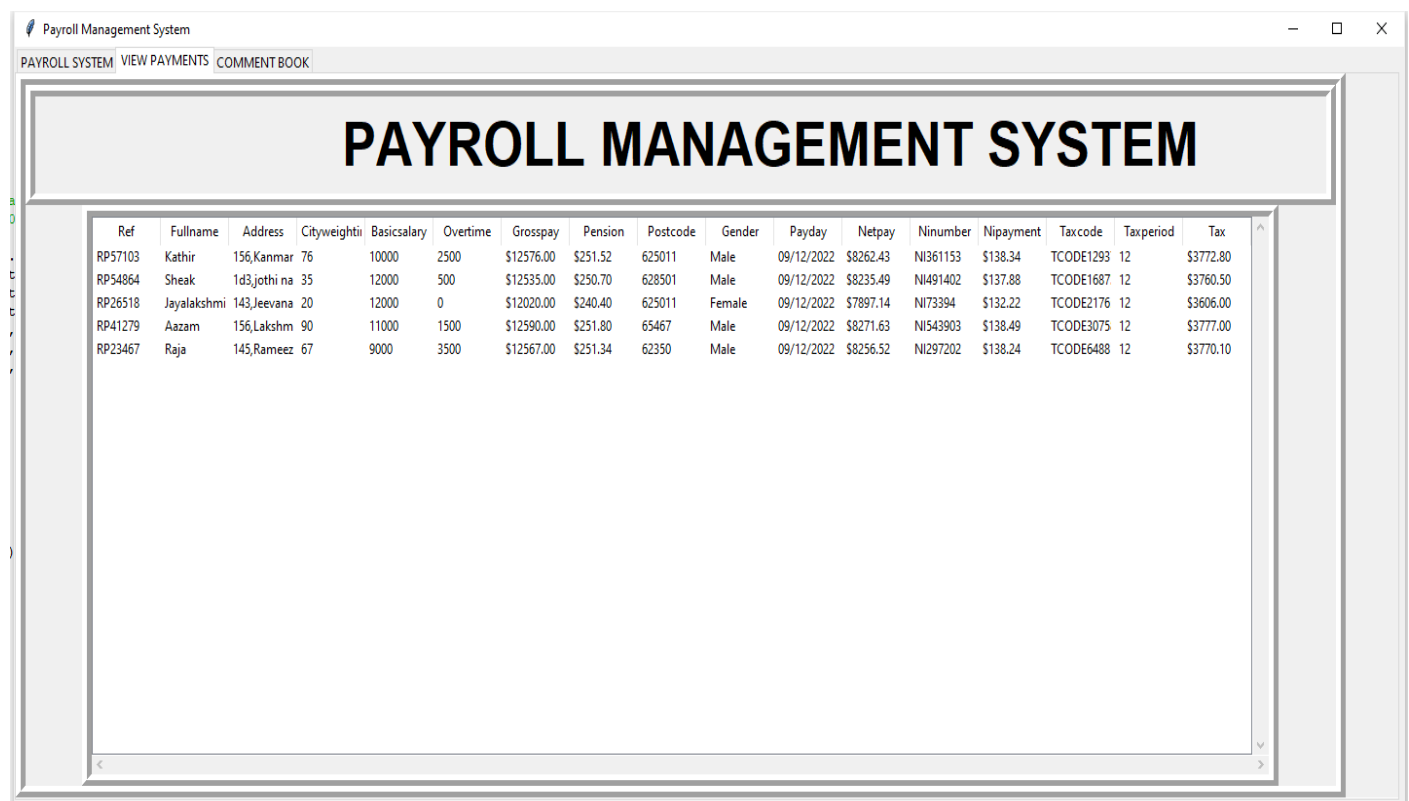
Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Fig 2.1.1.1 the Tkinter module is imported.

NOTEBOOK

Notebook widget is an inbuilt widget of **ttk** library in tkinter. It enables the user to create Tabs in the window application. Tabs are generally used to separate the workspace and specialize the group of operations in applications at the same time.

TREE VIEW

Tkinter Treeview widget is used to display the data in a hierarchical structure. In this structure, each row can represent a file or a directory. Each directory contains files or additional directories. If we want to create a Treeview widget, then we can use `Treeview(parent, columns)` constructor to build the table.



The screenshot shows a window titled "Payroll Management System" with three tabs: "PAYROLL SYSTEM", "VIEW PAYMENTS", and "COMMENT BOOK". The "PAYROLL SYSTEM" tab is active, displaying a table with the following data:

Ref	Fullname	Address	Cityweight	Basic salary	Overtime	Gross pay	Pension	Postcode	Gender	Payday	Net pay	Ninumber	Nipayment	Taxcode	Taxperiod	Tax
RP57103	Kathir	156,Kanmar	76	10000	2500	\$12576.00	\$251.52	625011	Male	09/12/2022	\$8262.43	NI361153	\$138.34	TCODE1293	12	\$3772.80
RP54864	Sheak	1d3,jothi na	35	12000	500	\$12535.00	\$250.70	628501	Male	09/12/2022	\$8235.49	NI491402	\$137.88	TCODE1687	12	\$3760.50
RP26518	Jayalakshmi	143,Jeevana	20	12000	0	\$12020.00	\$240.40	625011	Female	09/12/2022	\$7897.14	NI73394	\$132.22	TCODE2176	12	\$3606.00
RP41279	Aazam	156,Lakshm	90	11000	1500	\$12590.00	\$251.80	65467	Male	09/12/2022	\$8271.63	NI543903	\$138.49	TCODE3075	12	\$3777.00
RP23467	Raja	145,Rameez	67	9000	3500	\$12567.00	\$251.34	62350	Male	09/12/2022	\$8256.52	NI297202	\$138.24	TCODE6488	12	\$3770.10

Fig. 2.1.2.3 Tree view

Fig. 2.1.2.3 is the image of the tkinter tree view widget for the source code

CHAPTER 3

DATABASE DESIGN

INTRODUCTION

Database Design is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems. Properly designed database are easy to maintain, improves data consistency and are cost effective in terms of disk storage space. The database designer decides how the data elements correlate and what data must be stored.

The main objectives of database design in DBMS are to produce logical and physical designs models of the proposed database system.

The logical model concentrates on the data requirements and the data to be stored independent of physical considerations. It does not concern itself with how the data will be stored or where it will be stored physically.

The physical data design model involves translating the logical DB design of the database onto physical media using hardware resources and software systems such as database management systems (DBMS).

DATABASE CONNECTIVITY

To create a connection between the MySQL database and Python, the connect() method of mysql.connector module is used. We pass the database details like HostName, username, and the password in the method call, and then the method returns the connection object.

DATA INSERTION

The button invokes and executes the insert query by getting the values from the text field or from the entry box and inserts the value with the database.

```
-
def addData():
    Payday.set(time.strftime("%d/%m/%Y"))
    refpay=random.randint(15000,99000)
    refpaid=("RP"+str(refpay))
    Reference.set(refpaid)
    if EmployeeName.get()==" " or Address.get()==" " or Reference.get()==" ":
        tkinter.messagebox.showerror("ENTER CORRECT MEMBER DETAILS","NO VALUES")
    else:
        sqlcon=pymysql.connect(host="localhost",user="root",password="@slAmalef12",database="payroll")
        cur=sqlcon.cursor()
        cur.execute("insert into ni_pay values(%s,%s)",(NINumber.get(),NIPayments.get()))
        cur.execute("insert into tax values(%s,%s,%s,%s)",(TaxCode.get(),TaxPeriod.get(),Tax.get(),NINumber.get()))
        cur.execute("insert into payroll_management values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",
            (Reference.get(),
             EmployeeName.get(),
             Address.get(),
             CityWeighting.get(),
             BasicSalary.get(),
             OverTime.get(),
             GrossPay.get(),
             Pension.get(),
             PostCode.get(),
             Gender.get(),
             Payday.get(),
             Netpay.get(),
             TaxCode.get(),
             NINumber.get()))

        sqlcon.commit()
        displaydata()
        sqlcon.close()
        tkinter.messagebox.showinfo("Data Entered Successfully!!!","SUCCESSFULLY INSERTED")
        Reset()
```

Fig.3.1.1.1 Code for data insertoin

Fig.3.1.1.1this figure shows , the code that reads the data from the user and adds the data to the database

PAYROLL MANAGEMENT SYSTEM																
Ref	Fullname	Address	Cityweightit	Basicsalary	Overtime	Grosspay	Pension	Postcode	Gender	Payday	Netpay	Ninumber	Nipayment	Taxcode	Taxperiod	Tax
RP57103	Kathir	156,Kanmar	76	10000	2500	\$12576.00	\$251.52	625011	Male	09/12/2022	\$8262.43	NI361153	\$138.34	TCODE1293	12	\$3772.80
RP23391	Nakshatara	19/6 Anbun	34	11000	2500	\$13534.00	\$270.68	626101	Female	09/12/2022	\$8891.84	NI153332	\$148.87	TCODE1597	12	\$4060.20
RP54864	Sheak	1d3,jothi na	35	12000	500	\$12535.00	\$250.70	628501	Male	09/12/2022	\$8235.49	NI491402	\$137.88	TCODE1687	12	\$3760.50
RP26518	Jayalakshmi	143,Jeevana	20	12000	0	\$12020.00	\$240.40	625011	Female	09/12/2022	\$7897.14	NI73394	\$132.22	TCODE2176	12	\$3606.00
RP41279	Aazam	156,Lakshm	90	11000	1500	\$12590.00	\$251.80	65467	Male	09/12/2022	\$8271.63	NI543903	\$138.49	TCODE3075	12	\$3777.00
RP23467	Raja	145,Rameez	67	9000	3500	\$12567.00	\$251.34	62350	Male	09/12/2022	\$8256.52	NI297202	\$138.24	TCODE6488	12	\$3770.10

Fig. 2.1.2.3 Tree view

Fig. 2.1.2.3 tree view image, shows that the data which is read from the user have been inserted into the database.

```

MySQL 8.0 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.31 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use payroll;
Database changed
mysql> select * from payroll_management;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| reference | Name | Address | Cityweighting | Basic_salary | Over_Time | Gross_pay | Pension | Post_Code | Gender | Pay_Day | Nett_pay | Tax_code | NI_Number |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| RP23391 | Nakshatara | 19/6 Anbunagar,Arupukottai | 34 | 11000 | 2500 | $13534.00 | $270.68 | 626101 | Female | 09/12/2022 | $8891.84 | TCODE15973 | NI153332 |
| RP26518 | Jayalakshmi | 143,Jeevanagar 1st street | 20 | 12000 | 0 | $12020.00 | $240.40 | 625011 | Female | 09/12/2022 | $7897.14 | TCODE21765 | NI73394 |
| RP41279 | Aazam | 156,Lakshmi Nagar 4th Street | 90 | 11000 | 1500 | $12590.00 | $251.80 | 65467 | Male | 09/12/2022 | $8271.63 | TCODE30758 | NI543903 |
| RP26807 | Aafrin Banu | 143,Jeeva nagar | 50 | 9000 | 3500 | $12550.00 | $251.00 | 628501 | Female | 09/12/2022 | $8245.35 | TCODE27590 | NI173204 |
| RP54864 | Sheak | 1d3,jothi nagar,virdhunagar | 35 | 12000 | 500 | $12535.00 | $250.70 | 628501 | Male | 09/12/2022 | $8235.49 | TCODE16873 | NI491402 |
| RP57103 | Kathir | 156,Kanmani nagar 1st street | 76 | 10000 | 2500 | $12576.00 | $251.52 | 625011 | Male | 09/12/2022 | $8262.43 | TCODE12937 | NI361153 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.03 sec)

mysql> select * from rax;
ERROR 1146 (42S02): Table 'payroll.rax' doesn't exist
mysql> select * from tax;
+-----+-----+-----+-----+
| Tax_code | Tax_Period | Tax | NI_Number |
+-----+-----+-----+-----+
| TCODE12937 | 12 | $3772.80 | NI361153 |
| TCODE13643 | 12 | $2904.00 | NI173807 |
| TCODE15973 | 12 | $4060.20 | NI153332 |
| TCODE16873 | 12 | $3760.50 | NI491402 |
| TCODE21514 | 12 | $1485.00 | NI268738 |
| TCODE21765 | 12 | $3606.00 | NI73394 |
| TCODE27599 | 12 | $3765.00 | NI173204 |
| TCODE30758 | 12 | $3777.00 | NI543903 |
+-----+-----+-----+-----+
8 rows in set (0.00 sec)

mysql> select * from ni_pay;
+-----+-----+
| NI_Number | NI_Payments |
+-----+-----+
| NI153332 | $148.87 |
| NI173204 | $138.05 |
| NI173807 | $106.48 |
| NI361153 | $138.24 |
| NI368738 | $54.47 |
| NI491402 | $137.88 |
| NI543903 | $138.49 |
| NI73394 | $132.22 |
+-----+-----+
8 rows in set (0.00 sec)

mysql>

```

Fig.3.1.1.1(1) Database

Fig.3.1.1.1(1) Database shows the data is inserted into the database. And then it is displayed to user by means GUI – tree view.

DATA DELETION

The button invokes and executes the delete query by getting the values from the text field or from the entry box and deletes the value from the database.

```
def deleteDB():
    sqlcon=pymysql.connect(host="localhost",user="root",password="@slAmalef12",database="payroll")
    cur=sqlcon.cursor()
    cur.execute("delete from payroll_management where reference=%s",Reference.get())
    cur.execute("delete from tax where Tax_code=%s",TaxCode.get())
    cur.execute("delete from ni_pay where NI_Number=%s",NINumber.get())
    sqlcon.commit()
    sqlcon.close()
    displaydata()
    Reset()
    clear_all()
    tkinter.messagebox.showinfo("Data Deleted Successfully!!!","SUCCESSFULLY DELETED👍")
```

Fig.3.1.1.2 Code for data deletion

Fig.3.1.1.1.2is the python code that does the deleting operation of the data from the database

DATA UPDATION

The button invokes and executes the update query by getting the values from the text field or from the entry box and updates the values in the database.

```
def update():
    sqlcon=pymysql.connect(host="localhost",user="root",password="@slAmalef12",database="payroll")
    cur=sqlcon.cursor()
    cur.execute("update ni_pay set ni_payments=%s where ni_number=%s", (NIPayments.get(), NINumber.get()))
    cur.execute("update tax set Tax_period=%s,Tax=%s where tax_code=%s", (TaxPeriod.get(), Tax.get(), TaxCode.get()))
    cur.execute("update payroll_management set name=%s,address=%s,cityweighting=%s,basic_salary=%s,\
                over_time=%s,gross_pay=%s,pension=%s,post_code=%s,gender=%s,\
                pay_day=%s,nett_pay=%s where reference=%s", (
        EmployeeName.get(),
        Address.get(),
        CityWeighting.get(),
        BasicSalary.get(),
        OverTime.get(),
        GrossPay.get(),
        Pension.get(),
        PostCode.get(),
        Gender.get(),
        Payday.get(),
        Netpay.get(),
        Reference.get()))
    sqlcon.commit()
    sqlcon.close()
    displaydata()
    tkinter.messagebox.showinfo("Data Entered Successfully!!!","SUCCESSFULLY UPDATED👍")
```

Fig.3.1.1.3 Code for data updation

Fig.3.1.1.3 shows the python code that updates the data in the database by getting the data as an input from the user

SCHEMA

Employee (Reference, Employee name, Address, CityWeighting, Basic salary, Overtime, Gross pay, Pension, Gender, Pay day, Net pay, Tax code, NI_Number, Tax_code, Tax_period, NI_Payments)

TABLE CREATION

```
mysql> Create table payroll_management
-> (
-> reference VARCHAR(50) PRIMARY KEY,
-> Name VARCHAR(50),
-> Address VARCHAR(100),
-> Cityweighting VARCHAR(50),
-> Basic_salary VARCHAR(50),
-> Over_Time VARCHAR(50),
-> Gross_pay VARCHAR(50),
-> Pension VARCHAR(50),
-> Post_Code VARCHAR(50),
-> Gender VARCHAR(50),
-> Pay_Day VARCHAR(50),
-> Nett_pay VARCHAR(50),
-> Tax_code VARCHAR(50) REFERENCES Tax(Tax_code),
-> NI_Number VARCHAR(50) REFERENCES NI_pay(NI_Number)
-> );
Query OK, 0 rows affected (1.62 sec)
```

```
mysql> Create Table Tax
-> (
-> Tax_code VARCHAR(50) PRIMARY KEY,
-> Tax_Period VARCHAR(50),
-> Tax VARCHAR(50),
-> NI_Number VARCHAR(50) REFERENCES NI_pay(NI_Number)
-> );
Query OK, 0 rows affected (0.23 sec)
```

```
mysql> Create table NI_pay
-> (
-> NI_Number VARCHAR(50) PRIMARY KEY,
-> NI_Payments VARCHAR(50)
-> );
Query OK, 0 rows affected (0.31 sec)
```

Fig.3.2.1 Table creation

ER DIAGRAM

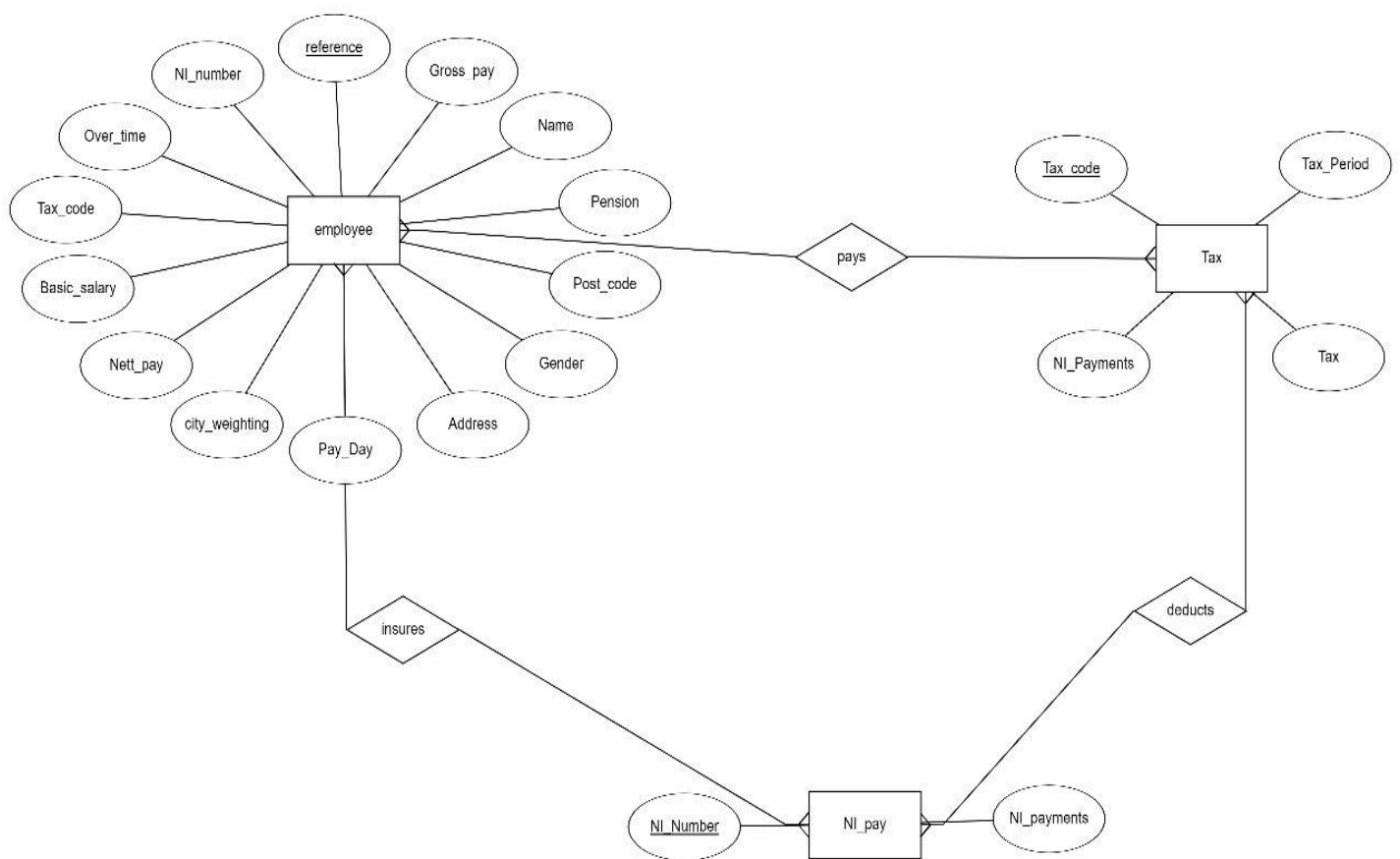


Fig.3.3 ER Diagram

SCHEMA DIAGRAM

- Employee (Reference, Employee name, Address, CityWeighting, Basicsalary, Overtime, Grosspay, Pension, Gender, Payday, Netpay, Taxcode, NINumber)
- ni_pay (ni_number, ni_paymentnts)
- tax(tax_code, tax_period, tax, ni_number

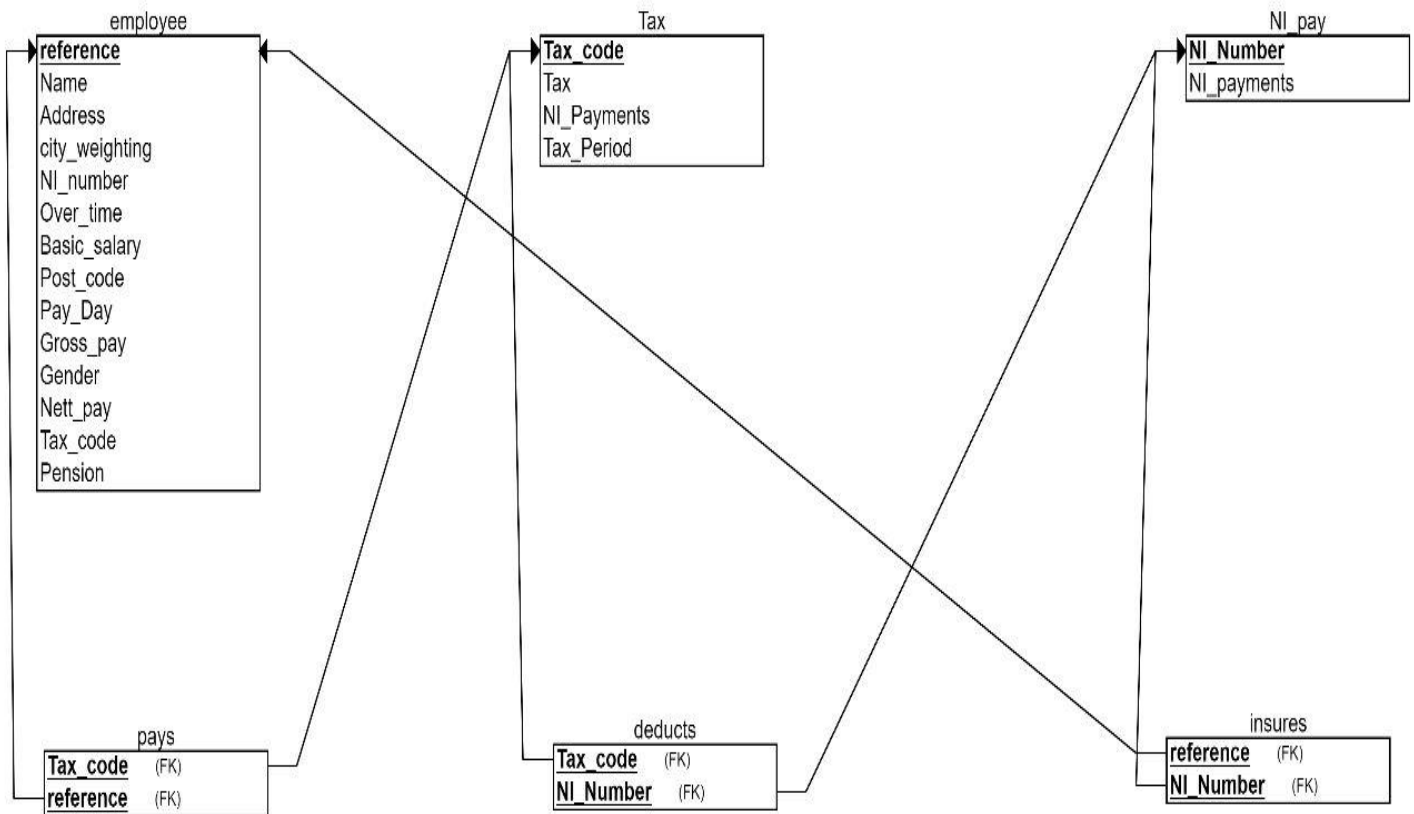


Fig.3.4 Schema Diagram

FLOW DIAGRAM

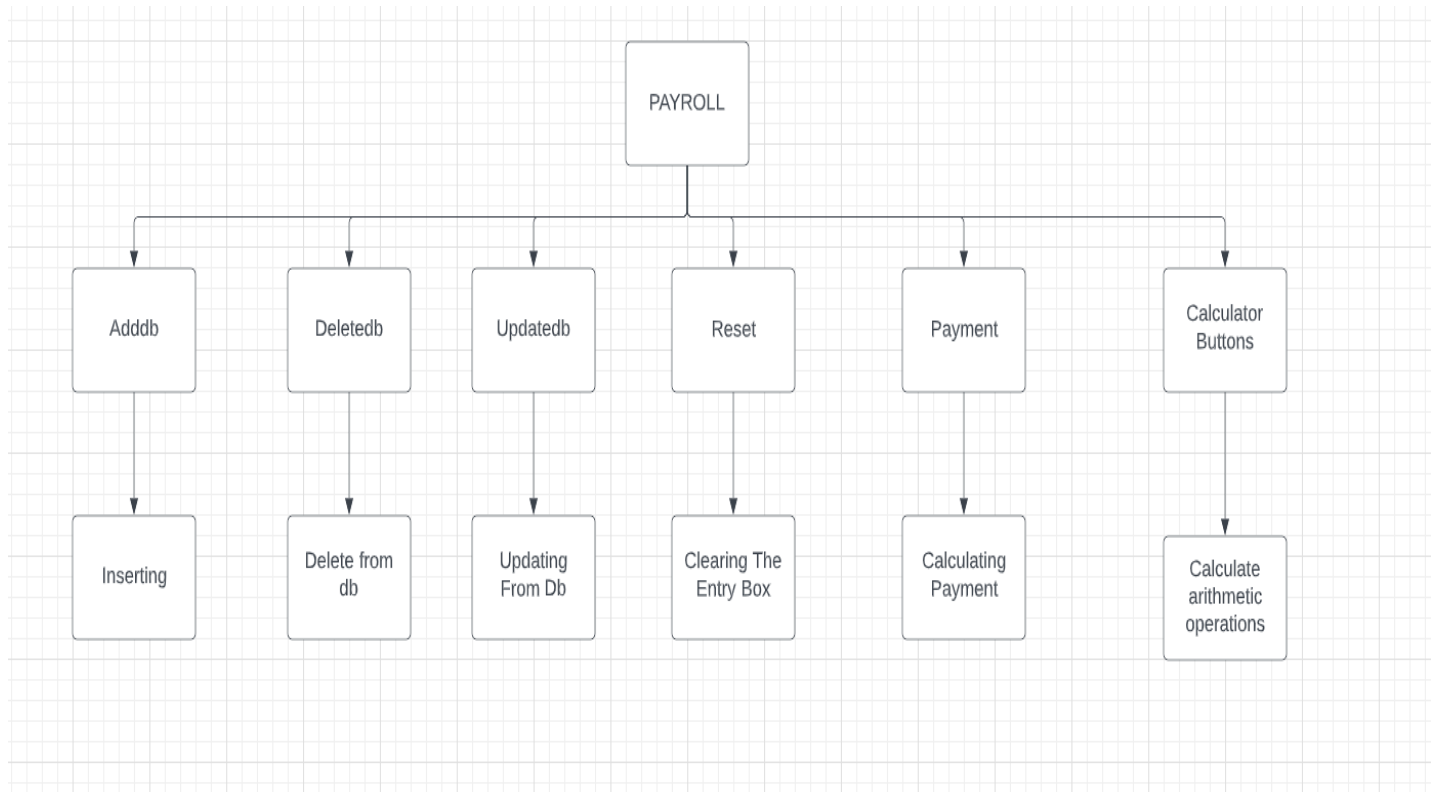


Fig.3.5 Flow Diagram

DEPENDENCIES

- {reference,taxcode}->tax //PARTIAL FD
- {reference,taxcode}->tax_period //PARTIAL FD
- {reference,ni_number}->ni_payments //PARTIAL FD
- reference->name
- reference->cityweighting
- reference->basic_salary
- reference->overtime
- reference->Gross_pay
- reference->Pension
- reference->postcode
- reference->pay_day
- reference->net_pay
- reference->tax_code
- reference->ni_number
- Tax_code->tax // TRANSITIVE FD
- Tax_code->tax_period //TRANSITIVE FD
- Ni_number->ni_payments // TRANSITIVE FD

NORMALIZATION

1 NF :

There is no Multivalued attribute. All attributes are Atomic.
It is in 1NF.

Employee (Reference, Employee name, Address, CityWeighting, Basicsalary, Overtime, Grosspay, Pension, Gender, Payday, Netpay, Taxcode, NI_Number, Tax_code, Tax_period, Ni_Payments) // NO DECOMPOSITION

2NF :

If there is partial functional dependency then it is not in 2NF.
If there is a PFD remove the PFD and convert the table into 2NF.
 {reference, taxcode} -> tax // PARTIAL FD
 {reference, taxcode} -> tax_period // PARTIAL FD
 {reference, ni_number} -> ni_payments // PARTIAL FD
It is removed So it is in 2NF.

3 NF :

The table should be free from Transitive Dependency then it is in 4NF.

Reference -> Taxcode

Taxcode -> Taxperiod

Taxcode -> Tax

Reference -> Ni_code

Ni_code -> Ni_key

Then,

- Employee (Reference, Employee name, Address, CityWeighting, Basicsalary, Overtime, Grosspay, Pension, Gender, Payday, Netpay, Taxcode, NI Number)
- ni_pay (ni_number, ni_paymentnts)
- tax(tax_code, tax_period, tax, ni_number)

SOURCE CODE

```
from tkinter import*
from tkinter import ttk
import random
import tkinter.messagebox
import datetime
import time
import tkinter.ttk as tktrtk
import tkinter as tt
import pymysql
class Payroll:

    def __init__(self,root):
        self.root=root
        #TITTLE FOR OUR TAB
        self.root.title("Payroll Management System")
        self.root.geometry("1350x800+0+0")
        #USING NOTE_BOOK WIDGET
        notebook=ttk.Notebook(self.root)
        self.control1=ttk.Frame(notebook)
        self.control2=ttk.Frame(notebook)
        self.control3=ttk.Frame(notebook)
        notebook.add(self.control1,text='PAYROLL SYSTEM')
        notebook.add(self.control2,text='VIEW PAYMENTS')
        notebook.add(self.control3,text='COMMENT BOOK')
        notebook.grid()

        EmployeeName=StringVar()
        Address=StringVar()
        Reference=StringVar()
        EmployerName=StringVar()
        CityWeighting=IntVar()
        BasicSalary=IntVar()
        OverTime=StringVar()
        OtherPaymentDue=StringVar()
        GrossPay=StringVar()
        Tax=StringVar()
        Pension=StringVar()
        StudentLoan=StringVar()
        NIPayments=StringVar()
        Deductions=StringVar()
        PostCode=StringVar()
        Gender=StringVar()
        Payday=StringVar()
        TaxPeriod=StringVar()
        TaxCode=StringVar()
        NINumber=StringVar()
        TaxablePay=StringVar()
        PensionablePay=StringVar()
        Netpay=StringVar()

        text_in=StringVar()
        global operator
        operator=""
        CityWeighting.set("")
        BasicSalary.set("")
```

```

#=====calculator=====
def btnClick(numbers):
    global operator
    operator=operator+str(numbers)
    text_in.set(operator)

def btnClear():
    global operator
    operator=""
    text_in.set("")

def btnEquals():
    global operator
    s=str(eval(operator))
    text_in.set(s)
    operator=""

#=====EXIT=====
def Exit():
    Exit=tkinter.messagebox.askyesno("Payroll System","Confirm if you want to exit")
    if Exit>0:
        root.destroy()
        return
def Reset():
    EmployeeName.set("")
    Address.set("")
    Reference.set("")
    EmployerName.set("")
    CityWeighting.set("")
    BasicSalary.set("")
    OverTime.set("")
    OtherPaymentDue.set("")
    GrossPay.set("")
    Tax.set("")
    Pension.set("")
    StudentLoan.set("")
    NIPayments.set("")
    Deductions.set("")
    PostCode.set("")
    Gender.set("")
    Payday.set("")
    TaxPeriod.set("")
    TaxCode.set("")
    NINumber.set("")
    TaxablePay.set("")
    PensionablePay.set("")
    Netpay.set("")

#=====WAGES=====
def Payref():
    Nipay=random.randint(32000,567890)
    nipaid=("NI"+str(Nipay))
    NINumber.set(nipaid)
    idate=datetime.datetime.now()
    TaxPeriod.set(idate.month)
    axcode=random.randint(1435,35768)
    acode=("TCODE"+str(axcode))
    TaxCode.set(acode)

```

```

def payment():
    Payref()
    bs=float(BasicSalary.get())
    cw=float(CityWeighting.get())
    ot=float(OverTime.get())
    mtax=((bs+cw+ot)*0.3)
    ttax=str('$.2f'%(mtax))
    Tax.set(ttax)

    mpen=((bs+cw+ot)*0.02)
    mpension=str('$.2f'%(mpen))
    Pension.set(mpension)

    mstudentloan=((bs+cw+ot)*0.012)
    mstudent=str('$.2f'%(mstudentloan))
    StudentLoan.set(mstudent)

    mni=((bs+cw+ot)*0.011)
    mnipay=str('$.2f'%(mni))
    NIPayments.set(mnipay)

    deduct=(mtax+mpen+mstudentloan+mni)
    dday=str('$.2f'%(deduct))
    Deductions.set(dday)

    gpay=str('$.2f'%(bs+cw+ot))
    GrossPay.set(gpay)

    npay=(bs+cw+ot)-deduct
    nafter=str('$.2f'%(npay))
    Netpay.set(nafter)

    TaxablePay.set(ttax)
    PensionablePay.set(mpension)
    OtherPaymentDue.set("0.00")

#=====data=====
def addData():
    Payday.set(time.strftime("%d/%m/%Y"))
    refpay=random.randint(15000,99000)
    repaid=("RP"+str(refpay))
    Reference.set(repaid)
    if EmployeeName.get()==" or Address.get()==" or Reference.get()=="":
        tkinter.messagebox.showerror("ENTER CORRECT MEMBER DETAILS")
    else:
        sqlcon=pymysql.connect(host="localhost",user="root",password="@slAmalef12",database="payroll
")
        cur=sqlcon.cursor()
        cur.execute("insert into ni_pay values(%s,%s)",(NINumber.get(),NIPayments.get()))
        cur.execute("insert into tax
values(%s,%s,%s,%s)",(TaxCode.get(),TaxPeriod.get(),Tax.get(),NINumber.get()))
        cur.execute("insert into payroll_management
values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",
        (Reference.get(),
        EmployeeName.get(),

```



```

        Address.get(),
        CityWeighting.get(),
        BasicSalary.get(),
        OverTime.get(),
        GrossPay.get(),
        Pension.get(),
        PostCode.get(),
        Gender.get(),
        Payday.get(),
        Netpay.get(),
        TaxCode.get(),
        NINumber.get()))

```

```

sqlcon.commit()
displaydata()
sqlcon.close()
tkinter.messagebox.showinfo("Data Entered Successfully!!!", "SUCCESSFULLY INSERTED")

```

```

#=====data=====

```

```

def displaydata():

```

```

    sqlcon=pymysql.connect(host="localhost",user="root",password="@slAmalef12",database="payroll")
    cur=sqlcon.cursor()
    cur.execute("select
p.reference,p.name,p.address,p.cityweighting,p.basic_salary,p.over_time,p.gross_pay,p.pension,p.post_code
,p.gender,p.pay_day,p.nett_pay,\
    n.ni_number,n.ni_payments,t.tax_code,t.tax_period,t.tax \
    from payroll_management p \
    inner join ni_pay n on p.ni_number=n.ni_number\
    inner join tax t on n.ni_number=t.ni_number")
    result=cur.fetchall()
    if len(result)!=0:
        self.payroll_rec.delete(*self.payroll_rec.get_children())
        for row in result:
            self.payroll_rec.insert("",END,values=row)
        sqlcon.commit()
def clear_all():
    for item in self.payroll_rec.get_children():
        self.payroll_rec.delete(item)
    break

```

```

def Wagesinfo(ev):
    view=self.payroll_rec.focus()
    learn=self.payroll_rec.item(view)
    row=learn['values']
    Reference.set(row[0])
    EmployerName.set(row[1])
    Address.set(row[2])
    CityWeighting.set(row[3])
    BasicSalary.set(row[4])
    OverTime.set(row[5])
    GrossPay.set(row[6])
    Pension.set(row[7])
    PostCode.set(row[8])
    Gender.set(row[9])

```

```

Payday.set(row[10])
Netpay.set(row[11])
NINumber.set(row[12])
NIPayments.set(row[13])
TaxCode.set(row[14])
TaxPeriod.set(row[15])
Tax.set(row[16])
def update():
    sqlcon=pymysql.connect(host="localhost",user="root",password="@slAmalef12",database="payroll")
    cur=sqlcon.cursor()
    cur.execute("update ni_pay set ni_payments=%s where
ni_number=%s",(NIPayments.get(),NINumber.get()))
    cur.execute("update tax set Tax_period=%s,Tax=%s where
tax_code=%s",(TaxPeriod.get(),Tax.get(),TaxCode.get()))
    cur.execute("update payroll_management set
name=%s,address=%s,cityweighting=%s,basic_salary=%s,\
over_time=%s,gross_pay=%s,pension=%s,post_code=%s,gender=%s,\
pay_day=%s,nett_pay=%s where reference=%s",(\
EmployeeName.get(),
Address.get(),
CityWeighting.get(),
BasicSalary.get(),
OverTime.get(),
GrossPay.get(),
Pension.get(),
PostCode.get(),
Gender.get(),
Payday.get(),
Netpay.get(),
Reference.get()))

    sqlcon.commit()
    sqlcon.close()
    displaydata()
    tkinter.messagebox.showinfo("Data Entered Successfully!!!","SUCCESSFULLY
UPDATED✦✧")

def deleteDB():
    sqlcon=pymysql.connect(host="localhost",user="root",password="@slAmalef12",database="payroll")
    cur=sqlcon.cursor()
    cur.execute("delete from payroll_management where reference=%s",Reference.get())
    cur.execute("delete from tax where Tax_code=%s",TaxCode.get())
    cur.execute("delete from ni_pay where NI_Number=%s",NINumber.get())
    sqlcon.commit()
    sqlcon.close()
    displaydata()
    Reset()
    clear_all()
    tkinter.messagebox.showinfo("Data Deleted Successfully!!!","SUCCESSFULLY DELETED✦✧")

```

#-----FRAMES-----

```
Tab1 = Frame(self.control1, bd=10, width=1350, height=700,relief=RIDGE)
Tab1.grid()
Tab2 = Frame(self.control2, bd=10, width=1350, height=700,relief=RIDGE)
Tab2.grid()
Tab3 = Frame(self.control3, bd=10, width=1350, height=700,relief=RIDGE)
Tab3.grid()
```

```
topframe1=Frame(Tab1, bd=10, width=1340, height=100, relief=RIDGE)
topframe1.grid()
topframe2=Frame(Tab1, bd=10, width=1340, height=100, relief=RIDGE)
topframe2.grid()
topframe3=Frame(Tab1, bd=10, width=1340, height=500, relief=RIDGE)
topframe3.grid()
```

```
leftframe=Frame(topframe3, bd=5, width=1340, height=400, padx=2,bg="VioletRed2",relief=RIDGE)
leftframe.pack(side=RIGHT)
leftframe1=Frame(leftframe, bd=5, width=600, height=180, padx=2,relief=RIDGE)
leftframe1.pack(side=TOP)
```

```
leftframe2=Frame(leftframe, bd=5, width=600, height=180, padx=2, bg="VioletRed2" ,relief=RIDGE)
leftframe2.pack(side=TOP)
leftframe2left=Frame(leftframe2, bd=5, width=300, height=170, padx=2 ,relief=RIDGE)
leftframe2left.pack(side=LEFT)
leftframe2right=Frame(leftframe2, bd=5, width=300, height=170, padx=2 ,relief=RIDGE)
leftframe2right.pack(side=RIGHT)
```

```
leftframe3left=Frame(leftframe, bd=5, width=320, height=50, padx=2, bg="VioletRed2", relief=RIDGE)
leftframe3left.pack(side=LEFT)
leftframe3right=Frame(leftframe, bd=5, width=320, height=50, padx=2, bg="VioletRed2", relief=RIDGE)
leftframe3right.pack(side=RIGHT)
```

```
rightframe1=Frame(topframe3, bd=5, width=320, height=400, padx=2, bg="VioletRed2", relief=RIDGE)
rightframe1.pack(side=RIGHT)
rightframe1a=Frame(rightframe1, bd=5, width=310, height=300, padx=2, relief=RIDGE)
rightframe1a.pack(side=TOP)
rightframe1b=Frame(rightframe1, bd=5, width=310, height=100, padx=2, relief=RIDGE)
rightframe1b.pack(side=TOP)
```

```
rightframe2=Frame(topframe3, bd=5, width=300, height=400, padx=2, bg="VioletRed2", relief=RIDGE)
rightframe2.pack(side=LEFT)
rightframe2a=Frame(rightframe2, bd=5, width=280, height=50, padx=2, relief=RIDGE)
rightframe2a.pack(side=TOP)
rightframe2b=Frame(rightframe2, bd=5, width=280, height=180, padx=2, relief=RIDGE)
rightframe2b.pack(side=TOP)
rightframe2c=Frame(rightframe2, bd=5, width=280, height=180, padx=2, relief=RIDGE)
rightframe2c.pack(side=TOP)
rightframe2d=Frame(rightframe2, bd=5, width=280, height=50, padx=2, bg="VioletRed2", relief=RIDGE)
rightframe2d.pack(side=TOP)
```

#-----TITTLE MAIN TITTLE-----

```
self.toptittle=Label(topframe1, font=('stylus',40,'bold'),text="\tPAYROLL MANAGEMENT
SYSTEM\t",justify=CENTER)
self.toptittle.grid(padx=76)
```

```

#-----EMPLOYEE-----
self.employeename=Label(topframe2, font=('stylus', 12, 'bold'), text="EMPLOYEE NAME", bd=10)
self.employeename.grid(row=0, column=0, sticky=W)
self.txtemployeename=Entry(topframe2, font=('stylus', 12, 'bold'), bd=5, width=59, justify='left', textvariable=
EmployeeName)
self.txtemployeename.grid(row=0, column=1)
#-----ADDRESS-----
self.lbladdress=Label(topframe2, font=('stylus', 12, 'bold'), text="ADDRESS", bd=10)
self.lbladdress.grid(row=1, column=0, sticky=W)
self.txtaddress=Entry(topframe2,
font=('stylus', 12, 'bold'), bd=5, width=60, justify='left', textvariable=Address)
self.txtaddress.grid(row=1, column=1)
#-----POST CODE-----
self.lblpostcode=Label(topframe2, font=('stylus', 12, 'bold'), text="POST CODE", bd=10)
self.lblpostcode.grid(row=0, column=2, sticky=W)
self.txtpostcode=Entry(topframe2, font=('stylus', 12, 'bold'), bd=5, width=50, justify='left', textvariable=
PostCode)
self.txtpostcode.grid(row=0, column=3)
#-----GENDER-----
self.lblgen=Label(topframe2, font=('stylus', 12, 'bold'), text="GENDER", bd=10)
self.lblgen.grid(row=1, column=2, sticky=W)
self.gen=ttk.Combobox(topframe2, textvariable=
Gender, state='readonly', font=('stylus', 12, 'bold'), width=46)
self.gen['value']=('NOT MENTION', 'Female', 'Male')
self.gen.current(0)
self.gen.grid(row=1, column=3)
#-----PAY DAY-----
self.lbpayday=Label(rightframe2a, font=('stylus', 12, 'bold'), text="PAY DAY", bd=10)
self.lbpayday.grid(row=0, column=0, sticky=W)
self.txtpayday=Entry(rightframe2a,
font=('stylus', 12, 'bold'), bd=5, width=20, justify='left', textvariable=Payday, state=DISABLED)
self.txtpayday.grid(row=0, column=1)
#-----TAX PERIOD-----

self.lbtaxperiod=Label(rightframe2b, font=('stylus', 12, 'bold'), text="TAX PERIOD", bd=10)
self.lbtaxperiod.grid(row=0, column=0, sticky=W)
self.txttaxperiod=Entry(rightframe2b,
font=('stylus', 12, 'bold'), bd=5, width=17, justify='left', textvariable=TaxPeriod, state=DISABLED)
self.txttaxperiod.grid(row=0, column=1)

#-----TAX CODE-----
self.lbtaxcode=Label(rightframe2b, font=('stylus', 12, 'bold'), text="TAX CODE", bd=10)
self.lbtaxcode.grid(row=1, column=0, sticky=W)
self.txttaxcode=Entry(rightframe2b,
font=('stylus', 12, 'bold'), bd=5, width=17, justify='left', textvariable=TaxCode, state=DISABLED)
self.txttaxcode.grid(row=1, column=1)
#-----NATIONAL INSURANCE NUMBER-----
self.lbninum=Label(rightframe2b, font=('stylus', 12, 'bold'), text="NI NUMBER", bd=10)
self.lbninum.grid(row=2, column=0, sticky=W)
self.txtninum=Entry(rightframe2b,
font=('stylus', 12, 'bold'), bd=5, width=17, justify='left', textvariable=NI Number, state=DISABLED)
self.txtninum.grid(row=2, column=1)
#-----TAXABLE PAY-----
self.lbtaxpay=Label(rightframe2c, font=('stylus', 12, 'bold'), text="TAXABLE PAY", bd=10)
self.lbtaxpay.grid(row=0, column=0, sticky=W)
self.txttaxpay=Entry(rightframe2c,

```

```
font=('stylus', 12, 'bold'), bd=5, width=11, justify='left', textvariable=TaxablePay, state=DISABLED)
self.txttaxpay.grid(row=0, column=1)
```

```
#-----PENSIONABLE PAY-----
```

```
self.lbppay=Label(rightframe2c, font=('stylus', 12, 'bold'), text="PENSIONABLE PAY", bd=10)
self.lbppay.grid(row=1, column=0, sticky=W)
self.txtppay=Entry(rightframe2c,
font=('stylus', 12, 'bold'), bd=5, width=11, justify='left', textvariable=PensionablePay, state=DISABLED)
self.txtppay.grid(row=1, column=1)
```

```
#-----NETT PAY-----
```

```
self.lbnetpay=Label(rightframe2d, font=('stylus', 12, 'bold'), text="NETT PAY", bd=10)
self.lbnetpay.grid(row=0, column=0, sticky=W)
self.txtnetpay=Entry(rightframe2d,
font=('stylus', 12, 'bold'), bd=5, width=19, justify='left', textvariable=Netpay, state=DISABLED)
self.txtnetpay.grid(row=0, column=1)
```

```
#-----REFERENCE-----
```

```
self.lbref=Label(leftframe1, font=('stylus', 12, 'bold'), text="REFERENCE", bd=10)
self.lbref.grid(row=0, column=0, sticky=W)
self.txtref=Entry(leftframe1,
font=('stylus', 12, 'bold'), bd=5, width=57, justify='left', textvariable=Reference, state=DISABLED)
self.txtref.grid(row=0, column=1)
```

```
#-----EMPLOYER NAME-----
```

```
self.lbempname=Label(leftframe1, font=('stylus', 12, 'bold'), text="EMPLOYER NAME", bd=10)
self.lbempname.grid(row=1, column=0, sticky=W)
self.txtempname=Entry(leftframe1,
font=('stylus', 12, 'bold'), bd=5, width=57, justify='left', textvariable=EmployerName)
self.txtempname.grid(row=1, column=1)
```

```
#-----EMPLOYEE NAME-----
```

```
self.lbempname=Label(leftframe1, font=('stylus', 12, 'bold'), text="EMPLOYEE NAME", bd=10)
self.lbempname.grid(row=2, column=0, sticky=W)
self.txtempname=Entry(leftframe1,
font=('stylus', 12, 'bold'), bd=5, width=57, justify='left', textvariable=EmployeeName, state=DISABLED)
self.txtempname.grid(row=2, column=1)
```

```
#-----CITY WEIGHTING-----
```

```
self.lbctywe=Label(leftframe2left, font=('stylus', 12, 'bold'), text="CITY WEIGHTING", bd=10, anchor='e')
self.lbctywe.grid(row=0, column=0, sticky=W)
self.txtctywe=Entry(leftframe2left,
font=('stylus', 12, 'bold'), bd=5, width=20, justify='left', textvariable=CityWeighting)
self.txtctywe.grid(row=0, column=1)
```

```
#-----BASIC SALARY-----
```

```
self.lbbsal=Label(leftframe2left, font=('stylus', 12, 'bold'), text="BASIC SALARY", bd=10)
self.lbbsal.grid(row=1, column=0, sticky=W)
self.txtbsal=Entry(leftframe2left,
font=('stylus', 12, 'bold'), bd=5, width=20, justify='left', textvariable=BasicSalary)
self.txtbsal.grid(row=1, column=1)
```

```
#-----OVER TIME-----
```

```
self.lbotpay=Label(leftframe2left, font=('stylus', 12, 'bold'), text="OVER TIME", bd=10)
self.lbotpay.grid(row=2, column=0, sticky=W)
self.txtotpay=Entry(leftframe2left,
font=('stylus', 12, 'bold'), bd=5, width=20, justify='left', textvariable=OverTime)
self.txtotpay.grid(row=2, column=1)
```

```
#-----OTHER PAY-----
```

```
self.lbopay=Label(leftframe2left, font=('stylus', 12, 'bold'), text="OTHER PAY", bd=10)
self.lbopay.grid(row=3, column=0, sticky=W)
```

```

self.txttopay=Entry(leftframe2left,
font=('stylus', 12, 'bold'), bd=5, width=20, justify='left', textvariable=OtherPaymentDue)
self.txttopay.grid(row=3, column=1)

```

```

#-----TAX-----

```

```

self.lbtax=Label(leftframe2right, font=('stylus', 12, 'bold'), text="TAX", bd=10, anchor='e')
self.lbtax.grid(row=0, column=0, sticky=W)
self.txttax=Entry(leftframe2right,
font=('stylus', 12, 'bold'), bd=5, width=20, justify='left', textvariable=Tax, state=DISABLED)
self.txttax.grid(row=0, column=1)

```

```

#-----PENSION-----

```

```

self.lbpension=Label(leftframe2right, font=('stylus', 12, 'bold'), text="PENSION", bd=10)
self.lbpension.grid(row=1, column=0, sticky=W)
self.txtpension=Entry(leftframe2right,
font=('stylus', 12, 'bold'), bd=5, width=20, justify='left', textvariable=Pension, state=DISABLED)
self.txtpension.grid(row=1, column=1)

```

```

#-----STUDENT LOAN-----

```

```

self.lbstloan=Label(leftframe2right, font=('stylus', 12, 'bold'), text="STUDENT LOAN", bd=10)
self.lbstloan.grid(row=2, column=0, sticky=W)
self.txtstloan=Entry(leftframe2right,
font=('stylus', 12, 'bold'), bd=5, width=20, justify='left', textvariable=StudentLoan, state=DISABLED)
self.txtstloan.grid(row=2, column=1)

```

```

#-----NI PAYMENT-----

```

```

self.lbnipay=Label(leftframe2right, font=('stylus', 12, 'bold'), text="NI PAYMENT", bd=10)
self.lbnipay.grid(row=3, column=0, sticky=W)
self.txtnipay=Entry(leftframe2right,
font=('stylus', 12, 'bold'), bd=5, width=20, justify='left', textvariable=NIPayments, state=DISABLED)
self.txtnipay.grid(row=3, column=1)

```

```

#-----GROSS PAY-----

```

```

self.lbgpay=Label(leftframe3left, font=('stylus', 12, 'bold'), text="GROSS PAY", bd=10)
self.lbgpay.grid(row=3, column=0, sticky=W)
self.txtgpay=Entry(leftframe3left,
font=('stylus', 12, 'bold'), bd=5, width=25, justify='left', textvariable=GrossPay, state=DISABLED)
self.txtgpay.grid(row=3, column=1)

```

```

#-----DEDUCTIONS-----

```

```

self.lbdpay=Label(leftframe3right, font=('stylus', 12, 'bold'), text="DEDUCTIONS", bd=10)
self.lbdpay.grid(row=3, column=0, sticky=W)
self.txtdpay=Entry(leftframe3right,
font=('stylus', 12, 'bold'), bd=5, width=23, justify='left', textvariable=Deductions, state=DISABLED)
self.txtdpay.grid(row=3, column=1)

```

```

#-----CALCULATOR-----

```

```

self.txtdisplay=Entry(rightframe1a,
font=('stylus', 19, 'bold'), bd=5, insertwidth=4, justify='right', textvariable=text_in)
self.txtdisplay.grid(row=0, column=0, columnspan=4)

```

```

#-----

```

```

self.btnwages=Button(rightframe1b, padx=16, pady=0, bd=5, font=('stylus', 16, 'bold'), width=4, text="WAGES",
command=lambda:payment()).grid(row=0, column=0)

```

```

self.btndisplay=Button(rightframe1b,padx=16,pady=0,bd=5,font=('stylus',16,'bold'),width=4,text="ADD",c
ommand=lambda:addData()).grid(row=0,column=1)
self.btnupdate=Button(rightframe1b,padx=16,pady=0,bd=5,font=('stylus',16,'bold'),width=4,text="UPDAT
E",command=lambda:update()).grid(row=0,column=2)
self.btndel=Button(rightframe1b,padx=16,pady=0,bd=5,font=('stylus',16,'bold'),width=4,text="DELETE",c
ommand=lambda:deleteDB()).grid(row=1,column=0)
self.btnreset=Button(rightframe1b,padx=16,pady=0,bd=5,font=('stylus',16,'bold'),width=4,text="RESET",
command=lambda:Reset()).grid(row=1,column=1)
self.btnexit=Button(rightframe1b,padx=16,pady=0,bd=5,font=('stylus',16,'bold'),width=4,text="EXIT",com
mand=lambda:Exit()).grid(row=1,column=2)

```

```

self.btn1=Button(rightframe1a,padx=6,pady=6,bd=2,font=('stylus',16,'bold'),width=4,text="1",command=l
ambda:btnClick(1)).grid(row=3,column=0)
self.btn2=Button(rightframe1a,padx=6,pady=6,bd=2,font=('stylus',16,'bold'),width=4,text="2",command=l
ambda:btnClick(2)).grid(row=3,column=1)
self.btn3=Button(rightframe1a,padx=6,pady=6,bd=2,font=('stylus',16,'bold'),width=4,text="3",command=l
ambda:btnClick(3)).grid(row=3,column=2)
self.btnmul=Button(rightframe1a,padx=6,pady=6,bd=2,font=('stylus',16,'bold'),width=4,text="x",command
=lambda:btnClick("*")).grid(row=3,column=3)
#-----

```

```

self.btn4=Button(rightframe1a,padx=6,pady=6,bd=2,font=('stylus',16,'bold'),width=4,text="4",command=l
ambda:btnClick(4)).grid(row=2,column=0)
self.btn5=Button(rightframe1a,padx=6,pady=6,bd=2,font=('stylus',16,'bold'),width=4,text="5",command=l
ambda:btnClick(5)).grid(row=2,column=1)
self.btn6=Button(rightframe1a,padx=6,pady=6,bd=2,font=('stylus',16,'bold'),width=4,text="6",command=l
ambda:btnClick(6)).grid(row=2,column=2)
self.btndiv=Button(rightframe1a,padx=6,pady=6,bd=2,font=('stylus',16,'bold'),width=4,text="/"
,command=lambda:btnClick("/")).grid(row=2,column=3)
#-----

```

```

self.btn7=Button(rightframe1a,padx=6,pady=6,bd=2,font=('stylus',16,'bold'),width=4,text="7",command=l
ambda:btnClick(7)).grid(row=1,column=0)
self.btn8=Button(rightframe1a,padx=6,pady=6,bd=2,font=('stylus',16,'bold'),width=4,text="8",command=l
ambda:btnClick(8)).grid(row=1,column=1)
self.btn9=Button(rightframe1a,padx=6,pady=6,bd=2,font=('stylus',16,'bold'),width=4,text="9",command=l
ambda:btnClick(9)).grid(row=1,column=2)
self.btnAdd=Button(rightframe1a,padx=6,pady=6,bd=2,font=('stylus',16,'bold'),width=4,text="+",comman
d=lambda:btnClick("+")).grid(row=1,column=3)

```

#-----BUTTON-----

```

self.btn0=Button(rightframe1a,padx=6,pady=6,bd=2,font=('stylus',16,'bold'),width=4,text="0",command=l
ambda:btnClick(0)).grid(row=4,column=0)
self.btnclear=Button(rightframe1a,padx=6,pady=6,bd=2,font=('stylus',16,'bold'),width=4,text="C",comma
nd=lambda:btnClear()).grid(row=4,column=1)
self.btnEqual=Button(rightframe1a,padx=6,pady=6,bd=2,font=('stylus',16,'bold'),width=4,text="=",comma
nd=lambda:btnEquals()).grid(row=4,column=2)
self.btndiv=Button(rightframe1a,padx=6,pady=6,bd=2,font=('stylus',16,'bold'),width=4,text="/",command=
lambda:btnClick("/")).grid(row=4,column=3)

```

#-----TAB-2-----

```

topframe11=Frame(Tab2,bd=10,width=1340,height=100,relief=RIDGE)
topframe11.grid(row=0,column=0)
topframe12=Frame(Tab2,bd=10,width=1340,height=100,relief=RIDGE)
topframe12.grid(row=1,column=0)

```

```

self.toptittle11=Label(topframe11, font=('stylus',40,'bold'),text="\tPAYROLL MANAGEMENT
SYSTEM\t",bd=10,justify=CENTER)
self.toptittle11.grid(padx=72)

```

```

scroll_x=Scrollbar(topframe12,orient=HORIZONTAL)
scroll_y=Scrollbar(topframe12,orient=VERTICAL)
self.payroll_rec=ttk.Treeview(topframe12,height=22,columns=("ref","Fullname","Address","Cityweighting",
"Basicsalary","Overtime","Grosspay","Pension","Postcode","Gender","Payday","Netpay","Ninumber","Nipay
ment","Taxcode","Taxperiod","Tax"),xscrollcommand=scroll_x.set,yscrollcommand=scroll_y.set)
scroll_x.pack(side=BOTTOM,fill=X)
scroll_y.pack(side=RIGHT,fill=Y)
self.payroll_rec.heading("ref",text="Ref")

```

```

self.payroll_rec.heading("Fullname",text="Fullname")

```

```

self.payroll_rec.heading("Address",text="Address")
self.payroll_rec.heading("Cityweighting",text="Cityweighting")
self.payroll_rec.heading("Basicsalary",text="Basicsalary")
self.payroll_rec.heading("Overtime",text="Overtime")
self.payroll_rec.heading("Grosspay",text="Grosspay")
self.payroll_rec.heading("Pension",text="Pension")
self.payroll_rec.heading("Postcode",text="Postcode")
self.payroll_rec.heading("Gender",text="Gender")
self.payroll_rec.heading("Payday",text="Payday")
self.payroll_rec.heading("Netpay",text="Netpay")
self.payroll_rec.heading("Ninumber",text="Ninumber")
self.payroll_rec.heading("Nipayment",text="Nipayment")
self.payroll_rec.heading("Taxcode",text="Taxcode")
self.payroll_rec.heading("Taxperiod",text="Taxperiod")
self.payroll_rec.heading("Tax",text="Tax")
self.payroll_rec['show']='headings'

```

```

#=====
self.payroll_rec.column("ref",width=70)
self.payroll_rec.column("Fullname",width=70)
self.payroll_rec.column("Address",width=70)
self.payroll_rec.column("Cityweighting",width=70)
self.payroll_rec.column("Basicsalary",width=70)
self.payroll_rec.column("Overtime",width=70)
self.payroll_rec.column("Grosspay",width=70)
self.payroll_rec.column("Pension",width=70)
self.payroll_rec.column("Postcode",width=70)
self.payroll_rec.column("Gender",width=70)
self.payroll_rec.column("Payday",width=70)
self.payroll_rec.column("Netpay",width=70)
self.payroll_rec.column("Ninumber",width=70)
self.payroll_rec.column("Nipayment",width=70)
self.payroll_rec.column("Taxcode",width=70)
self.payroll_rec.column("Taxperiod",width=70)
self.payroll_rec.column("Tax",width=70)
self.payroll_rec.pack(fill=BOTH,expand=1)
self.payroll_rec.bind("<ButtonRelease-1>",Wagesinfo)
displaydata()

```

```

#=====
topframe13=Frame(Tab3,bd=10,width=1340,height=100,relief=RIDGE)
topframe13.grid(row=0,column=0)

```



```

self.toptittle13=Label(topframe13,
font=('stylus',40,'bold'),text="\tPAYROLLCOMMENTBOOK\t",bd=10,justify=CENTER)
self.toptittle13.grid(row=0,column=0)
#=====
self.txtcmt=Text(topframe13,width=100,height=22,font=('stylus',14,'bold'))
self.txtcmt.grid(row=1,column=0)

```

#CREATING MAIN FUNCTION

```

if __name__=='__main__':
    root=Tk()
    application = Payroll(root)

```

RESULT 1

PAYROLL SYSTEM | [VIEW PAYMENTS](#) | [COMMENT BOOK](#)

PAYROLL MANAGEMENT SYSTEM

EMPLOYEE NAME

ADDRESS

POST CODE

GENDER

PAY DAY

TAX PERIOD

TAX CODE

NI NUMBER

TAXABLE PAY

PENSIONABLE PAY

NETT PAY

7	8	9	+
4	5	6	-
1	2	3	x
0	C	=	/

WAGESADDUPDATE

DELETERESETEXIT

REFERENCE

EMPLOYER NAME

EMPLOYEE NAME

Data Entered Successfully!!

SUCCESSFULLY UPDATED

OK

TAX

PENSION

STUDENT LOAN

NI PAYMENT

GROSS PAY

DEDUCTIONS

Fig.5.1 Result 1

RESULT 2

PAYROLL MANAGEMENT SYSTEM																
Ref	Fullname	Address	Cityweightii	Basicsalary	Overtime	Grosspay	Pension	Postcode	Gender	Payday	Netpay	Ninumber	Nipayment	Taxcode	Taxperiod	Tax
RP57103	Kathir	156,Kanmar	76	10000	2500	\$12576.00	\$251.52	625011	Male	09/12/2022	\$8262.43	NI361153	\$138.34	TCODE1293	12	\$3772.80
RP23391	Nakshatara	19/6 Anbun	34	11000	2500	\$13534.00	\$270.68	626101	Female	09/12/2022	\$8891.84	NI153332	\$148.87	TCODE1597	12	\$4060.20
RP54864	Sheak	1d3,jothi na	35	12000	500	\$12535.00	\$250.70	628501	Male	09/12/2022	\$8235.49	NI491402	\$137.88	TCODE1687	12	\$3760.50
RP26518	Jayalakshmi	143,Jeevana	20	12000	0	\$12020.00	\$240.40	625011	Female	09/12/2022	\$7897.14	NI73394	\$132.22	TCODE2176	12	\$3606.00
RP41279	Aazam	156,Lakshm	90	11000	1500	\$12590.00	\$251.80	65467	Male	09/12/2022	\$8271.63	NI543903	\$138.49	TCODE3075	12	\$3777.00
RP23467	Raja	145,Rameez	67	9000	3500	\$12567.00	\$251.34	62350	Male	09/12/2022	\$8256.52	NI297202	\$138.24	TCODE6488	12	\$3770.10

Fig.5 .2 Result 2

CONCLUSION

We gained lot of coding experience in designing forms and managing a database from this project. We would like to thank our HOD for giving this wonderful opportunity. Hope this project will help improve our career in future.