

**Faculty of Engineering and Technology**

**Department of Computer Science and Engineering**

Jain Global Campus, Kanakapura Taluk - 562112  
Ramanagara District, Karnataka, India

**A Project Report on**

**“MIRANGI PRINTS DATABASE SYSTEM”**

**For the partial fulfilment of**

**Bachelor of Technology**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted by**

**RAJAT SURANA**

**17BTRCS053**

**TANISHQ DALAL**

**17BTRCS055**

****

**Faculty of Engineering & Technology**

**Department of Computer Science and Engineering**

Jain Global campus

Kanakapura Taluk - 562112  
Ramanagara District

Karnataka, India

**CERTIFICATE**

This is to certify that the Project work titled **“MIRANGI PRINTS DATABASE SYSTEM”** for the course **Scripting Language Lab** during **5th semester,** is carried out by **RAJAT SURANA(17BTRCS053), TANISHQ DALAL(17BTRCS055)** are bonafide students at the Faculty of Engineering & Technology, JAIN (Deemed-to-be-University), Bangalore in partial fulfilment for the award of degree in Bachelor of Technology in Computer Science and Engineering, during the year **2019 - 2020**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prof. Deepa.T.P.** |  | **Prof. Prashanth** |  | **Head of the Department** |
| Assistant Professor  Dept. of Computer Science and Engineering ,  Faculty of Engineering & Technology,  JAIN (Deemed-to-be-University)  Date:05-12-2019 |  | Assistant Professor  Dept. of Computer Science and Engineering ,  Faculty of Engineering & Technology,  JAIN (Deemed-to-be-University)  Date::05-12-2019 |  | Dept. of Computer Science and Engineering ,  Faculty of Engineering & Technology,  JAIN (Deemed-to-be-University)  Date:05-12-2019 |

**ABSTRACT**

The Project "**MIRANGI DATABASE SYSTEM**" is actually a wholesale market system.This WHOLESALE MARKET SYSTEM is a wide area market system needs big databses system.So this is about the workers working in this company.This database includes the details of the worker which are currently working in this company.Worker Details like their name,their address,date of birth,phone\_number and some more personal details.In this databse,we can INSERT any new data,DELETE any data,UPDATE any existing data.More or over we can also delete the existing data like example in case if any worker resign this company.We can also see the worker details using that DISPLAY button in the frame.

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
|  |  |
| **Chapter 1** | **01** |
| 1. **Introduction** | **01** |
| 1. Tittle | 01 |
| 1. Description | 01 |
| 1. Problem Defination | 01 |
| 1. Objectives | 01 |
| 1. Hardware and Software | 01 |
|  |  |
| **Chapter 2** | **03** |
| 1. **Implementation** | **03** |
| 2.1 Design and Implementation | 03 |
| 2.2 Major considerations for Implementations | 04 |
| 2.2.1 Implementation Mechanism | 04 |
| 2.2.2 Source Code | 05 |
|  |  |
| **Chapter 3** | 14 |
| 1. **Results And Discussion** | 14 |
|  |  |
| **Chapter 4** | 16 |
| 1. **Conclusion** | **16** |
|  |  |
| **References** | 17 |
|  |  |

**Chapter 1**

**Introduction**

**1.1-TITTLE:**

**MIRANGI PRINTS DATABSE SYSTEM**

**1.2-Description of the MIRANGI PRINTS:**

MIRANLGI PRINTS is a wholesale saree market system with a more than 100 workers currently working in this company.

So as it is a large scale market this needs a big databse system which includes the details of the workers which are currently working in this company or which are previously worked for this company.

**1.3-Problem Definations:**

Many a times it is not easy to trackdown the details which we requires from the workers.So always calling them and getting the details from the workers is not possible.

So if we make a databse of all that workers whiich are working in this company,we can easily trackdown their details from them.

**1.4-Objectives:**

The main objectives of this database is one can easily operate or can track the worker details which are working in this company.

It is easily managable task provided by this database system.

**1.5-Hardware and Software used:**

**SOFTWARE USED:**

Operating system: Windows 10/Linux (Ubuntu 19.04)

Frontend: IDLE(Python 3.8)

GUI: Tkinter

IIS : Configured

Backend: SQLite3

**HARDWARE USED:**

1. Intel Core

2. 40GB hard disk

3. 8 GB RAM

RECOMMENDED:

1. Intel Core

2. 20 GB hard disk

3. 1 GB RAM

**Chapter 2**

**Implementation**

* 1. **Design & Implementation:**

• Implementation Approaches

The Software Design Description Document has been used as input in the implementation process. The actual implementation of the system has been done using Python (Tkinter), which is a GUI based application language. Tkinter and ttk has been used to interact with the backend database. In this implementation, MySQLite3 has been used as the backend. Tkinter processes the inputs or commands given by the user and translates them in the commands understandable to the backend database. The output produced by the backend database are also handled by Tkinter. Since Tkinter and MySQLite3 are both available in varied platforms like Windows, Unix, Macintosh and Linux, Criminal Record System can be implemented in a truly platform-independent manner.

This databse is about the worker details.This is WHOLESALE SAREE MARKET database system.

Includes their unique workerId, their full name, their address, their phone number and many more details of the workers which is related to the company.

We can **INSERT** more worker details if any new worker join this company.

We can **DELETE** worker details if in case any worker resign from this company.

More or over we can **UPDATE** worker details also.

The database will **display** all the details about all the workers which are present in this company or moreover we can **search** for a particular worker using their unique workerId.

We can update all the details of particular worker by **clearing** present values.

• Code Efficiency

Efficiency is the number of computing resources and code required by a program to perform its functions. Efficient codes are required for the better performance of the system. Efficient coding makes a system robust. A method is robust if it does not fail even if it receives improper parameters. Robustness against internal bugs may be a trade-off against efficiency. The need for efficiency arises due to the cost of consideration. If some resources are scarce and expensive, it is desirable that those resources should be used efficiently. In the computer system, the resources the most often considered for efficiency are processor time and less memory. This coding for the system has been started after completing the design phase because all software methodologies emphasize the importance of first designing then coding. The codes that construct this system are structured, modular, efficient, and require less amount of computing resources. All functions of this system are understandable since the functions are small and coherent. Functions and variables are meaningful variable names to increase the readability and avoided abbreviations that may confuse the users

**2.2 Major considerations for Implementation**

Some points are kept in mind while making the project:

• The code repeating again was made as a module which is common for all.

• Proper indentation is given to understand the code.

• The forms are designed in a way that all the data and buttons are properly viewed and spaced.

• Users can easily see and understand.

• Any person can use it easily as it is user-friendly.

**2.2.1. Implementation Mechanism:**

## Install MySQLite3 Driver:

Python needs a MySQLite3 driver to access the MySQLite database.

we will use the driver "MySQLite3 Connector".

We recommend that you use PIP to install "MySQLite3 Connector".

PIP is most likely already installed in your Python environment.

Navigate your command line to the location of PIP, and type the following:

Download and install "MySQLite3 Connector":

C:\Users\*Your Name*\AppData\Local\Programs\Python\Python36-32\Scripts>python -m pip install MySQLite3-connector

**2.2.2. SOURCE CODE:**

#FRONTEND:

from tkinter import \*

import tkinter.messagebox

import MirangiDatabase\_Backend

class Mirangi:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("MIRANGI\_PRINTS\_DATABASE")

self.root.geometry("1350x7500+0+0")

self.root.config(bg="cadet blue")

worker\_id = StringVar()

first\_name = StringVar()

last\_name = StringVar()

birth\_date = StringVar()

address = StringVar()

mobile\_number = StringVar()

#============================Functions==============================

def iExit():

iExit = tkinter.messagebox.askyesno("MIRANGI\_PRINTS\_DATABASE","Bubyee buddy!see u later.")

if iExit > 0:

root.destroy()

return

def clearData():

self.txtworker\_id.delete(0,END)

self.txtfirst\_name.delete(0,END)

self.txtlast\_name.delete(0,END)

self.txtbirth\_date.delete(0,END)

self.txtaddress.delete(0,END)

self.txtmobile\_number.delete(0,END)

def addData():

if(len(worker\_id.get())!=0):

MirangiDatabase\_Backend.addwdata(worker\_id.get(), first\_name.get(), last\_name.get(), birth\_date.get(), address.get(), mobile\_number.get())

workerlist.delete(0,END)

workerlist.insert(END,(worker\_id.get(), first\_name.get(), last\_name.get(), birth\_date.get(), address.get(), mobile\_number.get()))

def DisplayData():

workerlist.delete(0,END)

for row in MirangiDatabase\_Backend.viewData():

workerlist.insert(END,row,str(""))

def WorkerRec(event):

global wk

searchWorker = workerlist.curselection()[0]

wk = workerlist.get(searchWorker)

self.txtworker\_id.delete(0,END)

self.txtworker\_id.insert(END,wk[0])

self.txtfirst\_name.delete(0,END)

self.txtfirst\_name.insert(END,wk[1])

self.txtlast\_name.delete(0,END)

self.txtlast\_name.insert(END,wk[2])

self.txtbirth\_date.delete(0,END)

self.txtbirth\_date.insert(END,wk[3])

self.txtaddress.delete(0,END)

self.txtaddress.insert(END,wk[4])

self.txtmobile\_number.delete(0,END)

self.txtmobile\_number.insert(END,wk[5])

def DeleteData():

if(len(worker\_id.get())!=0):

MirangiDatabase\_Backend.deleteRec(wk[0])

clearData()

DisplayData()

def searchDatabase():

workerlist.delete(0,END)

for row in MirangiDatabase\_Backend.searchData(worker\_id.get(), first\_name.get(), last\_name.get(), birth\_date.get(), address.get(), mobile\_number.get()):

workerlist.insert(END,row,str(""))

def update():

if(len(worker\_id.get())!=0):

MirangiDatabase\_Backend.deleteRec(wk[0])

if(len(worker\_id.get())!=0):

MirangiDatabase\_Backend.addwdata(worker\_id.get(), first\_name.get(), last\_name.get(), birth\_date.get(), \

address.get(), mobile\_number.get())

workerlist.delete(0,END)

workerlist.insert(END,(worker\_id.get(), first\_name.get(), last\_name.get(), birth\_date.get(), \

address.get(), mobile\_number.get()))

#=====================================================FREMES===========================================================

MainFrame = Frame(self.root, bg="cadet blue")

MainFrame.grid()

TitFrame = Frame(MainFrame, bd=2, padx=54,pady=8, bg="Ghost White", relief=RIDGE)

TitFrame.pack(side=TOP)

self.lblTit = Label(TitFrame, font=('arial', 42,'bold'), text="MIRANGI PRINTS DATABASE MANAGEMENT SYSTEM", bg="Ghost White")

self.lblTit.grid()

ButtonFrame = Frame(MainFrame, bd=2, width=1350, height=70, padx=18,pady=10, bg="Ghost White", relief=RIDGE)

ButtonFrame.pack(side=BOTTOM)

DataFrame = Frame(MainFrame, bd=1, width=1300, height=400, padx=20,pady=20, bg="cadet blue", relief=RIDGE)

DataFrame.pack(side=BOTTOM)

DataFrameLeft = LabelFrame(DataFrame, bd=1, width=1000, height=600, padx=20, bg="Ghost White", font=('arial', 20,'bold'), text="WORKERS AUTHENTICATIONS\n", relief=RIDGE)

DataFrameLeft.pack(side=LEFT)

DataFrameRight = LabelFrame(DataFrame, bd=1, width=450, height=300, padx=31,pady=3, bg="Ghost White", relief=RIDGE, font=('arial', 20,'bold'), text="WORKERS DETAILS\n")

DataFrameRight.pack(side=RIGHT)

#=============================LABELS\_AND\_ENTRIES=================

self.lblworker\_id = Label(DataFrameLeft, font=('arial', 20,'bold'), text="WorkerID:", padx=2,pady=2, bg="Ghost White")

self.lblworker\_id.grid(row=0, column=0, sticky=W)

self.txtworker\_id = Entry(DataFrameLeft, font=('arial', 20,'bold'), textvariable=worker\_id, width=39, bg="Ghost White")

self.txtworker\_id.grid(row=0, column=1)

self.lblfirst\_name = Label(DataFrameLeft, font=('arial', 20,'bold'), text="FirstName:", padx=2,pady=2, bg="Ghost White")

self.lblfirst\_name.grid(row=1, column=0, sticky=W)

self.txtfirst\_name = Entry(DataFrameLeft, font=('arial', 20,'bold'), textvariable=first\_name, width=39, bg="Ghost White")

self.txtfirst\_name.grid(row=1, column=1)

self.lbllast\_name = Label(DataFrameLeft, font=('arial', 20,'bold'), text="LastName:", padx=2,pady=2, bg="Ghost White")

self.lbllast\_name.grid(row=2, column=0, sticky=W)

self.txtlast\_name = Entry(DataFrameLeft, font=('arial', 20,'bold'), textvariable=last\_name, width=39, bg="Ghost White")

self.txtlast\_name.grid(row=2, column=1)

self.lblbirth\_date = Label(DataFrameLeft, font=('arial', 20,'bold'), text="Birth\_Date:", padx=2,pady=2, bg="Ghost White")

self.lblbirth\_date.grid(row=3, column=0, sticky=W)

self.txtbirth\_date = Entry(DataFrameLeft, font=('arial', 20,'bold'), textvariable=birth\_date, width=39, bg="Ghost White")

self.txtbirth\_date.grid(row=3, column=1)

self.lbladdress = Label(DataFrameLeft, font=('arial', 20,'bold'), text="Address:", padx=2,pady=2, bg="Ghost White")

self.lbladdress.grid(row=4, column=0, sticky=W)

self.txtaddress = Entry(DataFrameLeft, font=('arial', 20,'bold'), textvariable=address, width=39, bg="Ghost White")

self.txtaddress.grid(row=4, column=1)

self.lblmobile\_number = Label(DataFrameLeft, font=('arial', 20,'bold'), text="PhoneNumber:", padx=2,pady=2, bg="Ghost White")

self.lblmobile\_number.grid(row=5, column=0, sticky=W)

self.txtmobile\_number = Entry(DataFrameLeft, font=('arial', 20,'bold'), textvariable=mobile\_number, width=39, bg="Ghost White")

self.txtmobile\_number.grid(row=5, column=1)

#==============LISTS AND SCROLLBAR WIDGET========================

scrollbar = Scrollbar(DataFrameRight)

scrollbar.grid(row=0, column=1, sticky='ns')

workerlist = Listbox(DataFrameRight, width=41, height=16, font=('arial', 12,'bold'), yscrollcommand=Scrollbar.set)

workerlist.bind('<<ListboxSelect>>', WorkerRec)

workerlist.grid(row=0, column=0, padx=8)

scrollbar.config(command = workerlist.yview)

#=========================BUTTON\_WIDGET=========================

self.btnAddData = Button(ButtonFrame, text="Add New", font=('arial', 20,'bold'), height=1,width=10, bd=4, command=addData)

self.btnAddData.grid(row=0, column=0)

self.btnDisplayData = Button(ButtonFrame, text="Display", font=('arial', 20,'bold'), height=1,width=10, bd=4, command=DisplayData)

self.btnDisplayData.grid(row=0, column=1)

self.btnClearData = Button(ButtonFrame, text="Clear", font=('arial', 20,'bold'), height=1,width=10, bd=4, command=clearData)

self.btnClearData.grid(row=0, column=2)

self.btnDeleteData = Button(ButtonFrame, text="Delete", font=('arial', 20,'bold'), height=1,width=10, bd=4, command=DeleteData)

self.btnDeleteData.grid(row=0, column=3)

self.btnSearchData = Button(ButtonFrame, text="Search", font=('arial', 20,'bold'), height=1,width=10, bd=4, command=searchDatabase)

self.btnSearchData.grid(row=0, column=4)

self.btnUpdateData = Button(ButtonFrame, text="Update", font=('arial', 20,'bold'), height=1,width=10, bd=4, command=update)

self.btnUpdateData.grid(row=0, column=5)

self.btnExit = Button(ButtonFrame, text="Exit", font=('arial', 20,'bold'), height=1,width=10, bd=4, command=iExit)

self.btnExit.grid(row=0, column=6)

if \_\_name\_\_ == '\_\_main\_\_':

MirangiDatabase\_Backend.workerDATA()

root = Tk()

application = Mirangi(root)

root.mainloop()

#BACKEND:

import sqlite3

#backEND

def workerDATA():

con=sqlite3.connect("worker.db")

cur = con.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS worker (worker\_id text PRIMARY KEY, first\_name text, last\_name text,birth\_date text, address text, mobile\_number text)")

con.commit()

con.close()

def addwdata(worker\_id, first\_name, last\_name, birth\_date, address, mobile\_number):

con=sqlite3.connect("worker.db")

cur = con.cursor()

cur.execute("INSERT INTO worker VALUES (?,?,?,?,?,?)",(worker\_id, first\_name, last\_name, birth\_date, address, mobile\_number))

con.commit()

con.close()

def viewData():

con=sqlite3.connect("worker.db")

cur = con.cursor()

cur.execute("SELECT \* FROM worker")

row = cur.fetchall()

con.close()

return row

def deleteRec(worker\_id):

con=sqlite3.connect("worker.db")

cur = con.cursor()

print(worker\_id)

cur.execute("DELETE FROM worker WHERE worker\_id=?"(worker\_id))

con.commit()

con.close()

def searchData(worker\_id="", first\_name="", last\_name="", birth\_date="", address="", mobile\_number=""):

con=sqlite3.connect("worker.db")

cur = con.cursor()

cur.execute("SELECT \* FROM worker WHERE worker\_id=? OR first\_name=? OR last\_name=? OR birth\_date=? OR address=? OR mobile\_number=?", (worker\_id, first\_name, last\_name, birth\_date, address, mobile\_number))

row = cur.fetchall()

con.close()

return row

def dataUpdate(worker\_id="", first\_name="", last\_name="", birth\_date="", address="", mobile\_number=""):

print(worker\_id)

con=sqlite3.connect("worker.db")

cur = con.cursor()

cur.execute("UPDATE worker SET worker\_id=?, first\_name=?, last\_name=?, birth\_date=?, address=?, mobile\_number=?", (worker\_id, first\_name, last\_name, birth\_date, address, mobile\_number))

con.commit()

con.close()

**Chapter 3**

**Results and Discussion**

User Documentation

The system is designed for saving the worker details which are working in this company.The

System can INSERT any new data,DELETE any data or can UPDATE a data into the the database.

The application has Project Module that shows the overview of project and provides an interface to go to all the operations that can be performed on the software.

* The application have ADD module to insert any new Worker Details.
* The application have DISPLAY module to see the Worker Details.
* The application have DELETE module to delete the worker details.
* The application have SEARCH module to search any particular worker details.

Final product:

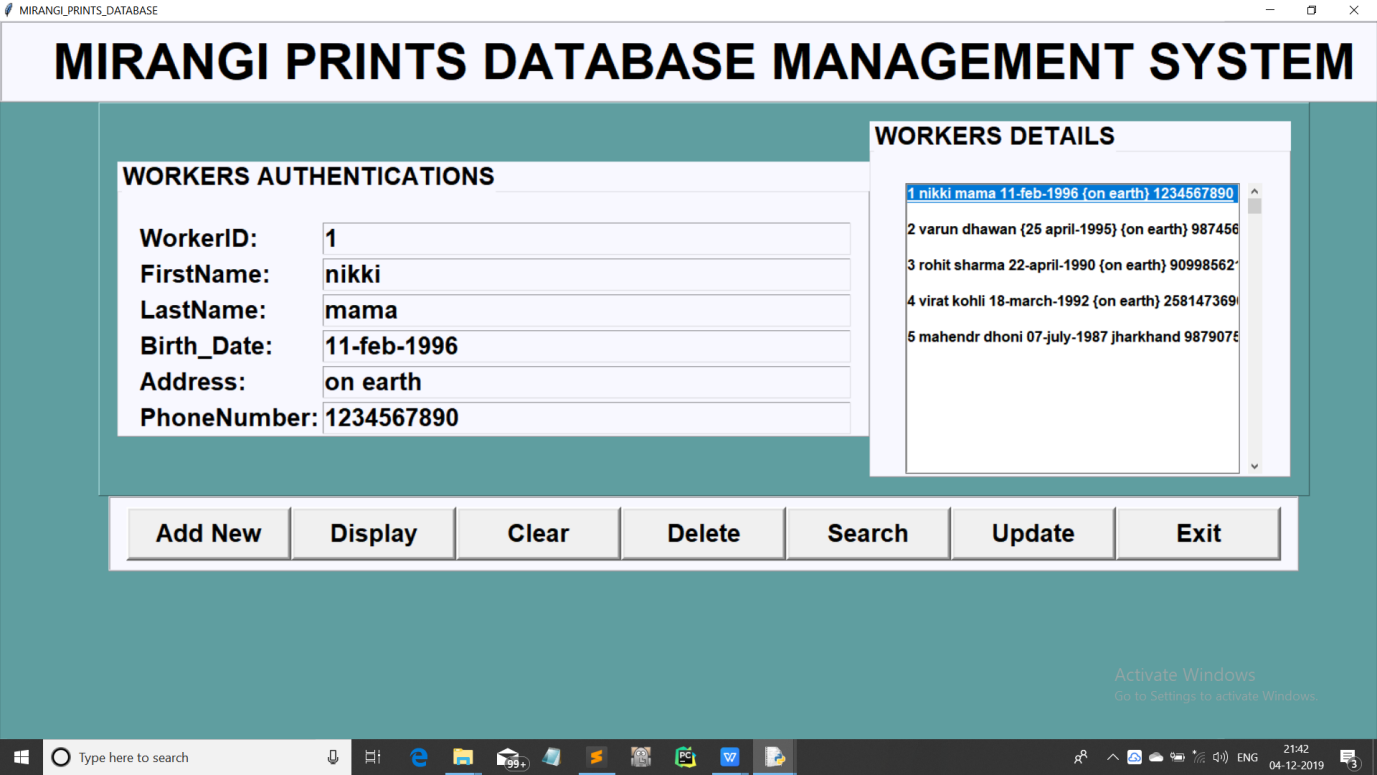


Fig 3.1: User Interface Displaying Present Data

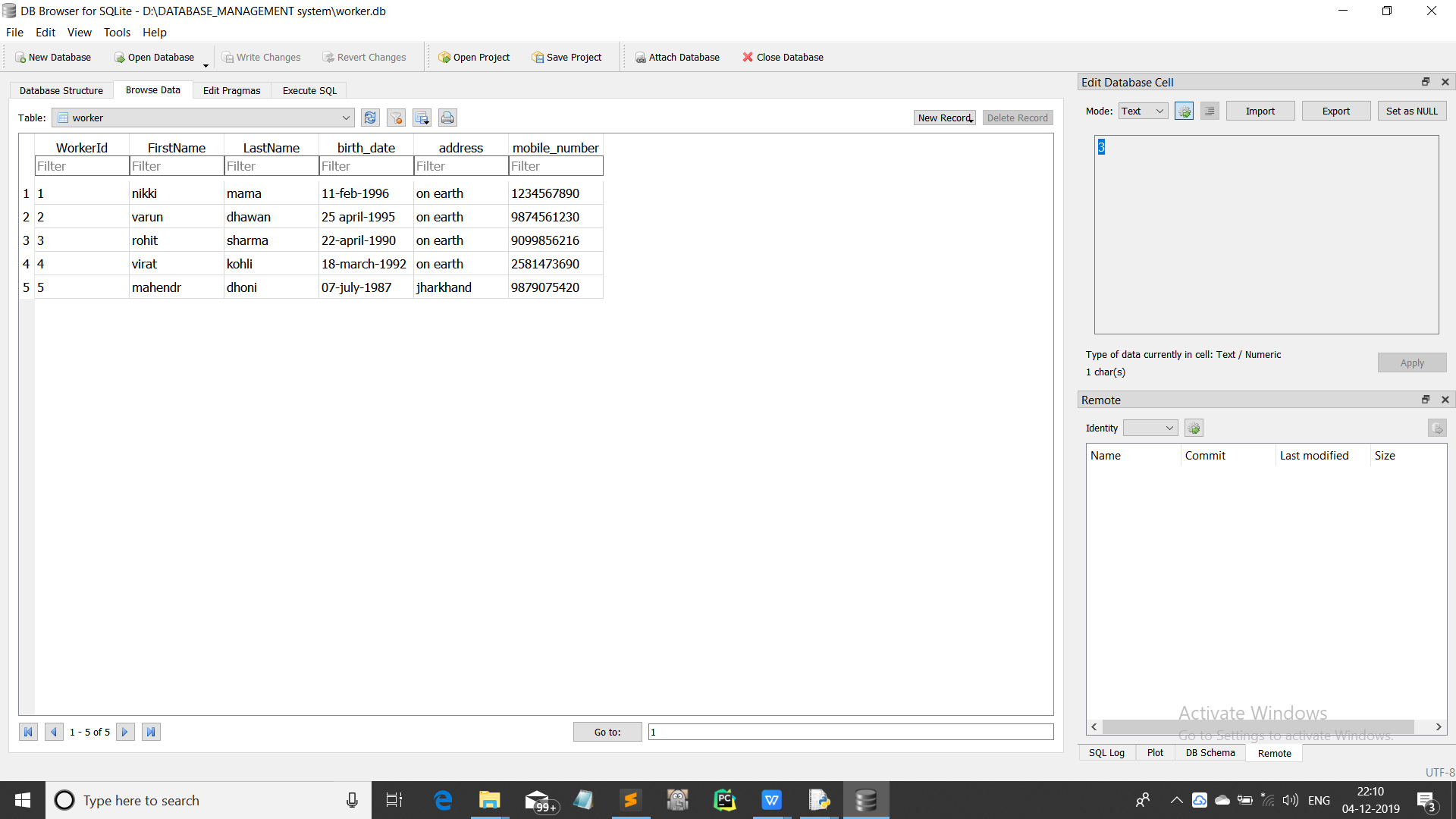


Fig 3.2: Data saved in worker Database

**Chapter 4**

**Conclusion**

Here we concluded that we can save permanently or temporarily the data of the workers in the database.Our intention is to make this simpler so that it would be easy for saving and tracking the worker details of the workers which are present in this company.

**References**

1) https://www.geeksforgeeks.org/ - geeks for geeks

2)https://www.w3schools.com/ -w3schools

3)https://stackoverflow.com/ -StackOverflow

4)python documentation

5)Tkinter documentation

6)MySQLite3 documentation