Coverpage Format

Application Design for an app launch on Google PlayStore



Submitted by: **Maanav Chetty**  
Date of Submission: 3rd January 2023

College Name & Logo:

**Thakur College of Engineering and Technology** 

**Certificate of completion**

This is to certify that, Mr. \_\_Maanav Chetty\_\_ has successfully implemented an application designed to study the data and generate insights for an app launch on Google PlayStore.

The Application has been accepted as a completed project as it meets all the requirements specified.

**Acknowledgements:**

I would like to express my sincere gratitude to my H.O.D Harshali Patil & Dr. Manish Rana for providing their invaluable guidance, comments and suggestions throughout the course of the project. I would specially thank our H.O.D Harshali Patil for constantly motivating me to work harder and Dr. Manish Rana for helping me with the Matplotlib parts and the graphs.

Also, I would like to thank Mrs. Rashmi Thakur for his assistance in the SQL Queries and to help me retrieve all the relevant Data & Mr. Nitin Harane for his help during the preparation of the Project Report and The Testing phase of the project, Mrs. Deepali Joshi for providing me an opinion how to improve the user experience on the App along with keeping the U.I perfect.

I would like to dedicate a special vote of thanks to our principal, our T.P.O and My Teacher Guardian to make me keep going throughout the internship and allowing me to do the outhouse Work from Home Internship during this Quarantine period.

|  |  |  |
| --- | --- | --- |
| Sr no | Section | Information |
| 1 | Section 1: | System Requirement Specifications |
| 2 | Section 2: | Technology used |
| 3 | Section 3: | Data Provided by the client: |
| 4 | Section 4: | Screenshots: |
| 5 | Section 5: | Testing: |
| 6 | Section 6: | Final code: |
|  |  |  |

Section 1:

System Requirement Specifications

(in this section explain in details, what is the system to be designed, requirements and the desired results)

Section 2:

Technology used

(Here mention about thew technologies and components being used and why have you chosen that technology)

Section 3:

Data Provided by the client:

(in this section add a snapshot of the data provided and your observations about the data. Also mention the database software that you are using.)

Section 4:

Screenshots:

(here add all screen shots with their respective codes) Section 5:

Testing:

(in this section, mention all the tests that you have carried out to ensure that the system works fine. Also mention about the validations that you have implemented)

Section 6:

Final code:

(add the complete source code)

Section 1:

System Requirement Specifications

System Requirement Specifications

* What is System requirement Specification?

1. A System Requirements Specification (SyRS) (abbreviated SysRS when need to be distinct from a software requirements specification (SRS) is a structured collection of information that embodies the requirements of a system.
2. System requirements specification is a document that describes what the system will do and how it will be expected to perform. It also describes the functionality the product needs to fulfill all stakeholders (business, users) needs.
3. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide.
4. A business analyst, sometimes titled system analyst, is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions.
5. Within the systems development life cycle domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers.
6. System requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function (in a market-driven project, these roles may be played by the marketing and development divisions).
7. System requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules.
8. In addition to specifying how the system should behave, the specification also defines at a high-level the main business processes that will be supported, what simplifying assumptions have been made and what key performance parameters will need to be met by the system

* Why to use a System requirement specification

1. We use the system requirements specification to validate and check the product to ensure that it has the required features. Development should start from a specification.
2. If somehow the delivered software doesn’t meet the requirements, the specification serves as a reference, and the development team works to meet all the described needs.
3. It helps the development team during the design and implementation of the product.
4. Moreover, early detection of problems in specification leads to effective time management since it is a lot easier to update specifications prior to any development than to update the specification then the corresponding functionalities.
5. A good specification makes the product easier to update. Any change in the software requires updating the project requirement specification inviting every party involved in the process to rethink the changes to be made.

* Types of Requirements:



* The System to be Designed:

1. The System needs to be designed to make an all feature containing dashboard which can catch the sight of each and every type of user during their time on the app.
2. The System has Different buttons which have different functionality when clicked on each one of them.
3. The System gives its user a very charismatic Insights of the company’s performance to its user.
4. The System gives complete analysis of company’s conduct in the past few years and not only about the company but it also gives a complete report on the achievement of their employees that is the agents.
5. The System gives a Complete Report that highlights the balance amount of all the orders and the name and code of the agent handling the order.
6. The System gives the Company A Search based Order Lookup Based upon the entries of Order number, Order Date and Customer Code.
7. The Company gets A Complete Insights of its past performances in the previous year’s just by selection of the year.
8. The following are the insights provided to the user about the company:
9. The graph of total property sold v/s total property leased of all the years.
10. The graph in which year the company got maximum leased area in CA and WS countries.
11. The Graph of Agent who got the deals in the OWNED Category all around the years.
12. The agent which has got maximum deals in leased form in the city of Chilliwack.
13. The graph of agents who has been the best performer always around the years.
14. The Amount of property area sold for the month of July for each and every year.
15. The Time Series Analysis report of all the order received by the company.

* System Requirements:
* Hardware Requirements:

1. Intel i5-6th Gen
2. 8GB Ram
3. DDR 4
4. 500 GB HDD
5. 4 GB Nvidia Graphics Card

* Software Requirements:

1. Windows Based 32-bit or 64-bit Operating System.
2. Python 2.8 or higher.
3. Wampp Server.
4. Microsoft Excel
5. Anaconda (Jupyter or Spyder) or Visual Studio 2018 or Any Python Text Editor.

* The Desired System Results:

1. The System needs to be User friendly and should be easy to use. So that it can be easily used by any naïve user.
2. The System Designed needs to give accurate values and graphs and the Time Series Analysis Report to the company as well as to its user to focus and improve their performance in the upcoming future.
3. The System designed should be used for the benefit for the Company and the employees. It should help them to be prepared and work hard for their betterment.
4. The Visual Interface Designed should only and only allow its registered user, customer, company and agents and should have proper validation to prevent from hacking and cyber security.
5. The System should be error free and be prevent from all kinds of attacks and should be easy to use along with an antivirus software of the user’s choice.
6. The System should store the data of user’s login and log off time and help the company monitor his/her performance.

Section 2:

Technology Used

* Technology Used:

Here we have implemented the project through the PYTHON which is a programming language

* About it:

1. Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.
2. Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)).
3. It is used for:
4. software development
5. mathematics
6. system scripting
7. web development (server-side)
8. Many other paradigms are supported via extensions, including design by contract[ and logic programming.
9. Rather than having all of its functionality built into its core, Python was designed to be highly extensible
10. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.
11. Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology.
12. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.
13. Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block.
14. Python's large standard library, commonly cited as one of its greatest strengths provides tools suited to many tasks. For Internet-facing applications, many standard formats and protocols such as MIME and HTTP are supported. It includes modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, arithmetic with arbitrary-precision decimals manipulating regular expressions, and unit testing.

* Why we have used it:

1. Readable and Maintainable Code:
2. While writing a software application, you must focus on the quality of its source code to simplify maintenance and updates. The syntax rules of Python allow you to express concepts without writing additional code. At the same time, Python, unlike other programming languages, emphasizes on code readability, and allows you to use English keywords instead of punctuations.
3. Hence, you can use Python to build custom applications without writing additional code. The readable and clean code base will help you to maintain and update the software without putting extra time and effort
4. Multiple Programming Paradigms:
5. Like other modern programming languages, Python also supports several programming paradigm. It supports object oriented and structured programming fully. Also, its language features support various concepts in functional and aspect-oriented programming.
6. At the same time, Python also features a dynamic type system and automatic memory management. The programming paradigms and language features help you to use Python for developing large and complex software applications.
7. Compatible with Major Platforms and Systems:
8. At present, Python is supports many operating systems. You can even use Python interpreters to run the code on specific platforms and tools. Also, Python is an interpreted programming language.
9. It allows you to you to run the same code on multiple platforms without recompilation. Hence, you are not required to recompile the code after making any alteration.
10. You can run the modified application code without recompiling and check the impact of changes made to the code immediately. The feature makes it easier for you to make changes to the code without increasing development time.
11. Robust Standard Library:
12. Its large and robust standard library makes Python score over other programming languages. The standard library allows you to choose from a wide range of modules according to your precise needs.
13. Each module further enables you to add functionality to the Python application without writing additional code. For instance, while writing a web application in Python, you can use specific modules to implement web services, perform string operations, manage operating system interface or work with internet protocols.
14. You can even gather information about various modules by browsing through the Python Standard Library documentation
15. Many Open Source Frameworks and Tools:
16. As an open source programming language, Python helps you to curtail software development cost significantly. You can even use several open source Python frameworks, libraries and development tools to curtail development time without increasing development cost.
17. You even have option to choose from a wide range of open source Python frameworks and development tools according to your precise needs. For instance, you can simplify and speedup web application development by using robust Python web frameworks like Django, Flask, Pyramid, Bottle and Cherrypy
18. Likewise, you can accelerate desktop GUI application development using Python GUI frameworks and toolkits like PyQT, PyJs, PyGUI, Kivy, PyGTK and WxPython.
19. Adopt Test Driven Development:
20. You can use Python to create prototype of the software application rapidly. Also, you can build the software application directly from the prototype simply by refactoring the Python code.
21. You can easily write the required tests before writing code and use the tests to assess the application code continuously. The tests can also be used for checking if the application meets predefined requirements based on its source code.
22. However, Python, like other programming languages, has its own shortcomings. It lacks some of the built-in features provided by other modern programming language. Hence, you have to use Python libraries, modules, and frameworks to accelerate custom software development.
23. You have to speed up the Python application by making changes to the application code or using custom runtime. But you can always use Python to speed up software development and simplify software maintenance.

* Components:

Components in Python are nothing but the modules used throughout the project.

Modules:

1. Tkinter:

Tkinter is Python's de-facto standard GUI (Graphical User Interface) package. It is a thin object-oriented layer on top of TCl/Tk.

Tkinter is not the only Gui Programming toolkit for Python. It is however the most commonly used one. Cameron Laird calls the yearly decision to keep Tkinter "one of the minor traditions of the Python world."

1. Message box:

The tkinter. message box module provides a template base class as well as a variety of convenience methods for commonly used configurations. The message boxes are modal and will return a subset of (True, False, OK, None, Yes, No) based on the user’s selection.

Syntax:

Here is the simple syntax to create this widget −

tkMessageBox.FunctionName(title, message [, options])

1. TTK:

The tkinter.ttk module provides access to the Tk themed widget set, introduced in Tk 8.5. If Python has not been compiled against Tk 8.5, this module can still be accessed if Tile has been installed. The former method using Tk 8.5 provides additional benefits including anti-aliased font rendering under X11 and window transparency (requiring a composition window manager on X11).

The basic idea for tkinter.ttk is to separate, to the extent possible, the code implementing a widget’s behavior from the code implementing its appearance.

1. ImageTk:

Pillow is a functional drop-in replacement for the Python Imaging Library. To run your existing PIL-compatible code with Pillow, it needs to be modified to import the Imaging module from the PIL namespace instead of the global namespace. I.e. change:

1. O.S:

This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see [open()](https://docs.python.org/3/library/functions.html#open), if you want to manipulate paths, see the [os.path](https://docs.python.org/3/library/os.path.html#module-os.path) module, and if you want to read all the lines in all the files on the command line see the [fileinput](https://docs.python.org/3/library/fileinput.html#module-fileinput) module.

For creating temporary files and directories see the [tempfile](https://docs.python.org/3/library/tempfile.html#module-tempfile) module, and for high-level file and directory handling see the [shutil](https://docs.python.org/3/library/shutil.html#module-shutil) module.

1. Matplotlib:

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram.

1. Pandas:

Pandas is an open-source library that is built on top of NumPy library. It is a Python package that offers various data structures and operations for manipulating numerical data and time series. It is mainly popular for importing and analyzing data much easier. Pandas is fast and it has high-performance & productivity for users.

1. TkCalendar:

tkcalendar is a python module that provides the Calendar and DateEntry widgets for Tkinter. The DateEntry widget is similar to a Combobox, but the drop-down is not a list but a Calendar to select a date. Events can be displayed in the Calendar with custom colors and a tooltip displays the event list for a given day. tkcalendar is compatible with both Python 2 and Python 3. It supports many locale settings (e.g. ‘fr\_FR’, ‘en\_US’, ...) and the colors are customizable.

1. NumPy:

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

1. MySQL. Connector:

For any application it is very important to store database on a server for easy data access. It is quite complicated to connect to database remotely because every service provider doesn’t provide remote access to MySQL database. Here I am using python’s MySQL dB module for connecting to our database which is at any server that provide remote access.

MySQL dB is an interface for connecting to a MySQL database server from Python. It implements the Python Database API v2.0 and is built on top of the MySQL C API.

How to connect to remote MySQL database using python?

Before we start you should know basics of SQL. Now let us discuss methods used in this code:

connect (): This method is used for creating a connection to our database it has four arguments:

Server Name

Database User Name

Database Provider

Database Name

cursor (): This method creates a cursor object that is capable for executing sql query on database.

execute (): This method is used for executing sql query on database. It takes a sql query (as string) as an argument.

fetchone (): This method retrieves the next row of a query result set and returns a single sequence, or None if no more rows are available.

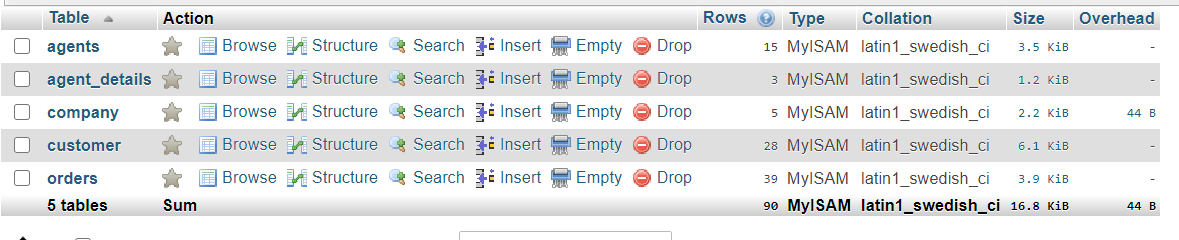
close (): This method closes the database connection.

Section 3:

Data Provided by the client:

In this project, we were given 2 dataset that is an Excel sheet containing all analytical data and an SQL dump containing all the information of the already registered agents, customers, companies and all their related order details.

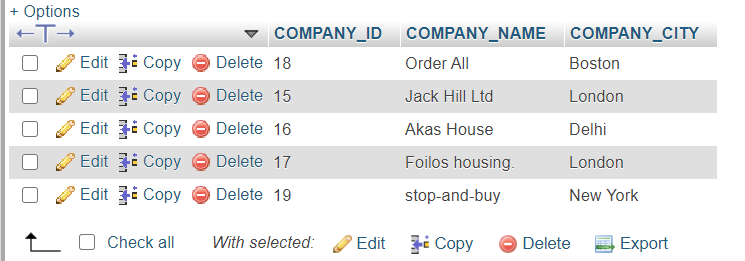
Screenshots of SQL Dump Database:



Agent Table:



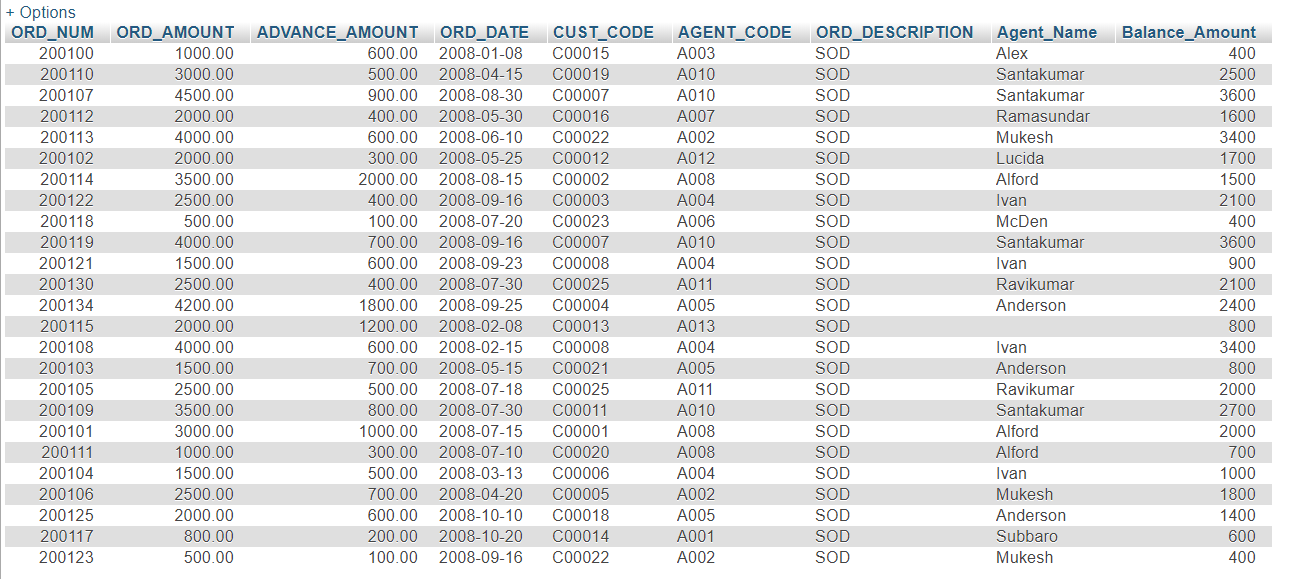
Company Table:



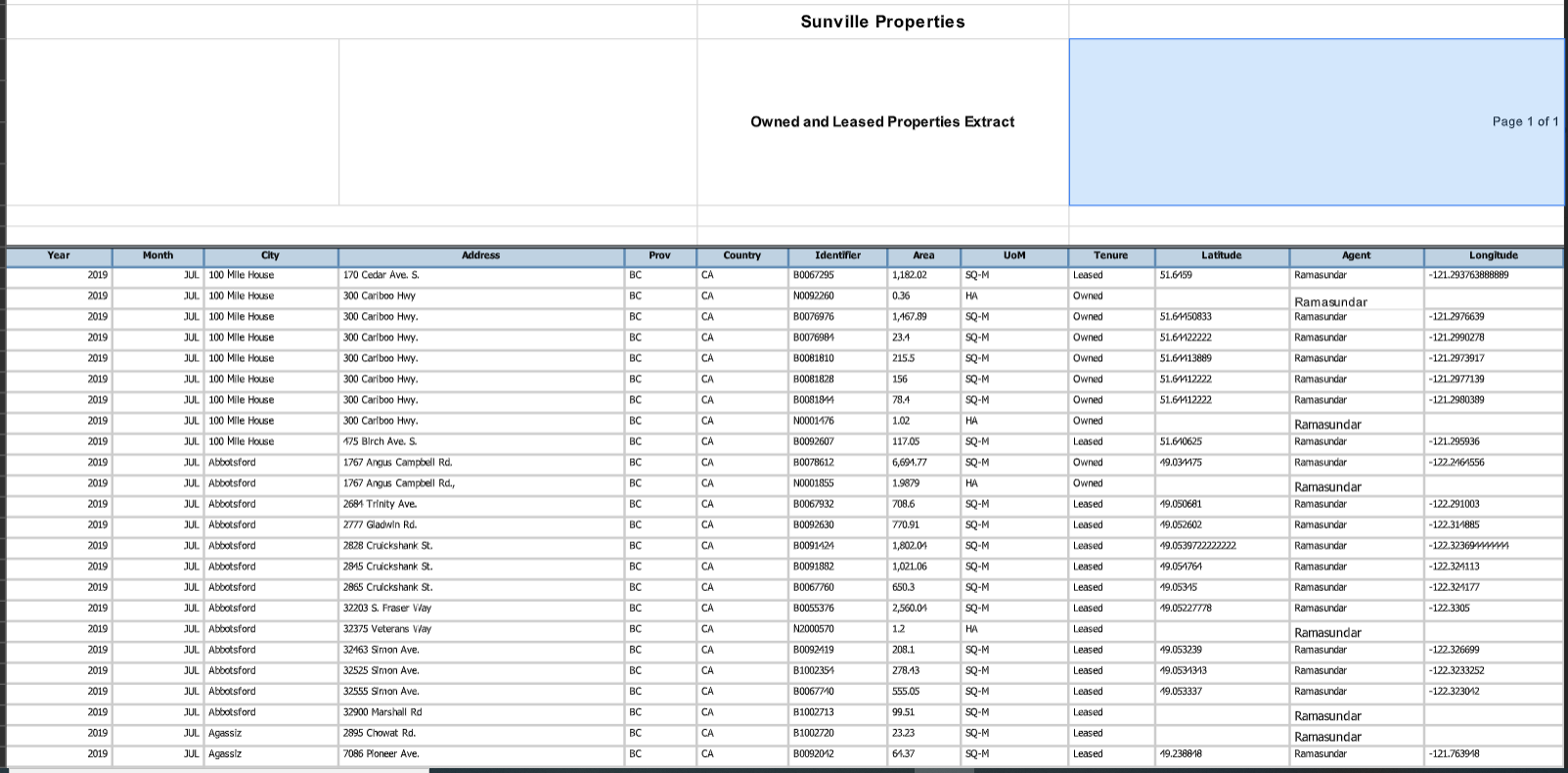
Customer Table:



Orders Table:



Screenshots of Excel Analytical data:



* Observation of Data:

* The Data provided in the excel sheet provides us all the details of order following are the helpful Insights:

1. It gives the proper year in which the order was completed.
2. It gives a proper insight of the order which was completed by which Agent and the information of the property if it was sold or leased.
3. It gives proper whereabouts of the location of property in which country it is placed.
4. It also provides the exact detailed address of the location providing the latitude and longitude of the property.
5. It also gives the exact area of property sold or given on leased by the agent.
6. It also has a column allotted to the Province in which the property was sold or leased.
7. It also shows the area of property was sold in which U.O.M SQ-M(Square-meter) OR H.A(Hectares).
8. It also has the proper month in which the property was sold by the Agent.
9. It gives the proper postal address of the property along with the City Name.
10. It also has a unique column which is Identifiers which uniquely identifies all the property in a unique way.
11. The graphs are made to show the insights in a pictorial format to help the company get a great knowledge about the data.

* The Data provided in the SQL Dump provides us all the details of the orders, customers, company, agents following are the helpful Insights:
* It gives Insights of the Orders table from the database in the following manner:

1. It gives each and every order a unique ORD\_NUMBER corresponding to the ORD\_NUM Column.
2. It stores the order amount of the order in figures and in a particular Column of ORD\_Amount.
3. It saves the advance amount related to the orders in the ADVANCE\_AMOUNT Column.
4. It reserves the date on which the order was placed in the ORD\_DATE Column.
5. It gives all his Customers a Unique code to identify them separately each time they make an order and it is saved in the Column named CUST\_CODE.
6. It gives the Order Description in the form of SOD and is stored in the Column ORD\_DESCRIPTION.
7. We have added 2 columns to get the name of the agent handling the order and the balance amount saved after the subtraction of ORD\_Amount and Advance\_Amount (ORD\_AMOUNT - ADVANCE\_AMOUNT) to keep the track how much the customer is required to pay.

* It gives the Insights of the Company table from the database in the following manner:

1. It has 3 columns Company\_ID, Company\_Name, Company\_City which is just used to keep the basic details of the registered company in a sorted manner.
2. The COMPANY\_ID Column uniquely identifies the ID of the company with which it registered itself.
3. The COMPANY\_NAME Column saves the Name of the Company used during registration.
4. The COMPANY\_CITY Column saves the city of the company in which it is located.

* It gives the Insights of the Customer Table from the database in the following manner:

1. It has 12 columns (CUST\_CODE, CUST\_NAME, CUST\_CITY, WORKING\_AREA, CUST\_COUNTRY, GRADE, OPENING\_AMT, RECEIVE\_AMT, PAYMENT\_AMT, OUTSTANDING\_AMT, PHONE\_NO, AGENT\_CODE) In the table customer.
2. It has 2 Columns which uniquely identifies the customer code and customer name of the registered customer into columns (CUST\_CODE, CUST\_NAME).
3. It has 2 Columns which identifies the area in which it is working and country of the registered customer into columns (CUST\_CITY, WORKING\_AREA).
4. It has 2 Columns which takes the Country and GRADE from the registered column and stores it into columns (CUST\_COUNTRY, GRADE).
5. It has 2 Columns which stores the opening amount and receive amount of the registered customer into columns (OPENING\_AMT, RECEIVE\_AMT).
6. It has 2 Columns which takes the outstanding\_amt and payment\_amt from the registered customer and stores it into columns (PAYEMENT\_AMT, OUTSTANDING\_AMT).
7. It has 2 Columns which takes the phone no. and agent\_code of the agent handling the order into columns (PHONE\_NO, AGENT\_CODE).

* It gives the Insights of the Agents Table from the database in the following manner:

1. It has 6 columns (AGENT\_CODE, AGENT\_NAME, WORKING\_AREA, COMMISSION, PHONE\_NO, COUNTRY) in the table Agents.
2. It has 2 columns which uniquely identifies each and every agent by their respective code and corresponding name into 2 Columns (AGENT\_CODE, AGENT\_NAME).
3. It has 2 columns which stores the working area of the agent and the Commission earned by the agent in the order into 2 columns (WORKING\_AREA, COMMISSION).
4. It has 2 columns which stores the phone no and the Country of the agent in which they are working into columns (PHONE\_NO, COUNTRY).

* WAMP SERVER (Database Software we used) :

It is the database software we used to save the Dump SQL in the phpMyAdmin server.

The SQL section in this server we passed on all the queries to execute and create Database and Create the required tables and the required data.

The server is best suited for SQL Database and it is easy to insert, select, delete and update the records in the database.

It has many options to import and export data from the database to create a dump and transfer it anyone’s computer.

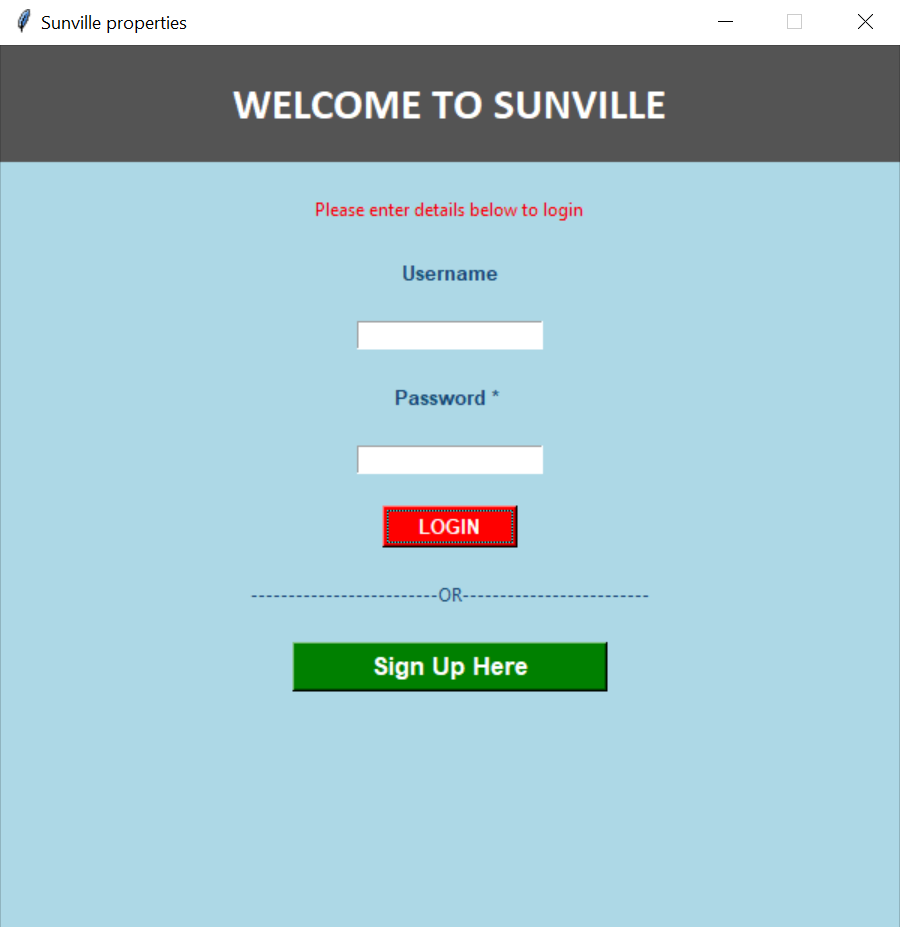
It can be used by any naïve user even if they do not have a great knowledge about SQL because it also provides its user to manipulate and create the database, tables.

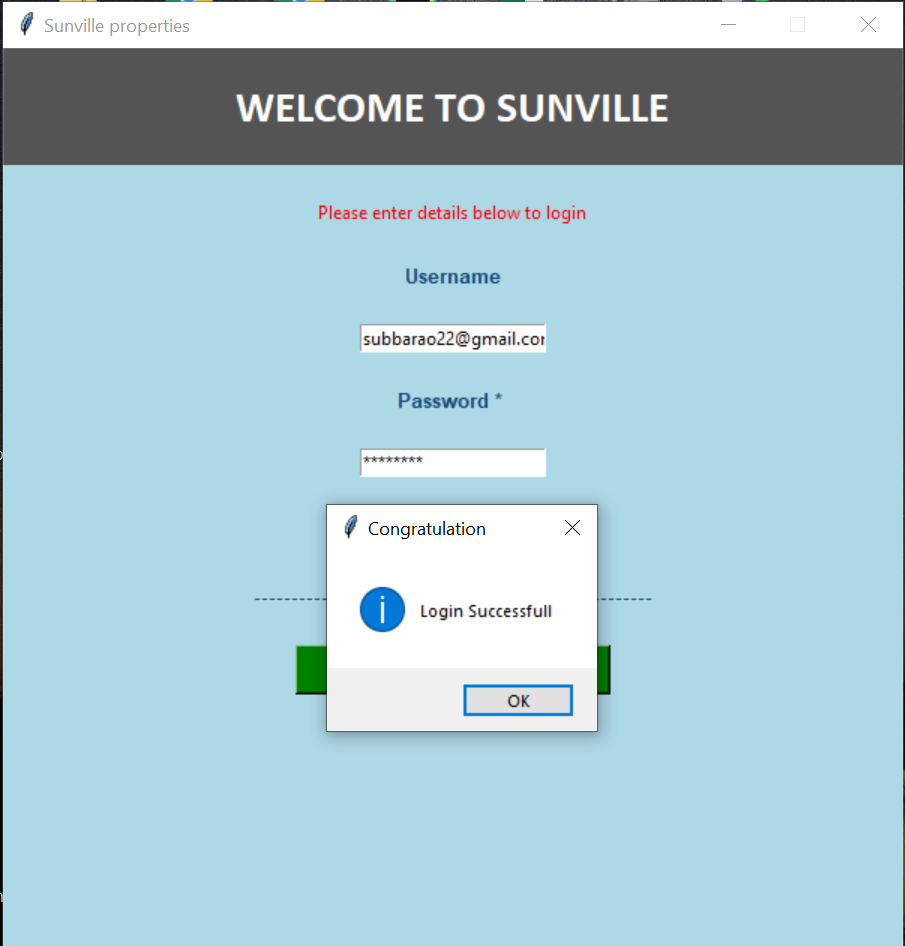
It can be used to insert data, delete the data manually without writing the Complete SQL Query.

Section 4:

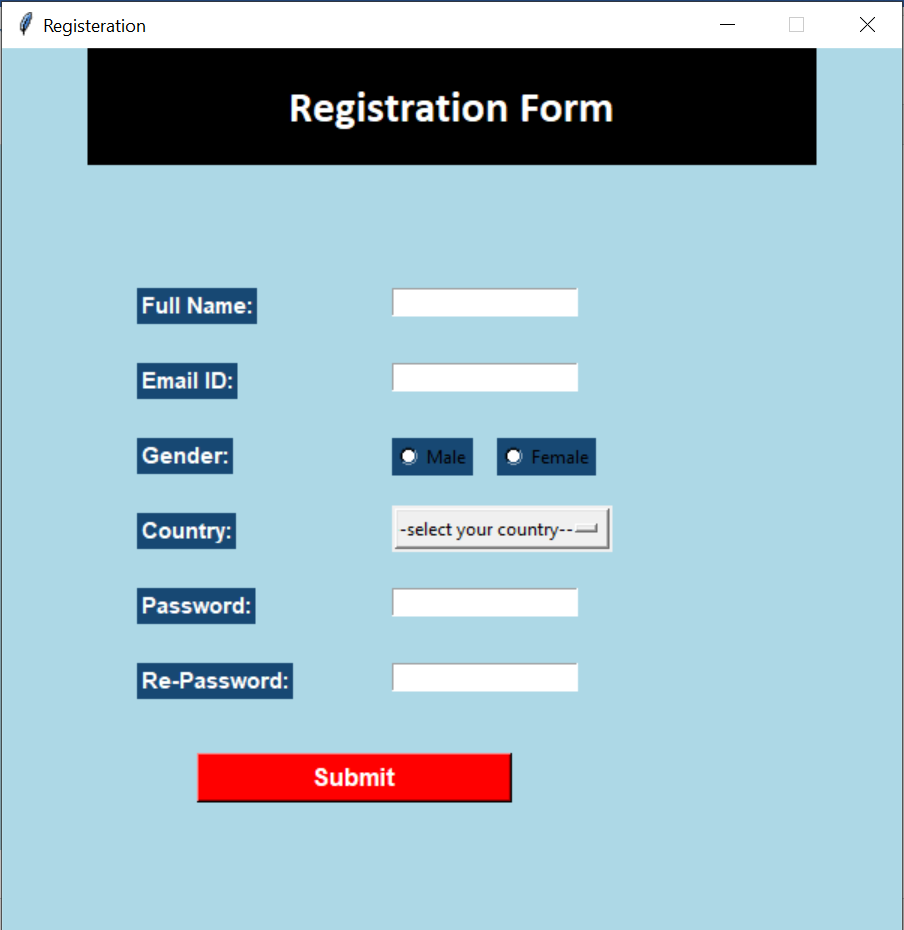
Screenshots:

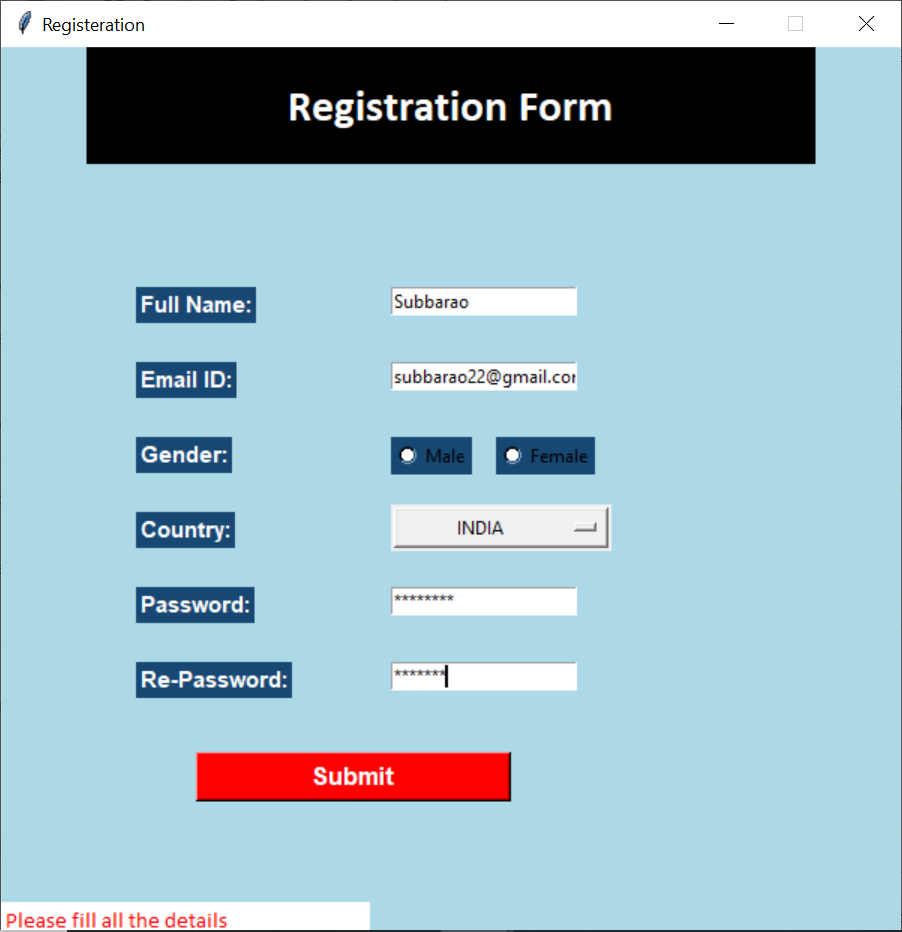
* For login page:





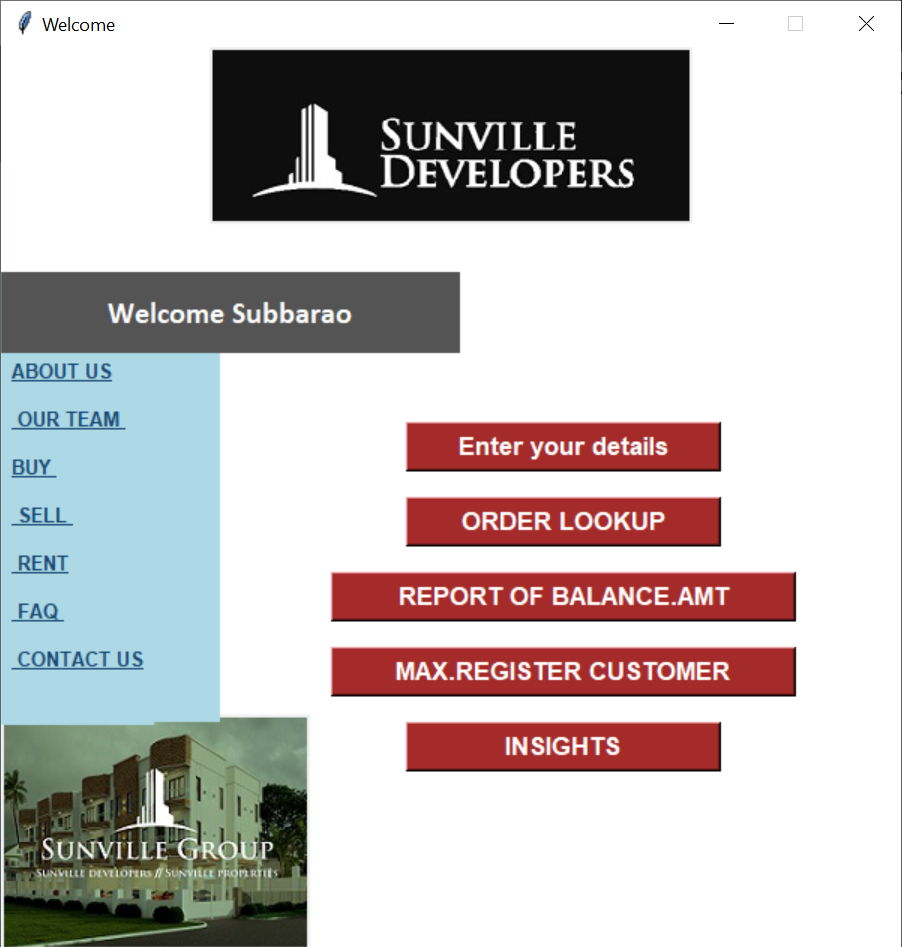
* Registration Form:



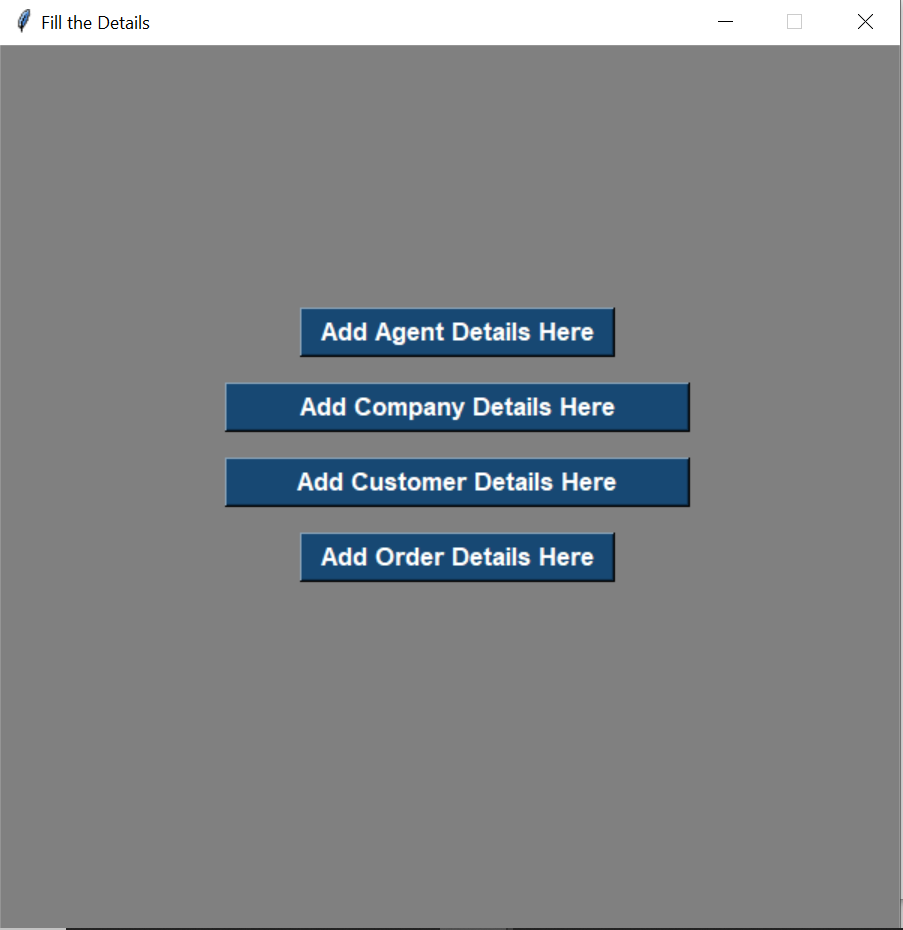




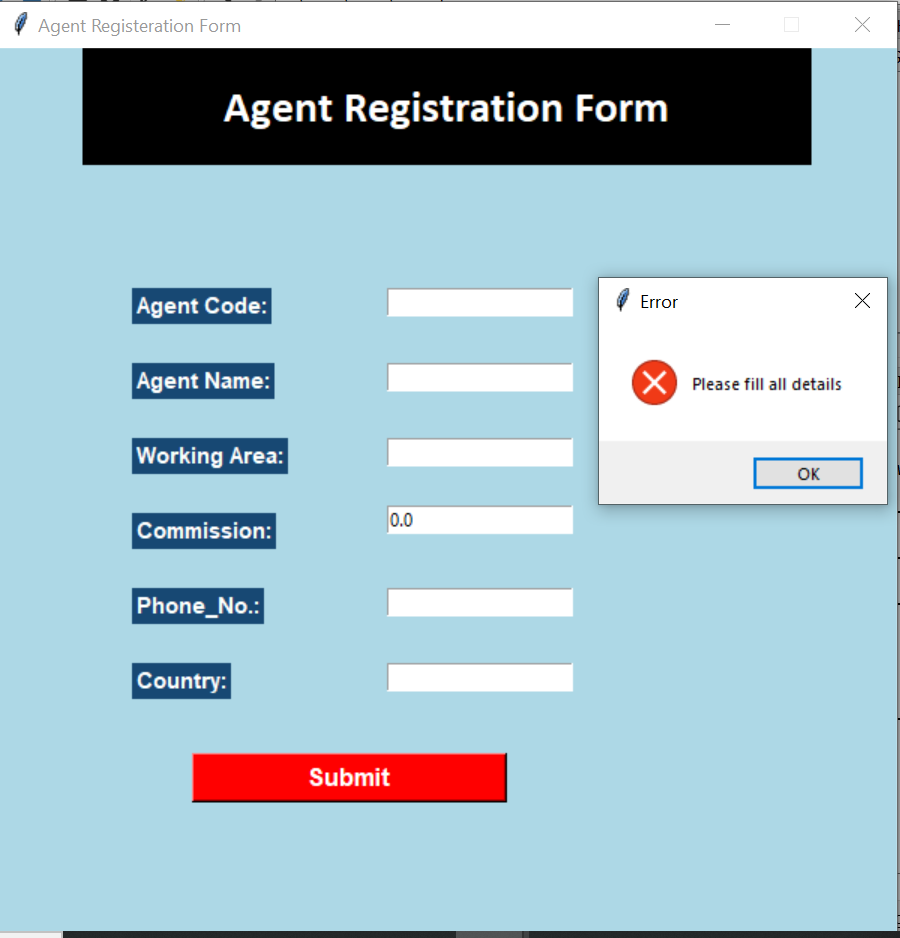
* Welcome Page:

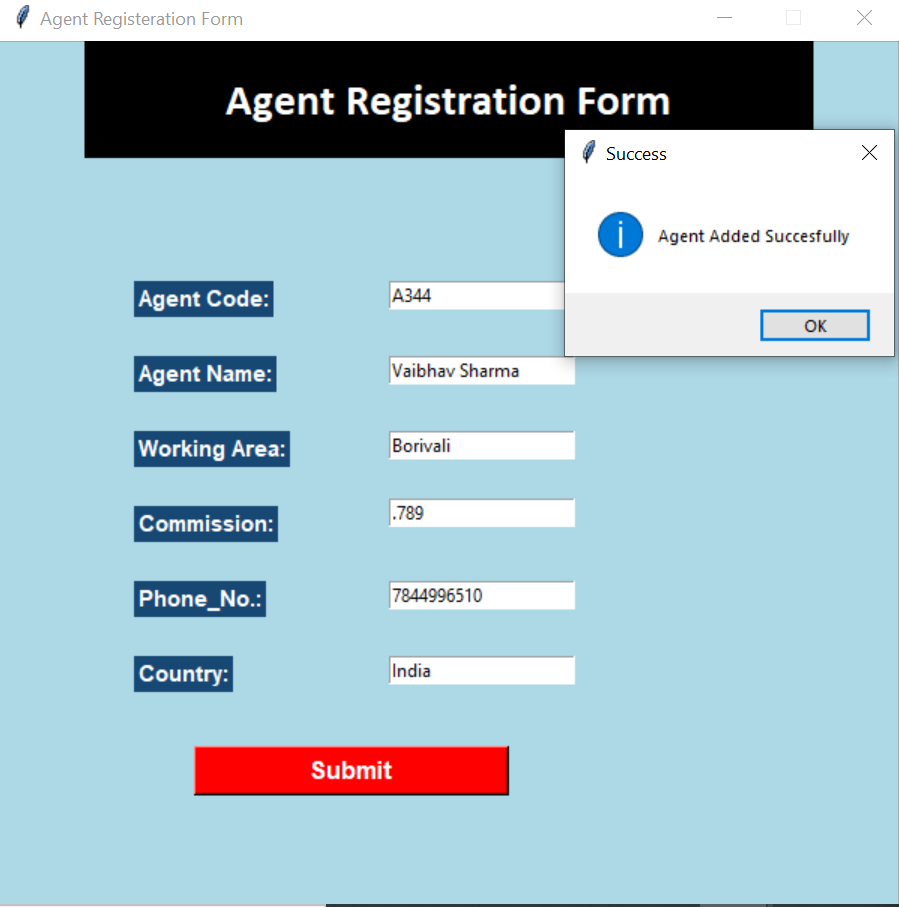


* Enter Your Details:

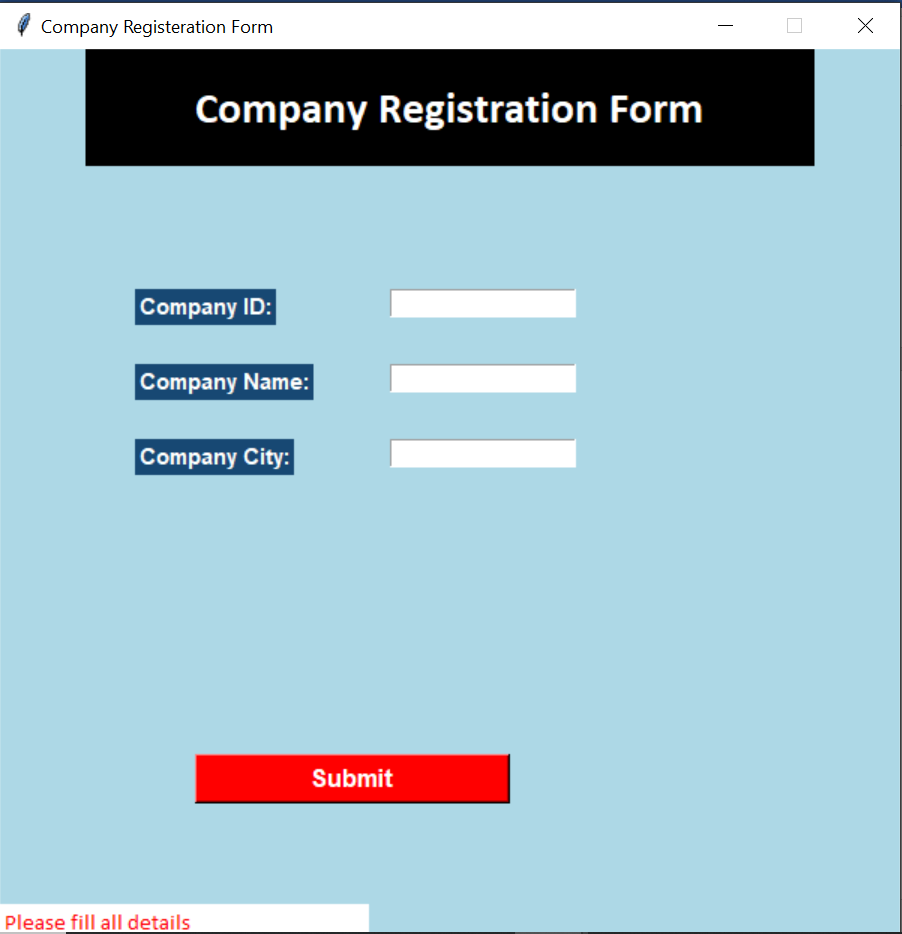


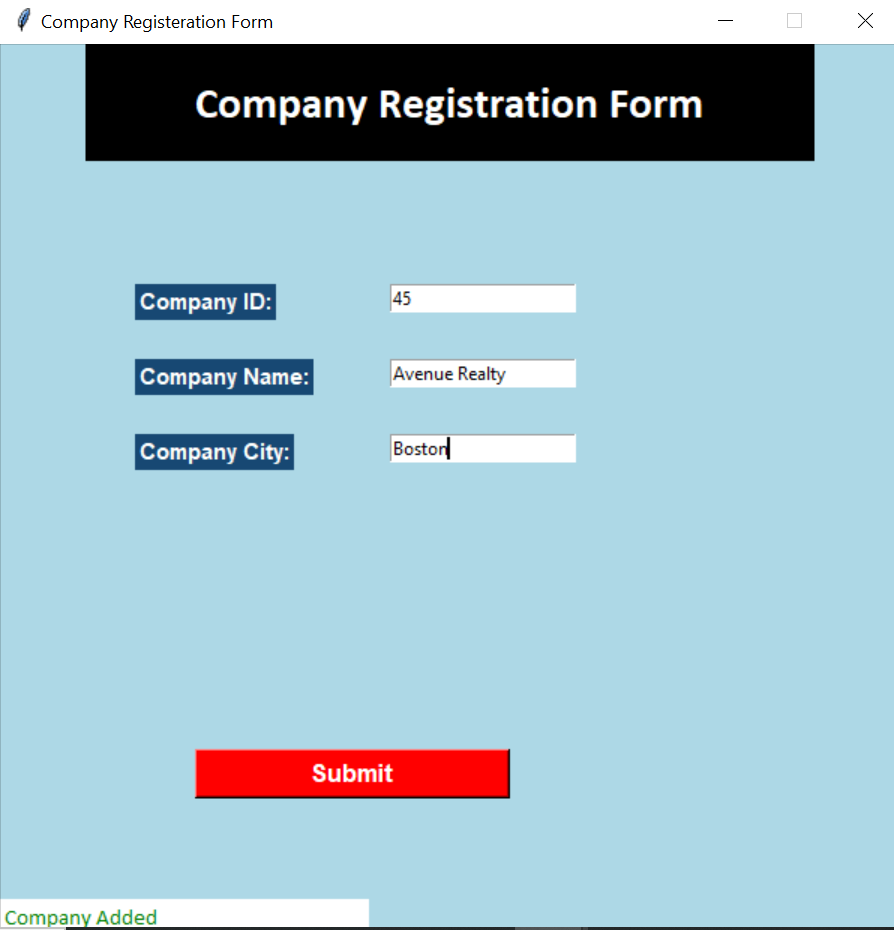
* Agent Details:



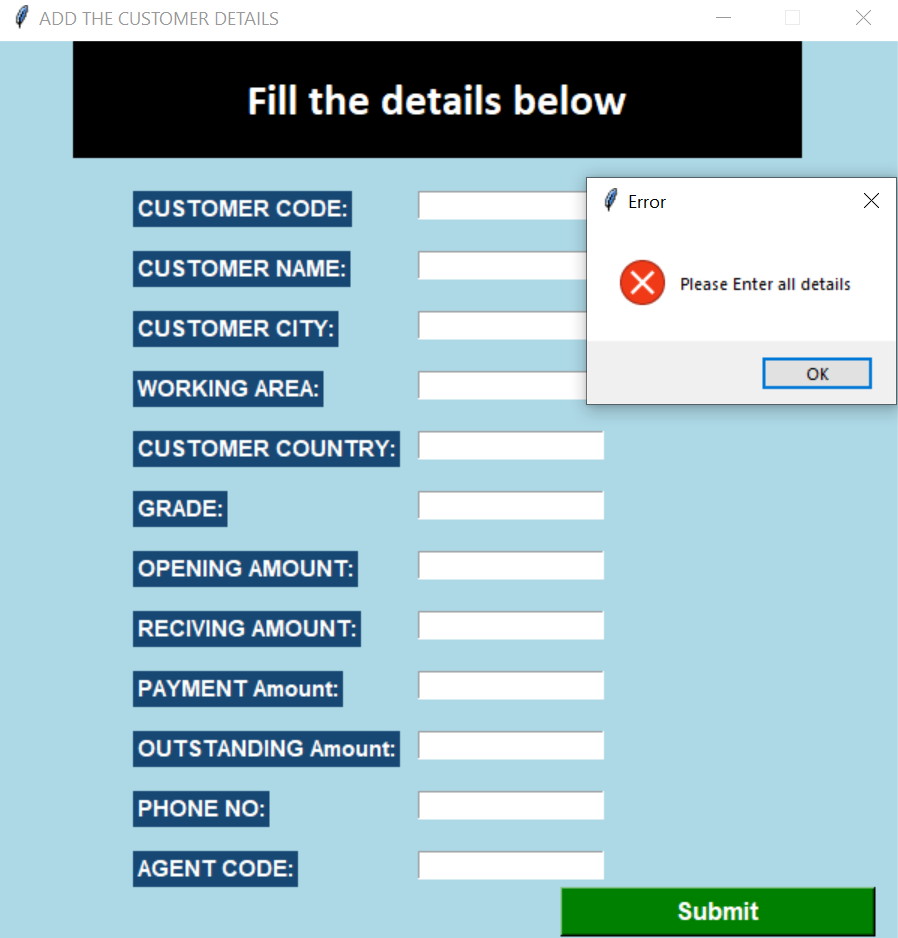


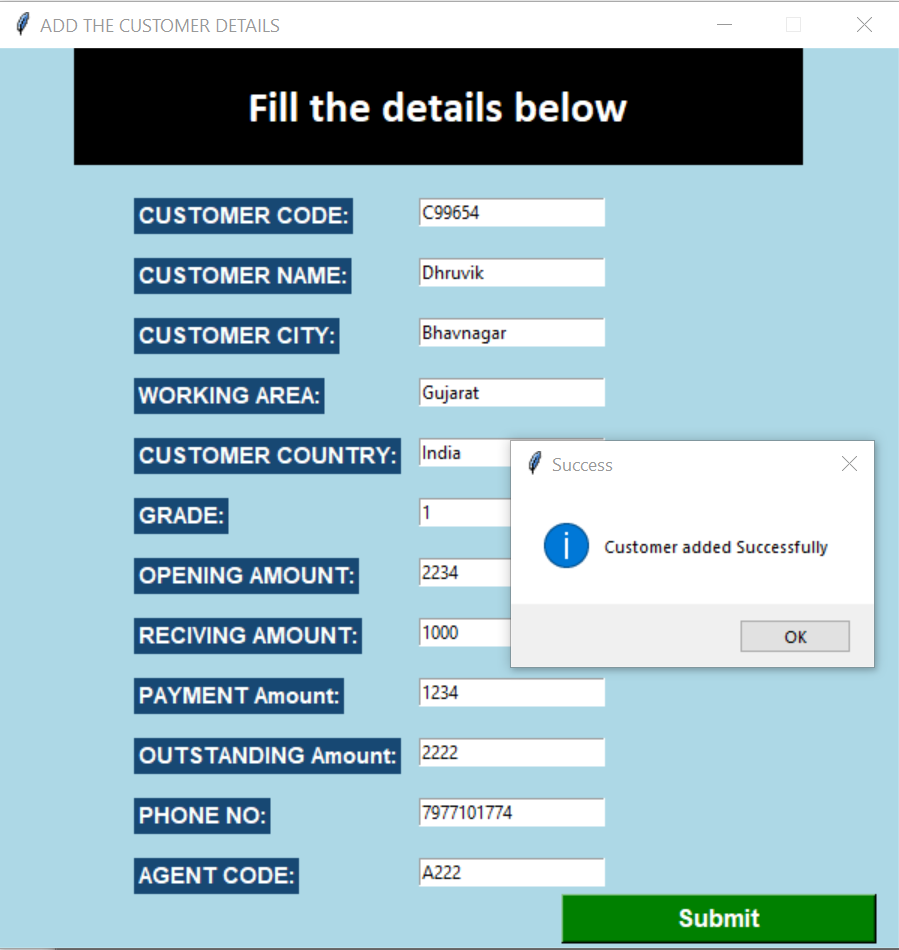
* Company Details



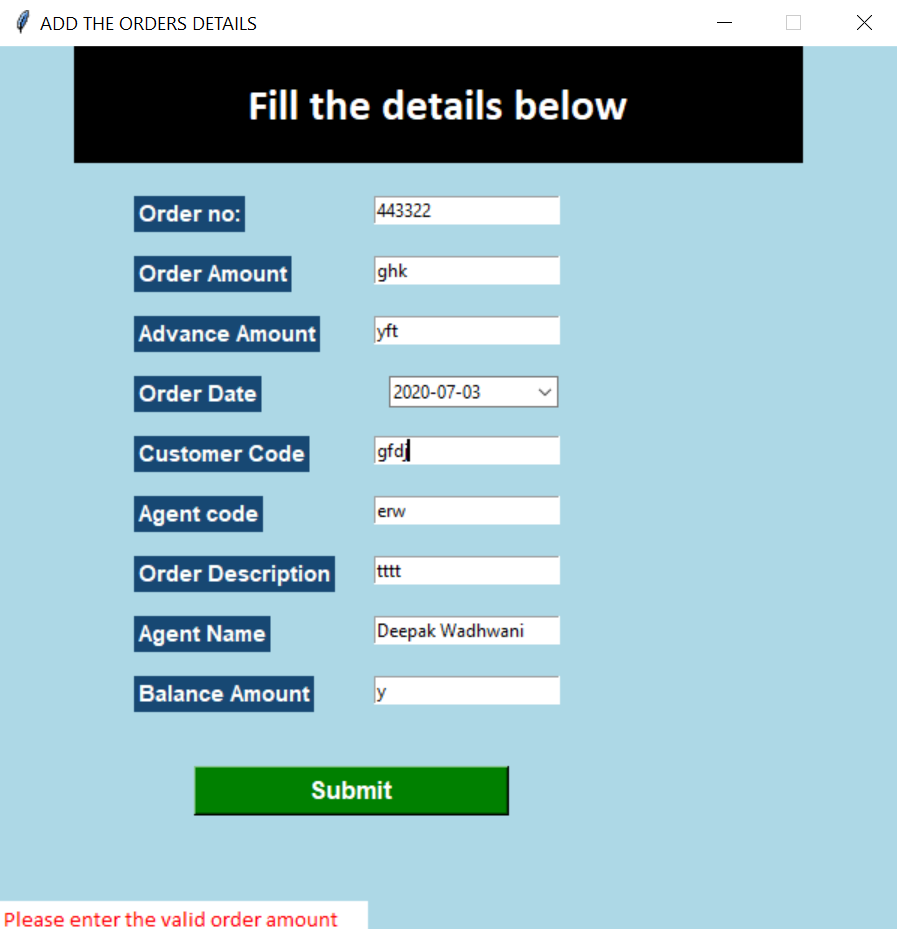


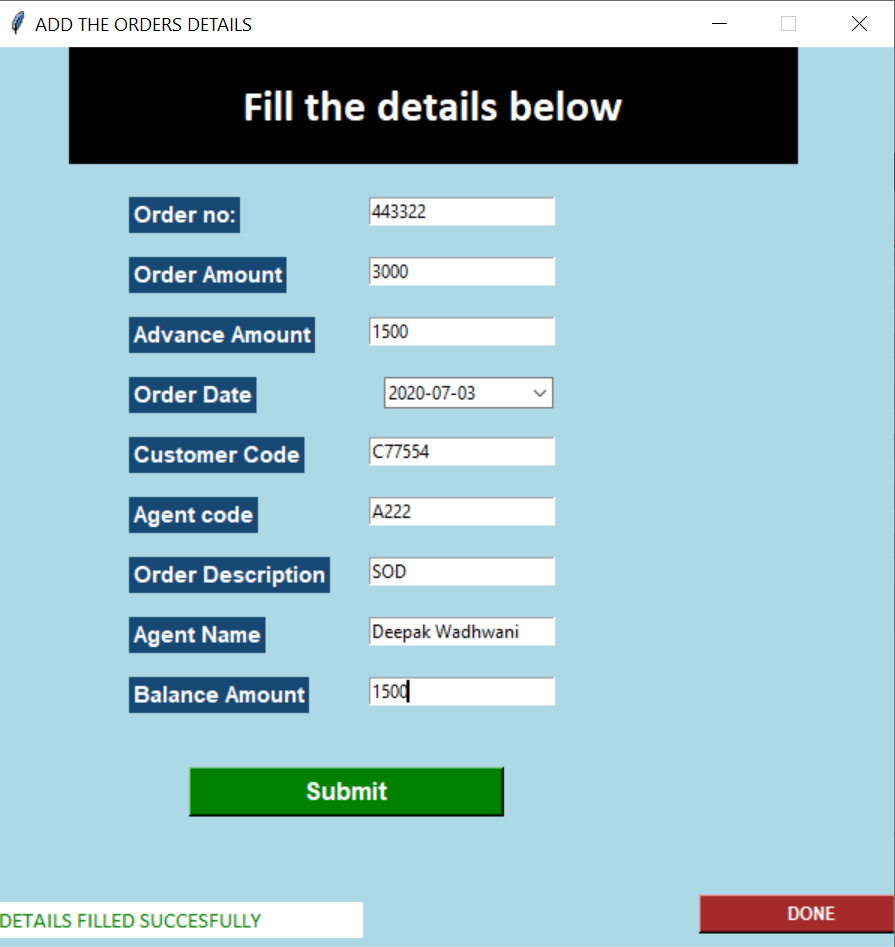
* Customer Details:



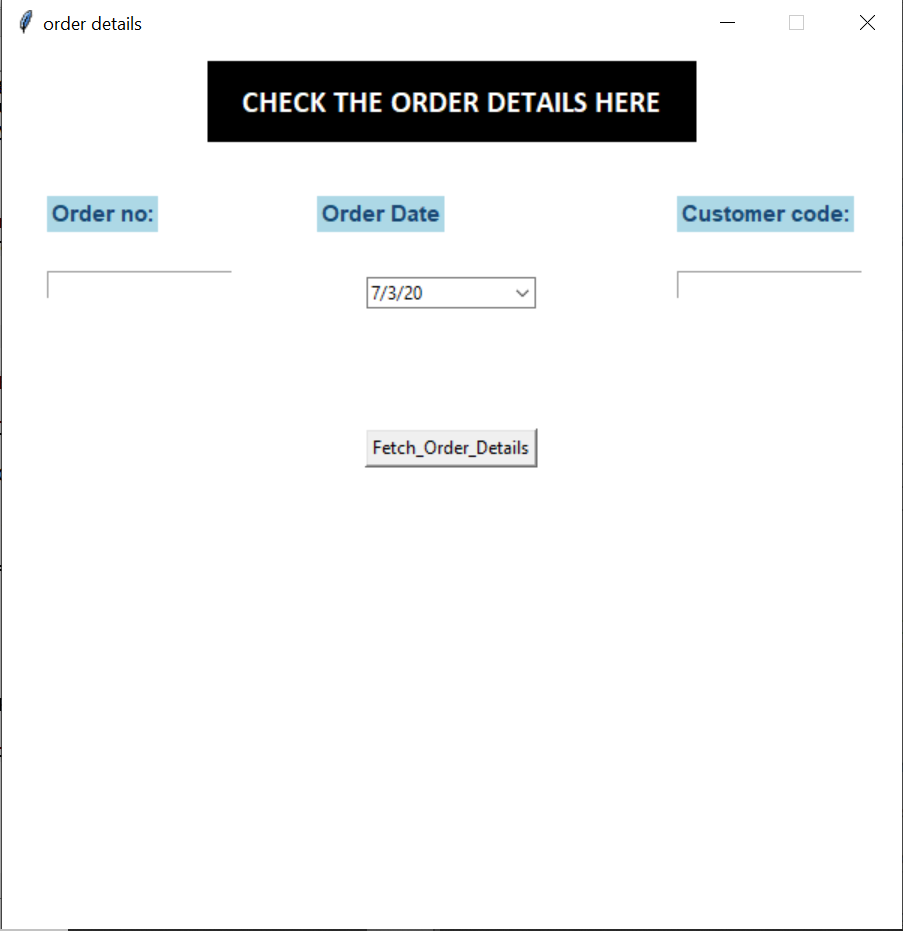


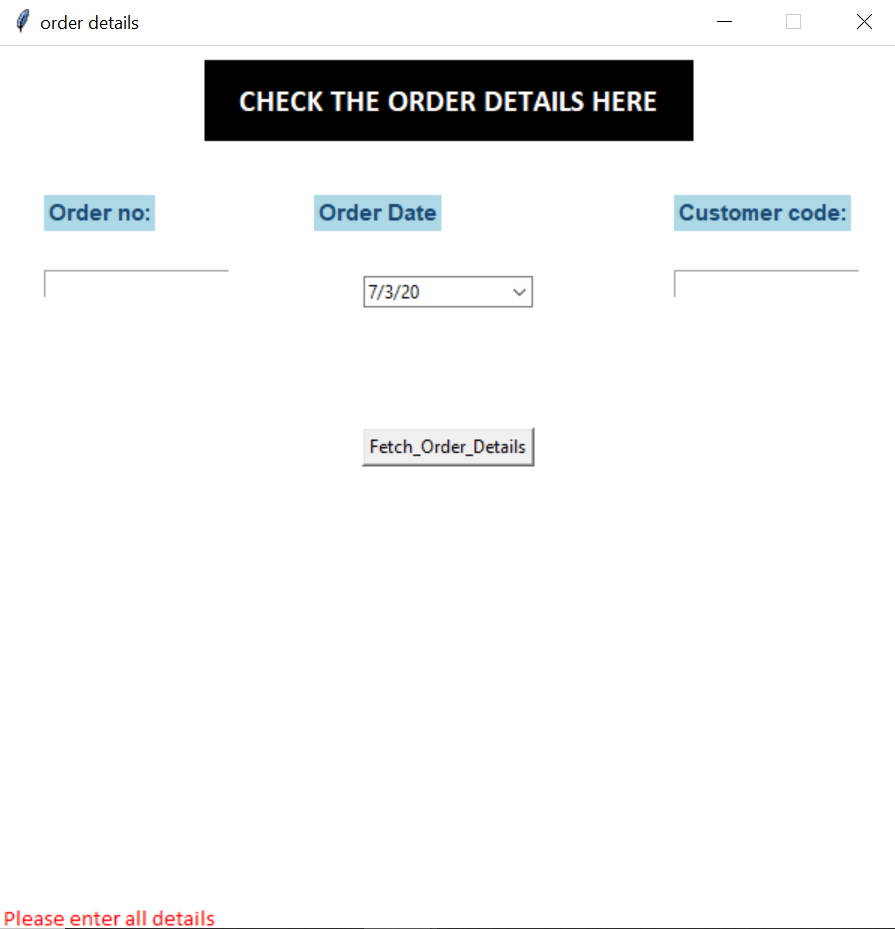
* Order Details:

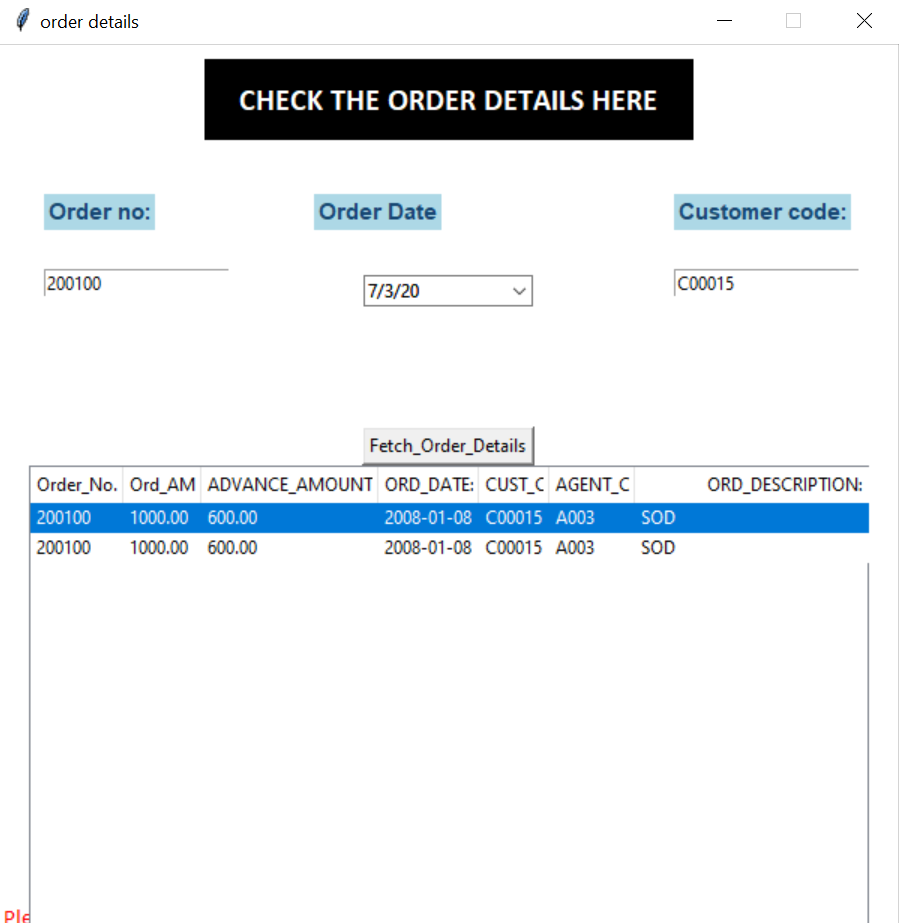




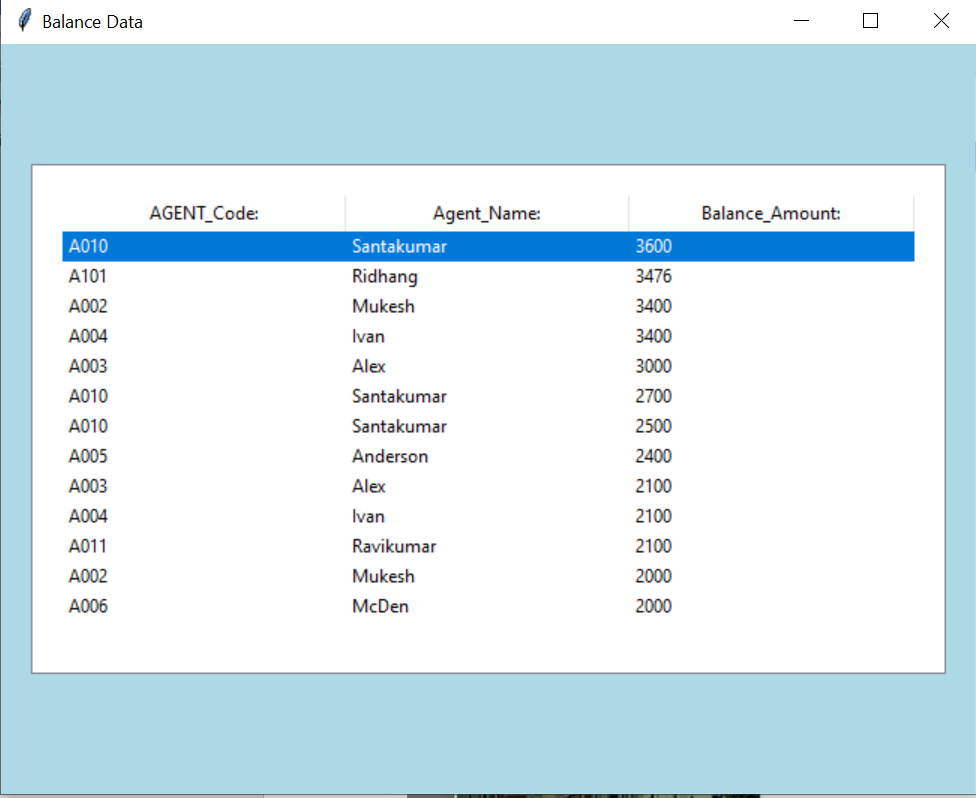
* Order Lookup:



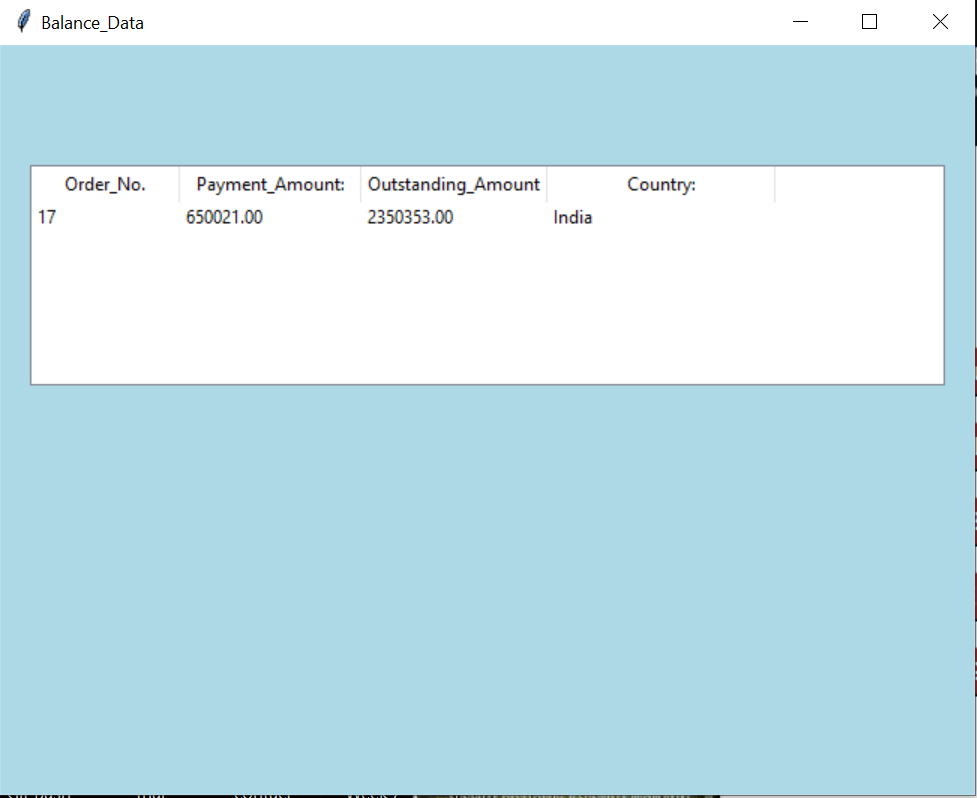




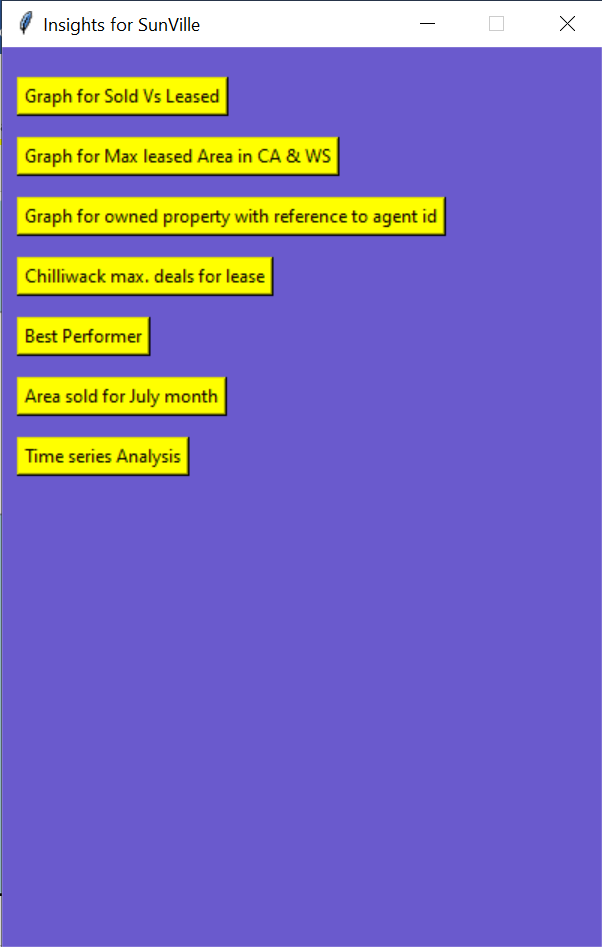
* Balance Amount

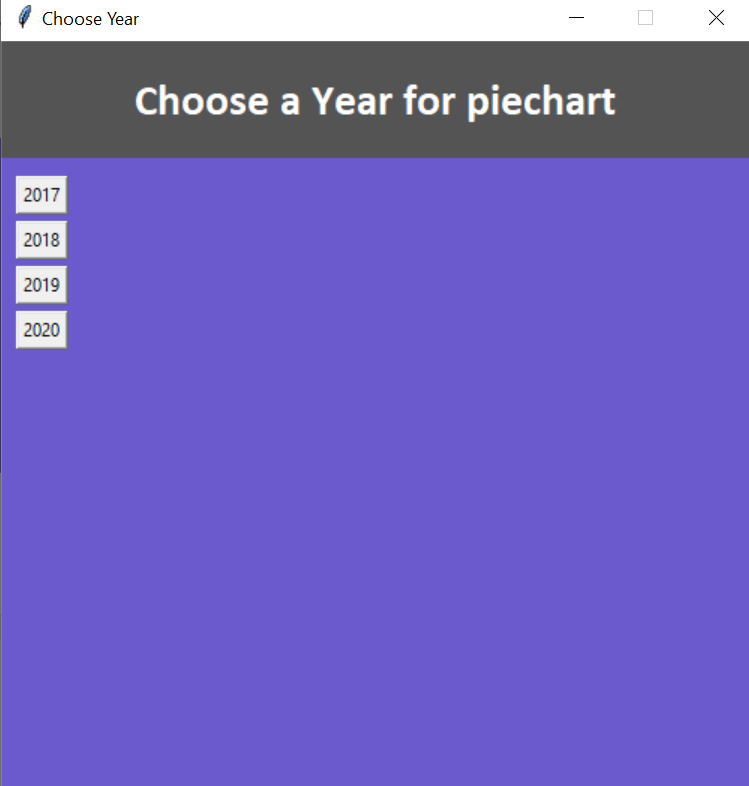


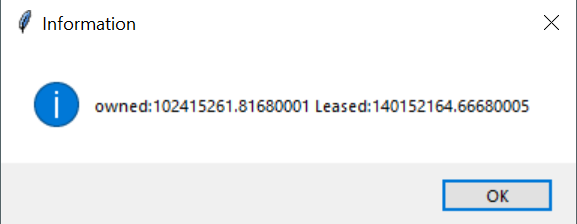
* Outstanding Amount

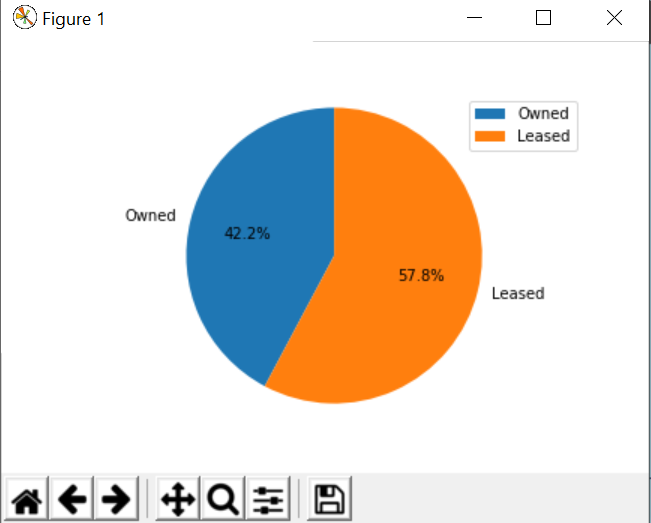


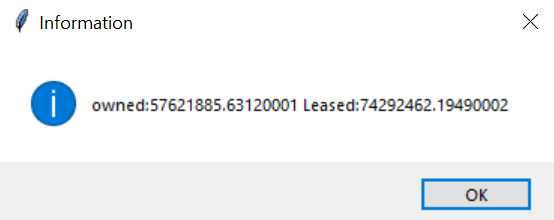
* Insights:

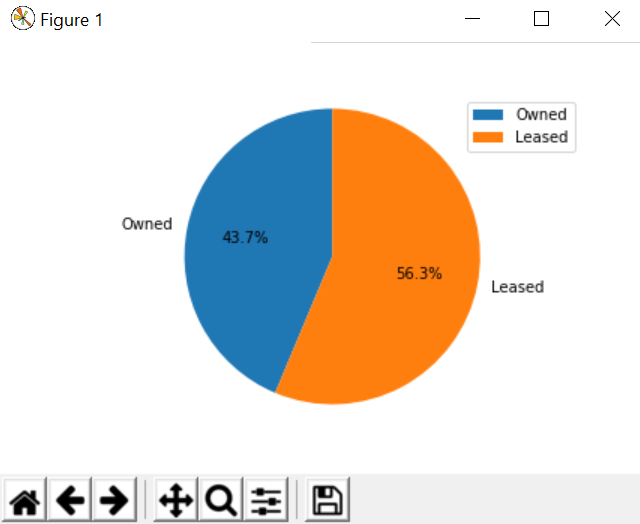


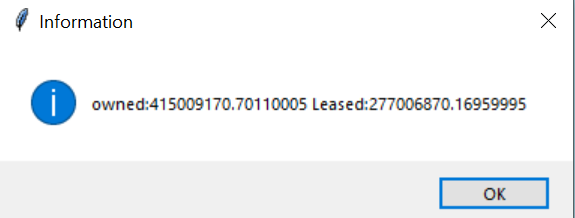


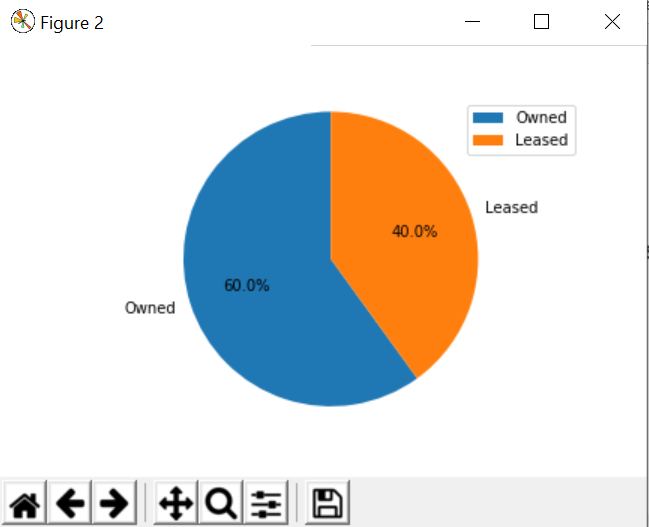


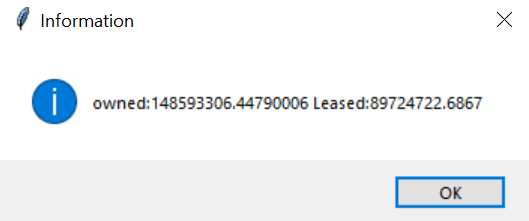


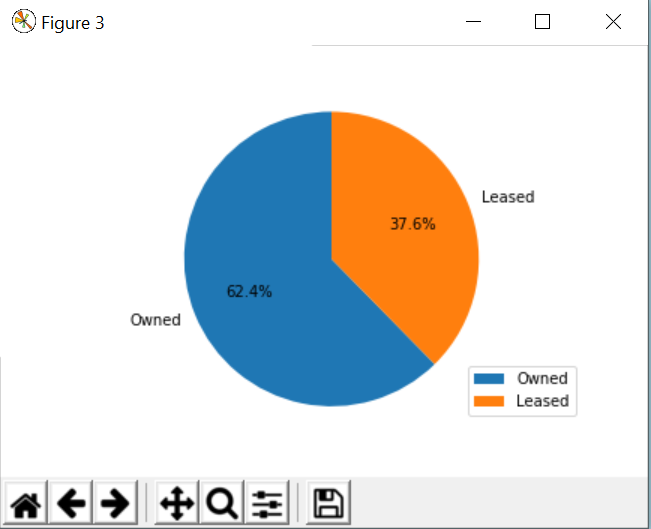


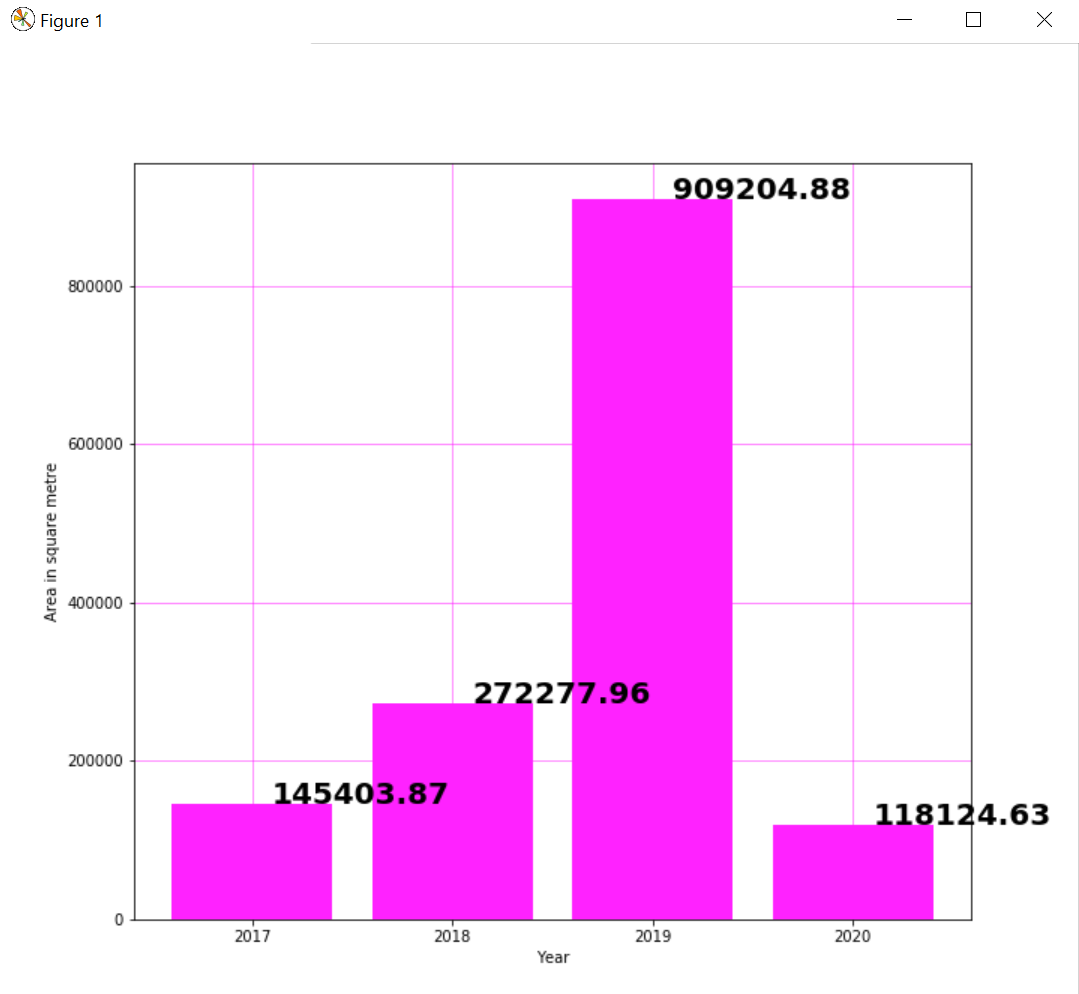


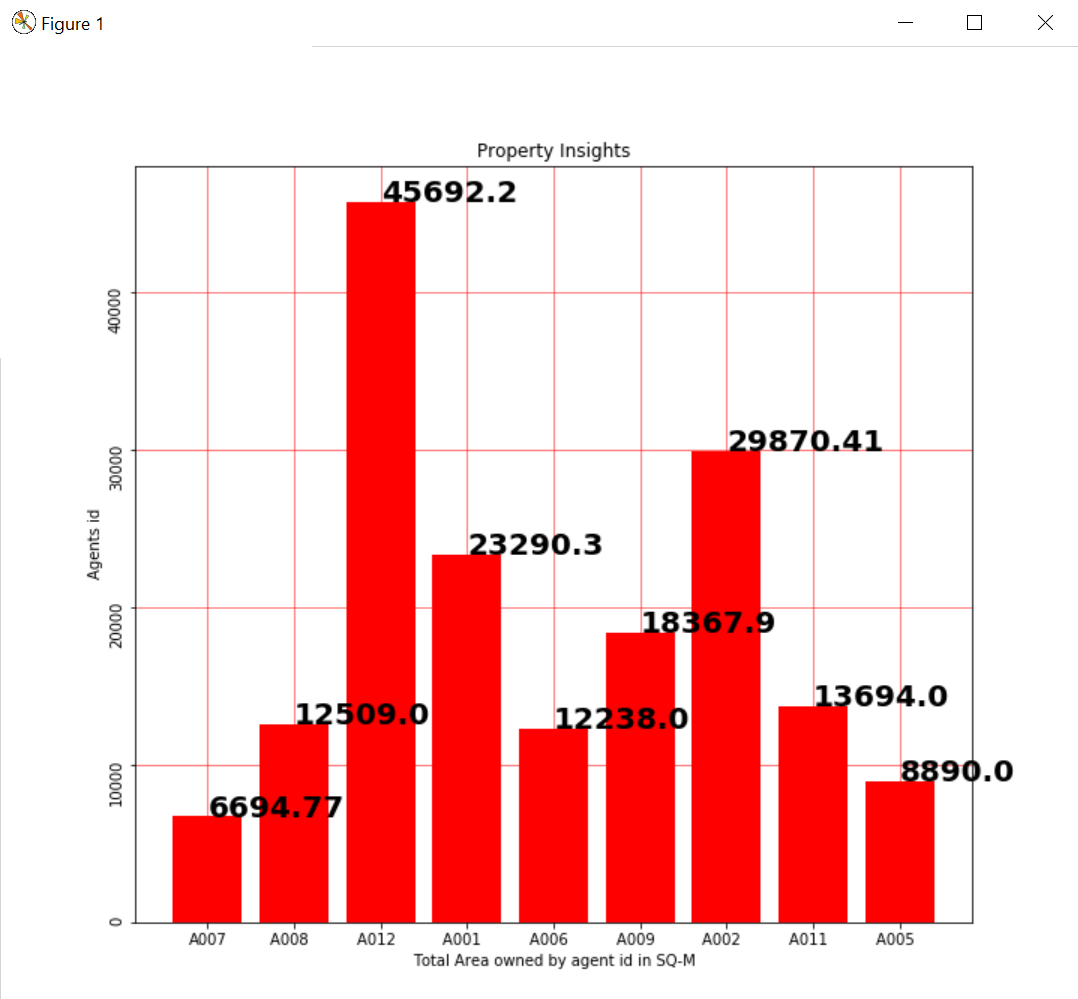


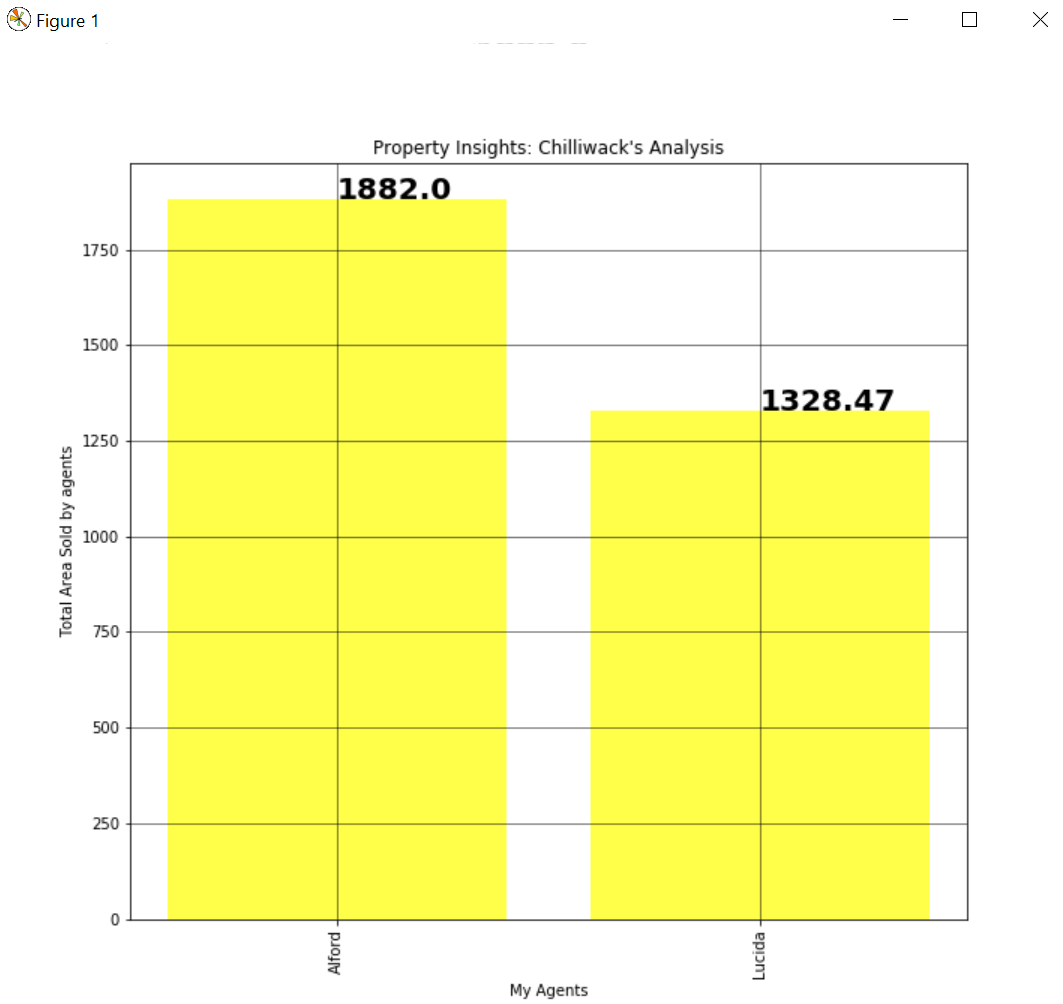


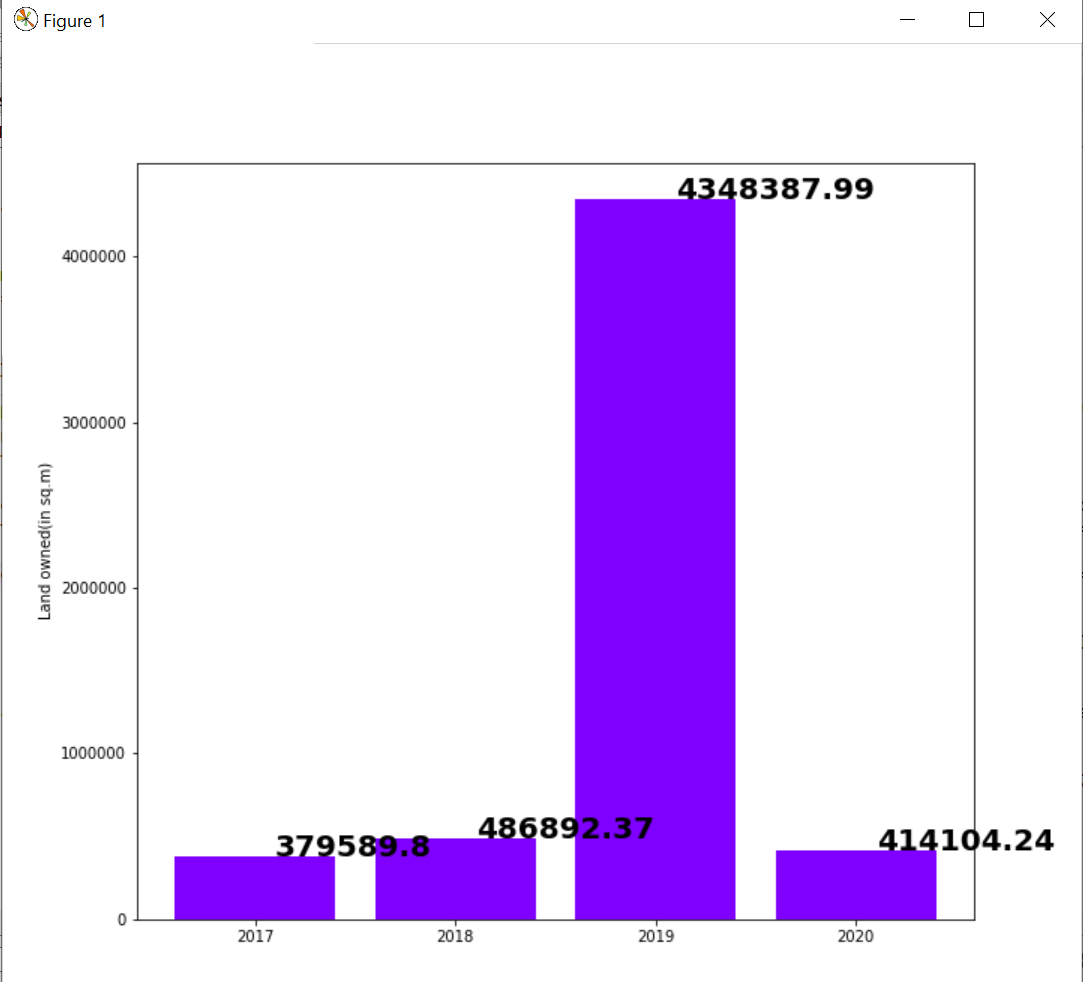


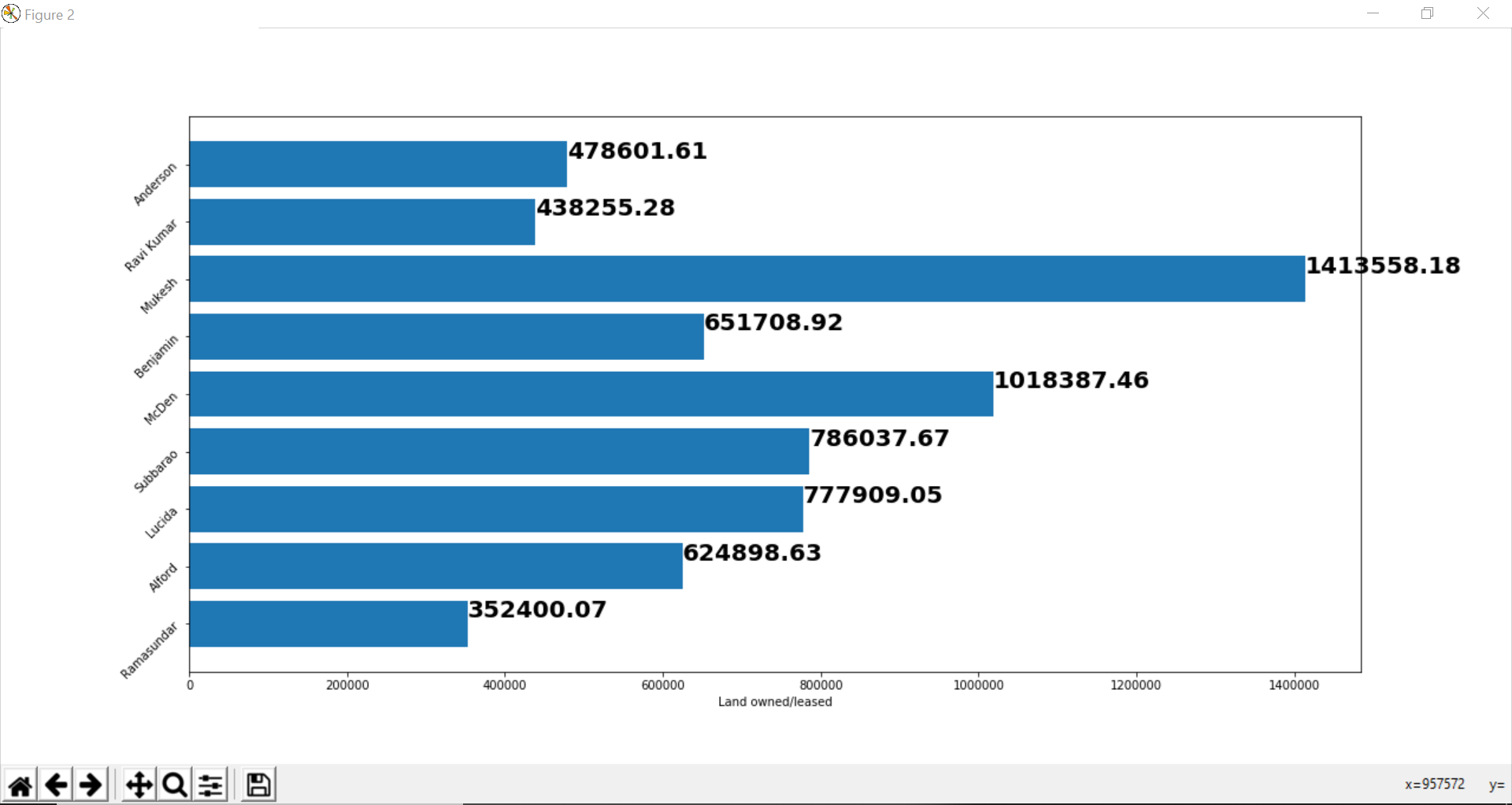












Section 5:

Testing:

* Testing

Graphical User-interface Testing or GUI testing is a process of testing the user interface of an application.

A graphical user interface includes all the elements such as menus, checkbox, buttons, colors, fonts, sizes, icons, content, and images. GUI testing is done to check the functionality and usability of design elements as a user for an application under test.

The five steps for GUI design are:

1. To create a Frame
2. To add a label
3. Add a button
4. To add button functionality using functions

If the above steps are successful then basic GUI is ready. Now one can background color or foreground color depending upon the requirements from the user.

Testing is majorly done to:

1. Check if all the GUI elements are visible w.r.t size, position and font type.
2. Check if font type used is readable.
3. Check if the images have good quality and properly aligned.
4. Check if labels are placed properly.
5. Check the positioning of GUI elements for different screen resolution.
6. Check if the data validation is taking place properly. For e.g. Entering numbers for agent name should show an error message either by labels or messageboxes.
7. Check if the button click event works for intended functions
8. Show message that user entered correct data for the respective entry field
9. Check if the database connected while execution of program.

10) Check for Clear demarcation of different sections on screen.

11) Check Error Messages are displayed correctly.

12) To check if value instances is getting updated or not while connection with database.

13) Check if no exception is generated during run-time.

Validation is an automated check, performed to guarantee that the data input is rational and acceptable.

The purpose of validation is to make sure any given set of data is logical, rational, complete, and within acceptable limits.

The types of validation used in the project are type check, check digit, length check, format check and presence check.

Type Check: - this is a way to confirm that the correct data type is inputted. For e.g. Writing letters for phone number in an entry field or writing numbers in agent name/customer name/country name field.

Check digit: - This is used to find out if a series of numbers has been keyed correctly. For e.g. Entering Company ID and adding a method isdigit () will check the instances whether it is a digit or not and returns a Boolean expression.

Length Check:- This is used to make sure that the correct number of [characters](https://teachcomputerscience.com/characters/) are entered into the field. It confirms that the character string entered is neither too short nor too long. For e.g. Phone numbers have a 10-digit numbers and if user is not entering 10-digit numbers so it shows an error message as invalid phone number.

Format Check: - This checks that the input data is in the right format. For e.g. The agent ID starts from A with 3-digit numbers and if user enters it in another format hence it shows an error.

Presence check: - this kind of check makes sure that an essential or required field cannot be left blank: it must be filled in. If someone attempts to leave the field blank, then an error message will be displayed, and they won’t be able to proceed to the next step, nor will they be able to save any other data which they have entered.

Section 6:

Source Code:

* Source Code

from tkinter import \*

from tkinter import messagebox

from tkinter import ttk

import tkinter as tk

import re, pymysql

from PIL import ImageTk, Image

import os

import matplotlib

matplotlib.use("TkAgg")

import matplotlib.pyplot as plt

from matplotlib.figure import Figure

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

from tkcalendar import \*

import pandas as pd

import numpy as np

import mysql.connector

import pandas.io.sql as sql

from pymysql import connect

#from mysql.connector import cursor

from mysql import \*

def adjustWindow(window):

w = 600 # width for the window size

h = 600 # height for the window size

ws = screen.winfo\_screenwidth()

hs = screen.winfo\_screenheight()

x = (ws/2) - (w/2) # calculate x and y coordinates for the Tk window

y = (hs/2) - (h/2)

window.geometry('%dx%d+%d+%d' % (w, h, x, y)) # set the dimensions of the screenand where it is placed

window.resizable(False, False)

window.configure(background='white')

#VALIDATION AND DATABASE CONNECTION FOR THE ORDER DETAILS FORM

def verify\_orderdetails():

if ord\_no.get() and ord\_amt.get() and adv\_amt.get() and cust\_code.get() and agt\_code.get() and ord\_des.get() and ent.get() and bal\_amt.get() and agt\_name.get():

if all(x.isalpha() or x.isspace() for x in agt\_name.get()) and (len(agt\_name.get())>0):

if re.match("[\d]{4}-[\d]{1,2}-[\d]{1,2}",ent.get()):

if len(ord\_no.get())==6 and re.match("[0-9][0-9][0-9][0-9][0-9][0-9]",ord\_no.get()):

if re.match("[0-9][0-9][0-9]",ord\_amt.get()):

if re.match("[C]+[0-9][0-9][0-9][0-9][0-9]",cust\_code.get()) and len(cust\_code.get())==6:

if re.match("[0-9][0-9][0-9]",adv\_amt.get()):

if re.match("[A]+[0-9][0-9][0-9]",agt\_code.get()) and len(agt\_code.get())==4:

if re.match("[S]+[O]+[D]",ord\_des.get()):

if re.match("[0-9][0-9][0-9]",bal\_amt.get()):

connection = pymysql.connect(host="localhost", user="root", passwd="", database="sales")

cursor = connection.cursor()

insert\_query ="INSERT INTO orders(ORD\_NUM,ORD\_AMOUNT,ADVANCE\_AMOUNT,ORD\_DATE,CUST\_CODE,AGENT\_CODE,ORD\_DESCRIPTION,Agent\_Name,Balance\_Amount) VALUES('"+ ord\_no.get() + "', '"+ ord\_amt.get() + "', '"+ adv\_amt.get() +"', '"+ ent.get() + "', '"+ cust\_code.get() + "', '"+ agt\_code.get() + "', '"+ ord\_des.get() + "', '"+ agt\_name.get() + "', '"+ bal\_amt.get() + "' );"

cursor.execute(insert\_query)

connection.commit()

connection.close()

print("DONE")

Label(screen5, text="DETAILS FILLED SUCCESFULLY", fg="green", font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

Button(screen5, text='DONE', width=20, font=("Open Sans", 9,'bold'), bg='brown', fg='white',command=screen5.destroy).place(x=470, y=565)

else:

Label(screen5, text="Please enter the valid Balance amount", fg="red",font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

else:

Label(screen5, text="Please enter the valid Order Description", fg="red",font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

else:

Label(screen5, text="Please enter the valid Agent Code", fg="red",font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

else:

Label(screen5, text="Please enter the valid Advance amount", fg="red",font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

else:

Label(screen5, text="Please enter the valid Customer code", fg="red",font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

else:

Label(screen5, text="Please enter the valid order amount", fg="red",font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

else:

Label(screen5, text="Please enter the valid order no", fg="red",font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

else:

Label(screen5, text="Please enter the valid date", fg="red",font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

return

else:

Label(screen5, text="Please Enter Valid Agent Name", fg="red",font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

return

else:

Label(screen5, text="Please Fill all the details", fg="red",font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

#GUI FOR THE ORDER DETAILS FORM

def add\_orderdetails():

global screen5, ord\_no, ord\_amt, adv\_amt, ord\_date, cust\_code,agt\_code,ord\_des,ent,agt\_name,bal\_amt

screen5=Toplevel(screen)

screen5.title("ADD THE ORDERS DETAILS")

adjustWindow(screen5)

ord\_no=StringVar()

ord\_amt=StringVar()

adv\_amt=StringVar()

ord\_date=StringVar()

cust\_code=StringVar()

agt\_code=StringVar()

ord\_des=StringVar()

agt\_name=StringVar()

bal\_amt=StringVar()

screen5.configure(bg="light blue")

Label(screen5, text="Fill the details below", width='32', height="2", font=("Calibri", 22, 'bold'), fg='white', bg="black").place(x=50,y=0)

Label(screen5, text="Order no:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=100)

Entry (screen5, textvar=ord\_no).place(x=250, y=100)

Label (screen5, text="Order Amount", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=140)

Entry (screen5, textvar=ord\_amt).place(x=250, y=140)

Label(screen5, text="Advance Amount", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=180)

Entry(screen5, textvar=adv\_amt).place(x=250, y=180)

Label(screen5, text="Order Date", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=220)

ent=DateEntry(screen5,width=15,bg="blue",fg="red",borderwidth=3,date\_pattern='yyyy-mm-dd')

ent.place(x=260,y=220)

Label(screen5, text="Customer Code", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=260)

Entry(screen5, textvar=cust\_code).place(x=250, y=260)

Label(screen5, text="Agent code", font=("Open Sans", 11, 'bold'), fg='white',

bg='#174873', anchor=W).place(x=90, y=300)

Entry(screen5, textvar=agt\_code).place(x=250, y=300)

Label(screen5, text="Order Description", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=340)

Entry(screen5, textvar=ord\_des).place(x=250, y=340)

Label(screen5, text="Agent Name", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=380)

Entry(screen5, textvar=agt\_name).place(x=250, y=380)

Label(screen5, text="Balance Amount", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=420)

Entry(screen5, textvar=bal\_amt).place(x=250, y=420)

Button(screen5, text='Submit', width=20, font=("Open Sans", 13, 'bold'), bg='green', fg='white',command=verify\_orderdetails).place(x=130, y=480)

def verify\_customer():

CUST\_CODE =cust\_code1.get()

CUST\_NAME = cust\_name.get()

CUST\_CITY = cust\_city.get()

WORKING\_AREA = work\_area.get()

CUST\_COUNTRY = cust\_country.get()

GRADE = grade.get()

OPENING\_AMT = op\_amt.get()

RECEIVE\_AMT = rcv\_amt.get()

PAYMENT\_AMT = pay\_amt.get()

OUTSTANDING\_AMT = out\_amt.get()

PHONE\_NO = phone.get()

AGENT\_CODE= agt\_code1.get()

if cust\_code1.get() and cust\_name.get() and cust\_city.get() and work\_area.get() and cust\_country.get() and op\_amt.get() and rcv\_amt.get() and pay\_amt.get() and out\_amt.get() and phone.get() and agt\_code1.get() and grade.get():

if re.match("[C]+[0-9][0-9][0-9][0-9][0-9]",cust\_code1.get()) and len(cust\_code1.get())==6:

if all(x.isalpha() or x.isspace() for x in cust\_name.get()) and (len(cust\_name.get())>0):

if all(y.isalpha() or y.isspace() for y in cust\_city.get()) and (len(cust\_city.get())>0):

if all(z.isalpha() or z.isspace() for z in work\_area.get()) and (len(work\_area.get())>0):

if all(w.isalpha() or w.isspace() for w in cust\_country.get()) and (len(cust\_country.get())>0):

if len(op\_amt.get())==4 and op\_amt.get().isdigit():

if len(rcv\_amt.get())==4 and rcv\_amt.get().isdigit():

if len(pay\_amt.get())==4 and pay\_amt.get().isdigit():

if len(out\_amt.get())==4 and out\_amt.get().isdigit():

if len(phone.get())==10 and phone.get().isdigit():

if re.match("[A]+[0-9][0-9][0-9]",agt\_code1.get()) and len(agt\_code1.get())==4:

if len(grade.get())==1 and grade.get().isdigit():

mydb = mysql.connector.connect(host="localhost",user="root",password="",database="sales")

cursor = mydb.cursor()

sql="INSERT INTO customer VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"

cursor.execute(sql,(CUST\_CODE,CUST\_NAME,CUST\_CITY,WORKING\_AREA,CUST\_COUNTRY,GRADE,OPENING\_AMT, RECEIVE\_AMT,PAYMENT\_AMT,OUTSTANDING\_AMT,PHONE\_NO,AGENT\_CODE))

mydb.commit()

print("Customer Added")

messagebox.showinfo("Success","Customer added Successfully")

#return True

else:

messagebox.showerror("Error","Invalid Grade.\nHint:- The grade is a single digit number")

else:

messagebox.showerror("Error","Invalid Agent Code")

else:

messagebox.showerror("Error","Invalid Phone Number")

else:

messagebox.showerror("Error","Invalid outstanding amount")

else:

messagebox.showerror("Error","Invalid payment amount")

else:

messagebox.showerror("Error","Invalid Receive amount")

else:

messagebox.showerror("Error","Invalid Opening amount")

else:

messagebox.showerror("Error","Invalid Customer's Country")

else:

messagebox.showerror("Error","Invalid Working area")

else:

messagebox.showerror("Error","Invalid Customer's City")

else:

messagebox.showerror("Error","Invalid Customer Name \nMake Sure it's NOT a number")

else:

messagebox.showerror("Error","Invalid Customer Code \nHint:The Customer Code starts from 'C' with five digit number")

else:

messagebox.showerror("Error","Please Enter all details")

def register\_customer():

global cust\_code1,cust\_name,cust\_city,work\_area,cust\_country,grade,op\_amt,rcv\_amt,pay\_amt,out\_amt,phone,agt\_code1,work\_area,screen2

screen2=Toplevel(screen)

screen2.title("ADD THE CUSTOMER DETAILS")

adjustWindow(screen2)

cust\_code1=StringVar()

cust\_name=StringVar()

cust\_city=StringVar()

work\_area =StringVar()

cust\_country=StringVar()

grade=StringVar()

op\_amt=StringVar()

rcv\_amt=StringVar()

pay\_amt=StringVar()

out\_amt=StringVar()

phone=StringVar()

agt\_code1=StringVar()

screen2.configure(bg="light blue")

Label(screen2, text="Fill the details below", width='32', height="2", font=("Calibri", 22, 'bold'), fg='white', bg="black").place(x=50,y=0)

Label(screen2, text="CUSTOMER CODE:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=100)

Entry(screen2, textvar=cust\_code1).place(x=280, y=100)

Label(screen2, text="CUSTOMER NAME:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=140)

Entry(screen2, textvar=cust\_name).place(x=280, y=140)

Label(screen2, text="CUSTOMER CITY:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=180)

Entry(screen2, textvar=cust\_city).place(x=280, y=180)

Label(screen2, text="WORKING AREA:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=220)

Entry(screen2, textvar=work\_area).place(x=280, y=220)

Label(screen2, text="CUSTOMER COUNTRY:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=260)

Entry(screen2, textvar=cust\_country).place(x=280, y=260)

Label(screen2, text="GRADE:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=300)

Entry(screen2, textvar=grade).place(x=280, y=300)

Label(screen2, text="OPENING AMOUNT:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=340)

Entry(screen2, textvar=op\_amt).place(x=280, y=340)

Label(screen2, text="RECIVING AMOUNT:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=380)

Entry(screen2, textvar=rcv\_amt).place(x=280, y=380)

Label(screen2, text="PAYMENT Amount:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=420)

Entry(screen2, textvar=pay\_amt).place(x=280, y=420)

Label(screen2, text="OUTSTANDING Amount:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=460)

Entry(screen2, textvar=out\_amt).place(x=280, y=460)

Label(screen2, text="PHONE NO:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=500)

Entry(screen2, textvar=phone).place(x=280, y=500)

Label(screen2, text="AGENT CODE:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=540)

Entry(screen2, textvar=agt\_code1).place(x=280, y=540)

Button(screen2, text='Submit', width=20, font=("Open Sans", 13, 'bold'), bg='green', fg='white',command=verify\_customer).place(x=375, y=564)

def verify\_company():

COMPANY\_ID=companyid.get()

COMPANY\_NAME = companyname.get()

COMPANY\_CITY= companycity.get()

if companyid.get() and companyname.get() and companycity.get():

if len(companyid.get())==2 and companyid.get().isdigit():

if all(x.isalpha() or x.isspace() for x in companyname.get()) and (len(companyname.get())>0):

if all(y.isalpha() or y.isspace() for y in companycity.get()) and (len(companycity.get())>0):

mydb = mysql.connector.connect(host="localhost",user="root",password="",database="sales")

cursor = mydb.cursor()

sql="INSERT INTO company VALUES(%s, %s, %s)"

cursor.execute(sql,(COMPANY\_ID, COMPANY\_NAME ,COMPANY\_CITY ))

mydb.commit()

#mydb.close()

#print("Company Added")

#messagebox.showinfo("Success","Company Added")

Label(screen7, text="Company Added", fg="green",font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

#return True

else:

Label(screen7, text="Invalid Company City", fg="red",font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

return

else:

Label(screen7, text="Invalid Company Name", fg="red",font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

return

else:

Label(screen7, text="Invalid Company ID", fg="red",font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

return

else:

Label(screen7, text="Please fill all details", fg="red",font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

return

def add\_company():

global companyid,companyname,companycity,screen7

companyid=StringVar()

companyname=StringVar()

companycity=StringVar()

screen7 = Toplevel(screen)

screen7.title("Company Registeration Form ")

adjustWindow(screen7)

screen7.configure(bg='light blue')

Label(screen7, text="Company Registration Form", width='32', height="2", font=("Calibri", 22, 'bold'), fg='white', bg="black").pack()

Label(screen7, text="Company ID:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=160)

Entry(screen7, textvar=companyid).place(x=260, y=160)

Label(screen7, text="Company Name:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=210)

Entry(screen7, textvar=companyname).place(x=260, y=210)

Label(screen7, text="Company City:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=260)

Entry(screen7, textvar=companycity).place(x=260, y=260)

Button(screen7, text='Submit', width=20, font=("Open Sans", 13, 'bold'), bg='red', fg='white',command=verify\_company).place(x=130, y=470)

def enter\_details():

global screen4

screen4=Toplevel(screen)

screen4.title("Fill the Details")

adjustWindow(screen4)

screen4.configure(bg="grey")

Button(screen4, text='Add Agent Details Here', width=20, font=("Open Sans", 13, 'bold'), bg='#174873', fg='white',command=register\_agent).place(x=200, y=175)

Button(screen4, text='Add Company Details Here', width=30, font=("Open Sans", 13, 'bold'), bg='#174873', fg='white',command=add\_company).place(x=150, y=225)

Button(screen4, text='Add Customer Details Here', width=30, font=("Open Sans", 13, 'bold'), bg='#174873', fg='white',command=register\_customer).place(x=150, y=275)

Button(screen4, text='Add Order Details Here', width=20, font=("Open Sans", 13, 'bold'), bg='#174873', fg='white',command=add\_orderdetails).place(x=200, y=325)

def verify\_register\_agent():

AGENT\_CODE =agentcode.get()

AGENT\_NAME = agentname.get()

WORKING\_AREA = workingarea.get()

try:

COMMISSION = commission.get()

except:

messagebox.showwarning("Warning","Invalid Commision")

PHONE\_NO = phoneno.get()

COUNTRY = country.get()

if agentcode.get() and agentname.get() and workingarea.get() and phoneno.get() and country.get() and commission.get():

if re.match("[A]+[0-9][0-9][0-9]",agentcode.get()) and len(agentcode.get())==4:

if all(x.isalpha() or x.isspace() for x in agentname.get()) and (len(agentname.get())>0):

if all(y.isalpha() or y.isspace() for y in workingarea.get()) and (len(workingarea.get())>0):

if commission.get():

if len(phoneno.get())==10 and phoneno.get().isdigit():

if all(z.isalpha() or z.isspace() for z in country.get()) and (len(country.get())>0):

mydb = mysql.connector.connect(host="localhost",user="root",password="",database="sales")

cursor = mydb.cursor()

sql="INSERT INTO agents VALUES(%s, %s, %s, %s, %s, %s)"

cursor.execute(sql,(AGENT\_CODE, AGENT\_NAME , WORKING\_AREA , COMMISSION ,PHONE\_NO , COUNTRY))

mydb.commit()

mydb.close()

print("Agent Added")

messagebox.showinfo("Success","Agent Added Succesfully")

#return True

else:

messagebox.showerror("Error","Invalid Country")

else:

messagebox.showerror("Error","Invalid phone number")

else:

try:

messagebox.showerror("Error","Invalid Commision")

except \_tkinter.TclError:

messagebox.showwarning("Alert","Commision cannot accept other than numbers")

else:

messagebox.showerror("Error","Invalid Working area")

else:

messagebox.showerror("Error","Invalid Agent name")

else:

messagebox.showerror("Error","Invalid agent code")

else:

messagebox.showerror("Error","Please fill all details")

def register\_agent():

global agentname, agentcode,workingarea,commission, phoneno, country,screen3 # making all entry field variable global

screen3=Toplevel(screen)

screen3.title("Agent Registeration Form ")

adjustWindow(screen3)

agentname = StringVar()

workingarea = StringVar()

commission=DoubleVar()

phoneno= StringVar()

country = StringVar()

agentcode = StringVar()

screen3.configure(bg='light blue')

Label(screen3, text="Agent Registration Form", width='32', height="2", font=("Calibri", 22, 'bold'), fg='white', bg="black").pack()

Label(screen3, text="Agent Code:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=160)

Entry(screen3, textvar=agentcode).place(x=260, y=160)

Label(screen3, text="Agent Name:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=210)

Entry(screen3, textvar=agentname).place(x=260, y=210)

Label(screen3, text="Working Area:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=260)

Entry(screen3, textvar=workingarea).place(x=260, y=260)

Label(screen3, text="Commission:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=310)

Entry(screen3,textvar=commission).place(x=260, y=305)

Label(screen3, text="Phone\_No.:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=360)

Entry(screen3, textvar=phoneno).place(x=260, y=360)

Label(screen3, text="Country:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=410)

entry\_4 = Entry(screen3, textvar=country)

entry\_4.place(x=260, y=410)

Button(screen3, text='Submit', width=20, font=("Open Sans", 13, 'bold'), bg='red', fg='white',command=verify\_register\_agent).place(x=130, y=470)

def Balance\_Amount():

conn= mysql.connector.connect(host="localhost",user="root",password="",database="sales")

c = conn.cursor()

c.execute("SELECT DISTINCT AGENT\_Code,Agent\_Name,Balance\_Amount from orders ORDER BY Balance\_Amount DESC")

rows = c.fetchall()

total = c.rowcount

print("Total Data Entries:"+str(total))

win=Tk()

frm = Frame(win)

frm.pack(side=tk.TOP,padx=20,pady=80)

tv = ttk.Treeview(frm,columns=(1,2,3),show="headings",height="30",padding=20)

tv.pack()

style = ttk.Style()

style.configure("Treeview.Heading", font=("Italic", 15),fg='#174873')

style.configure("Treeview.Insert",fg='#174873',font=("Italic", 15))

win.configure(background='light blue')

tv.heading(1,text="AGENT\_Code:")

tv.heading(2,text="Agent\_Name:")

tv.heading(3,text="Balance\_Amount:")

for i in rows:

tv.insert('','end',values=i)

win.title("Balance Data")

win.geometry("650x500")

win.mainloop()

def Outstanding\_Amount():

conn= mysql.connector.connect(host="localhost",user="root",password="",database="sales")

c = conn.cursor()

c.execute("select \* from (select count(cust\_code) as cust\_ct, sum(payment\_amt), sum(outstanding\_amt), cust\_country from customer group by cust\_country) as Q where cust\_ct in (select max(cust) from (select count(cust\_code) as cust, cust\_country from customer group by cust\_country) AS T);")

rows = c.fetchall()

total = c.rowcount

print("Total Data Entries:"+str(total))

win=Tk()

frm = Frame(win)

frm.pack(side=tk.TOP,padx=20,pady=80)

tv = ttk.Treeview(frm,columns=(1,2,3,4),show="headings",height="06")

tv.pack()

style = ttk.Style()

style.configure("Treeview.Heading", font=("Italic", 15),fg='#174873')

win.configure(background='light blue')

tv.heading(1,text="Order\_No.")

tv.heading(2,text="Payment\_Amount:")

tv.heading(3,text="Outstanding\_Amount")

tv.heading(4,text="Country:")

for i in rows:

tv.insert('','end',values=i)

win.title("Balance\_Data")

win.geometry("650x500")

win.mainloop()

def fetch\_records():

if orderno.get() or cust\_code.get() or orderdate.get():

if len(orderno.get())==6 and orderno.get().isdigit():

if re.match("[C]+[0-9][0-9][0-9][0-9][0-9]",cust\_code.get()) and len(cust\_code.get())==6:

mydb= mysql.connector.connect(host="localhost",user="root",password="",database="sales")

cursor = mydb.cursor()

select\_query="SELECT ORD\_NUM,ORD\_AMOUNT,ADVANCE\_AMOUNT,ORD\_DATE,CUST\_CODE,AGENT\_CODE,ORD\_DESCRIPTION,Agent\_Name,Balance\_Amount FROM orders WHERE ORD\_NUM LIKE '%"+orderno.get()+"%' AND CUST\_CODE LIKE '%"+cust\_code.get()+"%'AND ORD\_DATE LIKE '%"+orderdate.get()+"%'"

cursor.execute(select\_query)

rows = cursor.fetchall()

mydb.commit()

mydb.close()

else:

Label(win, text="Please enter Valid Customer Code", fg="red",font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

return

else:

Label(win, text="Please enter valid order number", fg="red",font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

return

else:

Label(win, text="Please enter all details", fg="red",font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

return

total = cursor.rowcount

print("Total Data Entries:"+str(total))

frm = Frame(win)

frm.pack(side=tk.LEFT,padx=20)

tv = ttk.Treeview(frm,columns=(1,2,3,4,5,6,7,8,9),show="headings",height="30")

tv.pack()

tv.heading(1,text="Order\_No.")

tv.heading(2,text="Ord\_AMOUNT:")

tv.heading(3,text="ADVANCE\_AMOUNT:")

tv.heading(4,text="ORD\_DATE:")

tv.heading(5,text="CUST\_CODE:")

tv.heading(6,text="AGENT\_CODE:")

tv.heading(7,text="ORD\_DESCRIPTION:")

tv.heading(8,text="Agent\_Name:")

tv.heading(9,text="Balance\_Amount:")

for i in rows:

tv.insert('','end',values=i)

def Search\_Order():

#win=Tk()

global orderno,cust\_code,orderdate,win

orderno=StringVar()

cust\_code=StringVar()

orderdate = StringVar()

win=Toplevel(screen)

win.configure(bg="grey")

win.title("order details")

adjustWindow(win)

Label(win, text="CHECK THE ORDER DETAILS HERE", width='32', height="2", font=("Calibri", 15, 'bold'), fg='white', bg="black").pack(padx=20,pady=10)

Label(win, text="Order no:", font=("Open Sans", 11, 'bold'), fg='#174873', bg='light blue', anchor=W).place(x=30, y=100)

e1=Entry(win, textvar=orderno).place(x=30, y=150)

Label(win, text="Customer code:", font=("Open Sans", 11, 'bold'), fg='#174873', bg='light blue', anchor=W).place(x=450, y=100)

e2=Entry(win, textvar=cust\_code).place(x=450, y=150)

Label(win, text="Order Date", font=("Open Sans", 11, 'bold'), fg='#174873', bg='light blue', anchor=W).place(x=210, y=100)

ent = DateEntry(win,width=15,bg="blue",fg="red",borderwidth=3)

ent.pack(padx=180,pady=80)

check=Button(win,text="Fetch\_Order\_Details",command=fetch\_records).pack()

#win.title("Order\_Details")

#win.geometry("650x500")

#win.mainloop()

#remove\_orderno=Button(win,text="Discard Order No.").place(x=10,y=80)

def welcome\_page(agent\_info):

global screen2

screen2 = Toplevel(screen)

screen2.title("Welcome")

img = ImageTk.PhotoImage(Image.open("D:\\Python Internship\\sunville1.jpg"))

panel = Label(screen2,image = img)

panel.place(x=0,y=0)

panel.image=img

panel.pack()

adjustWindow(screen2)

Label(screen2, text="Welcome " + agent\_info[0][0], width='30', height="2", font=("Calibri", 15, 'bold'), fg='white', bg='#545454').place(x=0, y=150)

img1 = ImageTk.PhotoImage(Image.open("D:\\Python Internship\\new-cover.jpeg"))

panel1 = Label(screen2,image = img1)

panel1.place(x=0,y=445)

panel1.image=img1

Label(screen2, text="", bg='light blue', width='20', height='16').place(x=0, y=204)

Message(screen2, text='ABOUT US\n\n OUR TEAM \n\nBUY \n\n SELL \n\n RENT\n\n FAQ \n\n CONTACT US\n\n',width='180', font=("Helvetica", 10, 'bold', 'underline'), fg='#174873', bg='light blue', anchor = CENTER).place(x=0, y=204)

Button(screen2, text='Enter your details', width=20, font=("Open Sans", 13, 'bold'), bg='brown', fg='white',command=enter\_details).place(x=270, y=250)

Button(screen2, text='ORDER LOOKUP', width=20, font=("Open Sans", 13, 'bold'), bg='brown', fg='white',command=Search\_Order).place(x=270, y=300)

Button(screen2, text='REPORT OF BALANCE.AMT', width=30, font=("Open Sans", 13, 'bold'), bg='brown', fg='white',command=Balance\_Amount).place(x=220, y=350)

Button(screen2, text='MAX.REGISTER CUSTOMER', width=30, font=("Open Sans", 13, 'bold'), bg='brown', fg='white',command=Outstanding\_Amount).place(x=220, y=400)

Button(screen2, text='INSIGHTS', width=20, font=("Open Sans", 13, 'bold'), bg='brown', fg='white',command=insights).place(x=270, y=450)

def register\_user():

if fullname.get() and email.get() and password.get() and repassword.get() and gender.get():

if country.get() == "--select your country--":

Label(screen1, text="Please select your country", fg="red", font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

return

else:

if all(x.isalpha() or x.isspace() for x in fullname.get()) and (len(fullname.get())>0):

if re.match("^.+@(\[?)[a-zA-Z0-9-.]+.([a-zA-Z]{2,3}|[0-9]{1,3})(]?)$", email.get()):

if password.get() == repassword.get() and len(password.get())>=8 and len(repassword.get())>=8:

gender\_value = 'male'

if gender.get() == 2:

gender\_value = 'female'

connection = pymysql.connect(host="localhost", user="root", passwd="", database="sales")

cursor = connection.cursor()

insert\_query ="INSERT INTO agent\_details (fullname, email, password,gender, country) VALUES('"+ fullname.get() + "', '"+ email.get() + "', '"+ password.get() +"', '"+ gender\_value + "', '"+ country.get() + "' );"

cursor.execute(insert\_query)

connection.commit()

connection.close()

Label(screen1, text="Registration Sucess", fg="green", font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

Button(screen1, text='Proceed to Login ->', width=20, font=("Open Sans", 9,'bold'), bg='brown', fg='white',command=screen1.destroy).place(x=170, y=565)

else:

Label(screen1, text="Either Password does not match or Length of the password is less than than 8", fg="red", font=("calibri", 11), width='120', anchor=W, bg='white').place(x=0, y=570)

return

else:

Label(screen1, text="Please enter valid email id", fg="red", font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

return

else:

Label(screen1, text="Please enter valid Full name", fg="red", font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

return

else:

Label(screen1, text="Please fill all the details", fg="red",font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)

return

def login\_verify():

global agentID

connection = pymysql.connect(host="localhost", user="root", passwd="", database="sales") # database connection

cursor = connection.cursor()

select\_query = "SELECT \* FROM agent\_details where email = '" + username\_verify.get() + "' AND password = '" + password\_verify.get() + "';" # queries for retrieving values

cursor.execute(select\_query) # executing the queries

agent\_info = cursor.fetchall()

connection.commit() # commiting the connection then closing it.

connection.close() # closing the connection of the database

if agent\_info:

messagebox.showinfo("Congratulation", "Login Successfull") # displaying message for successful login

agentID = agent\_info[0][0]

welcome\_page(agent\_info) # opening welcome window

else:

messagebox.showerror("Error", "Invalid Username or Password") # displaying message for invalid

def register():

global screen1, fullname, email, password, repassword, country, gender, tnc # making all entry field variable global

fullname = StringVar()

email = StringVar()

password = StringVar()

repassword = StringVar()

country = StringVar()

gender = IntVar()

tnc = IntVar()

screen1 = Toplevel(screen)

adjustWindow(screen1)

screen1.title("Registeration")

screen1.configure(bg='light blue')

Label(screen1, text="Registration Form", width='32', height="2", font=("Calibri", 22, 'bold'), fg='white', bg="black").pack()

Label(screen1, text="Full Name:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=160)

Entry(screen1, textvar=fullname).place(x=260, y=160)

Label(screen1, text="Email ID:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=210)

Entry(screen1, textvar=email).place(x=260, y=210)

Label(screen1, text="Gender:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=260)

Radiobutton(screen1, text="Male", variable=gender, value=1, bg='#174873').place(x=260, y=260)

Radiobutton(screen1, text="Female", variable=gender, value=2, bg='#174873').place(x=330, y=260)

Label(screen1, text="Country:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=310)

list1 = ['INDIA', 'USA','UK', 'AUSTRALIA']

droplist = OptionMenu(screen1,country, \*list1)

droplist.config(width=17)

country.set('--select your country--')

droplist.place(x=260, y=305)

Label(screen1, text="Password:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=360)

Entry(screen1, textvar=password, show="\*").place(x=260, y=360)

Label(screen1, text="Re-Password:", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873', anchor=W).place(x=90, y=410)

entry\_4 = Entry(screen1, textvar=repassword, show="\*")

entry\_4.place(x=260, y=410)

Button(screen1, text='Submit', width=20, font=("Open Sans", 13, 'bold'), bg='red', fg='white',command=register\_user).place(x=130, y=470)

def insights():

#root=Tk()

root = Toplevel(screen)

root.title("Insights for SunVille")

adjustWindow(root)

root.geometry("400x600")

root.configure(background='Slate blue')

def graph1():

def soldvleased(year):

leased=0

owned=0

test = pd.read\_excel("D:\Python Internship\Dataset.xlsx",header=8)

for row in test.itertuples(index=False):

if row[0]==year:

temp=row[7]\*1000

else:

temp=row[7]

if row[9]=='Leased':

leased=leased+temp

elif row[9]=='Owned':

owned=owned+temp

print("THE LAND OWNED IN SQM ",int(owned),"sqm")

print("THE LAND LEASED IN SQM ",int(leased),"sqm")

labels = 'Owned', 'Leased'

sizes = [owned, leased]

#sizes = [owned, leased]

messagebox.showinfo("Information","owned:"+str(sizes[0])+" Leased:"+str(sizes[1]))

fig1, ax1 = plt.subplots()

ax1.pie(sizes,labels=labels, startangle=90,autopct="%.1f%%")

ax1.axis('equal')

plt.show()

def maxleasedarea():

test = pd.read\_excel("D:\Python Internship\Dataset.xlsx",header=8)

di={"2017":0,"2018":0,"2019":0,"2020":0}

for row in test.itertuples(index=False):

if row[8]=='HA':

temp=row[7]\*10000

else:

temp=row[7]

if (row[5]=='CA' or row[5]=='WS') and row[9]=='Leased':

di[str(row[0])]=di[str(row[0])]+temp

Keymax = max(di, key=di.get)

print(Keymax)

print("A")

global year

#insight=Tk()

insight = Toplevel(screen)

adjustWindow(insight)

insight.title("Choose Year")

insight.configure(background='slate blue')

chooseyear=Label(insight,width="500", height="2",text="Choose a Year for piechart" ,font=("Calibri", 22, 'bold'), fg='white', bg='#545454').pack()

insight.geometry("500x500")

#year=IntVar()

def year2017():

soldvleased(2017)

plt.legend(loc='best')

#plt.text(0.05,10,"owned:"+str(sizes[0])+" Leased:"+str(sizes[1]))

def year2018():

soldvleased(2018)

plt.legend(loc='best')

def year2019():

soldvleased(2019)

plt.legend(loc='best')

def year2020():

soldvleased(2020)

plt.legend(loc='best')

Button(insight,text="2017",command=year2017).place(x=10,y=90)

Button(insight,text="2018",command=year2018).place(x=10,y=120)

Button(insight,text="2019",command=year2019).place(x=10,y=150)

Button(insight,text="2020",command=year2020).place(x=10,y=180)

insight.mainloop()

print("The best performer of 2017 is Subbaro and the amount of area sold in SQ-M is 2,15,080.56")

print("The best performer of 2018 is Anderson and the amount of area sold in SQ-M is 105568.05")

print("The best performer of 2019 is Mukesh and the amount of area sold in SQ-M is 106087.4492")

print("B")

maxleasedarea()

def graph2():

excel\_file='D:\Python Internship\Dataset.xlsx'

test = pd.read\_excel(io=excel\_file,header=8)

di={"2017":0,"2018":0,"2019":0,"2020":0}

for row in test.itertuples(index=False):

if row[8]=='HA':

temp=row[7]\*10000

else:

temp=row[7]

if (row[5]=='CA' or row[5]=='WS') and row[9]=='Leased':

di[str(row[0])]=di[str(row[0])]+temp

Keymax = max(di, key=di.get)

fig, ax = plt.subplots(figsize =(10, 10))

ax.bar(range(len(di)), list(di.values()), align='center',color='#ff22ff')

for i in ax.patches:

plt.text(i.get\_x()+0.5, i.get\_height()+0.2,str(round((i.get\_height()), 2)), fontsize = 20, fontweight ='bold', color ='black')

#plt.text(886876,886876,di)

plt.xticks(range(len(di)), list(di.keys()))

print(di)

#messagebox.showinfo("Information",di)

#print("A")

#print("B")

plt.grid(which='major', linestyle='-', linewidth='0.5', color='#ff22ff')

plt.ylabel("Area in square metre")

plt.xlabel("Year")

plt.show()

def graph3():

con=connect(user="root",password="",host="localhost",database="sales")

df=sql.read\_sql('select AGENT\_CODE, AGENT\_NAME FROM agents',con)

print(df)

df.to\_excel('D:\\Python Internship\\agent code and name.xlsx',index=False)

dataset1="D:\\Python Internship\\Dataset.xlsx"

dataset2="D:\\Python Internship\\agent code and name.xlsx"

df1=pd.read\_excel(dataset1,header=8)

df2=pd.read\_excel(dataset2,header=0)

print(list(df2.iloc[:,0]))

res=dict(zip(list(df2.iloc[:,1]),list(df2.iloc[:,0])))

print(res)

df3=df1.copy()

df3['Agent\_Code']=df3['Agent']

df3["Agent\_Code"].replace(

{'Alex': 'A003 ',

'Alford': 'A008 ',

'Anderson': 'A005 ',

'Benjamin': 'A009 ',

'Ivan': 'A004 ',

'Lucida': 'A012 ',

'McDen': 'A006 ',

'Mukesh': 'A002 ',

'Ramasundar ': 'A007 ',

'Ramasundar ': 'A007',

'Ravi Kumar': 'A011 ',

'Santakumar': 'A010 ',

'Subbarao': 'A001 '}, inplace=True)

print(df3)

df3=df3.to\_excel("D:\\Python Internship\\Recent Dataset.xlsx")

df4=pd.read\_excel("D:\\Python Internship\\Recent Dataset.xlsx")

property\_owned=df4[df4['UoM']=='SQ-M']

property\_owned=property\_owned[property\_owned['Tenure']=='Owned']

print(property\_owned)

property\_owned\_id=property\_owned.to\_excel("D:\\Python Internship\\propertyownedidsqm.xlsx")

new\_excel\_owned=pd.read\_excel("D:\\Python Internship\\propertyownedidsqm.xlsx")

x1=new\_excel\_owned['Agent\_Code']

y=new\_excel\_owned['Area']

fig, ax = plt.subplots(figsize =(10, 10))

ax.bar(x1,y,align='center',color='red')

plt.text(0,6674.77,"6694.77",fontsize = 20, fontweight ='bold', color ='black')

plt.text(1,12509.0,"12509.0",fontsize = 20, fontweight ='bold', color ='black')

plt.text(2,45692.2,"45692.2",fontsize = 20, fontweight ='bold', color ='black')

plt.text(3,23290.3,"23290.3",fontsize = 20, fontweight ='bold', color ='black')

plt.text(4,12238.0,"12238.0",fontsize = 20, fontweight ='bold', color ='black')

plt.text(5,18367.9,"18367.9",fontsize = 20, fontweight ='bold', color ='black')

plt.text(6,29870.41,"29870.41",fontsize = 20, fontweight ='bold', color ='black')

plt.text(7,13694.0,"13694.0",fontsize = 20, fontweight ='bold', color ='black')

plt.text(8,8890.0,"8890.0",fontsize = 20, fontweight ='bold', color ='black')

plt.yticks(rotation=90)

plt.grid(which='major', linestyle='-', linewidth='0.5', color='red')

# Customize the minor grid

plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')

#messagebox.showinfo("Best Performer","The best performer of 2017 is Subbaro and the amount of area sold in SQ-M is 2,15,080.56 \nThe best performer of 2018 is Anderson and the amount of area sold in SQ-M is 105568.05 \nThe best performer of 2019 is Mukesh and the amount of area sold in SQ-M is 106087.4492")

#messagebox.showinfo("Information",res)

plt.xlabel('Total Area owned by agent id in SQ-M')

plt.ylabel('Agents id')

plt.title("Property Insights")

def graph4():

df1=pd.read\_excel("D:/Python Internship/Recent Dataset.xlsx")

cityleased=df1[df1['City']=='Chilliwack']

cityleased=cityleased[cityleased['Tenure']=='Leased']

cityleased=cityleased[cityleased['UoM']=='SQ-M']

#print(cityleased)

new\_excel=cityleased.to\_excel("Chilliwack city leased.xlsx",index=False)

df2=pd.read\_excel("Chilliwack city leased.xlsx",index=False)

x=df2['Agent']

y=df2['Area']

fig, ax = plt.subplots(figsize =(10, 10))

ax.bar(x,y,align='center',color='#FFFF49')

plt.grid(which='major', linestyle='-', linewidth='0.5', color='black')

# Customize the minor grid

plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')

plt.text(0,1882.0,"1882.0",fontsize = 20, fontweight ='bold', color ='black')

plt.text(1,1328.47,"1328.47",fontsize = 20, fontweight ='bold', color ='black')

plt.xticks(rotation=90)

plt.ylabel('Total Area Sold by agents')

plt.xlabel('My Agents')

plt.title("Property Insights: Chilliwack's Analysis")

plt.show()

def graph5():

excel\_file='D:\\Python Internship\\Dataset.xlsx'

test = pd.read\_excel(excel\_file,header=8)

di={};

for row in test.itertuples(index=False):

if row[8]=='HA':

temp=row[7]\*10000

else:

temp=row[7]

if di.get(str(row[11]).strip())==None:

di[str(row[11]).strip()]=temp

elif row[0]!=2020:

di[str(row[11]).strip()]=di.get(str(row[11]).strip())+temp

#fig = plt.figure()

#ax = fig.add\_axes([1,0,1,1])

fig, ax = plt.subplots(figsize =(16, 9))

agents = list(di.keys())

land = list(di.values())

ax.barh(agents,land)

for i in ax.patches:

plt.text(i.get\_width()+0.2, i.get\_y()+0.5,str(round((i.get\_width()), 2)),fontsize = 20, fontweight ='bold',color ='black')

plt.yticks(rotation=45)

plt.xlabel("Land owned/leased")

plt.show()

def graph6():

excel\_file='D:\Python Internship\Dataset.xlsx'

test = pd.read\_excel(io=excel\_file,header=8)

ans={"2017":0,"2018":0,"2019":0,"2020":0}

for row in test.itertuples(index=False):

if row[8]=='HA':

temp=row[7]\*10000

else:

temp=row[7]

if row[1]=='JUL':

if row[9]=='Owned':

ans[str(row[0])]=ans[str(row[0])]+temp

#fig = plt.figure()

#ax = fig.add\_axes([0,0,1,1])

fig, ax = plt.subplots(figsize =(10, 10))

agents = list(ans.keys())

land = list(ans.values())

plt.ylabel("Land owned(in sq.m)")

#plt.bar(agents,land)

ax.bar(agents,land,color='#8000ff')

#print(ans)

for i in ax.patches:

plt.text(i.get\_x()+0.5, i.get\_height()+0.2,str(round((i.get\_height()), 2)), fontsize = 20, fontweight ='bold', color ='black')

#messagebox.showinfo("Information",ans)

sl=Button(root,text="Graph for Sold Vs Leased",bg="yellow",fg="black",command=graph1).place(x=10,y=20)

ml=Button(root,text="Graph for Max leased Area in CA & WS",bg="yellow",fg="black",command=graph2).place(x=10,y=60)

do=Button(root,text="Graph for owned property with reference to agent id",bg="yellow",fg="black",command=graph3).place(x=10,y=100)

cmd=Button(root,text="Chilliwack max. deals for lease",bg="yellow",fg="black",command=graph4).place(x=10,y=140)

bp=Button(root,text="Best Performer",bg="yellow",fg="black",command=graph5).place(x=10,y=180)

area=Button(root,text="Area sold for July month",bg="yellow",fg="black",command=graph6).place(x=10,y=220)

tsa=Button(root,text="Time series Analysis",bg="yellow",fg="black").place(x=10,y=260)

root.mainloop()

def main\_screen():

global screen, username\_verify, password\_verify

screen = Tk() # initializing the tkinter window

username\_verify = StringVar()

password\_verify = StringVar()

adjustWindow(screen)

screen.title("Sunville properties") # mentioning title of the window # configuring the window

screen.configure(bg='light blue')

Label(screen, text="WELCOME TO SUNVILLE", width="500", height="2", font=("Calibri", 22, 'bold'), fg='white', bg='#545454').pack()

Label(text="", bg='light blue').pack() # for leaving a space in between

Label (screen, text="Please enter details below to login”, bg='light blue',fg='red').pack()

Label(text="", bg='light blue').pack()

Label(screen, text="Username", font=("Open Sans", 10, 'bold'), bg='light blue', fg='#174873').pack()

Label (text="", bg='light blue'). pack ()

Entry(screen, textvar=username\_verify).pack()

Label (text="", bg='light blue'). pack ()

Label (screen, text="Password \* ", font= ("Open Sans", 10, 'bold'), bg='light blue', fg='#174873'). pack ()

Label (text="", bg='light blue'). pack ()

Entry (screen, textvar=password\_verify, show="\*"). pack ()

Label (text="", bg='light blue'). pack ()

Button (screen, text="LOGIN", bg="red", width=10, height=1, font=("Open Sans", 10, 'bold'), fg='white’, command=login\_verify).pack()

Label (text="", bg='light blue'). pack ()

Label(text="-------------------------OR-------------------------", bg='light blue',fg='#174873').pack()

Label(text="", bg='light blue').pack()

Button(screen, text="Sign Up Here", height="1", width="20", bg='green', font=("Open Sans", 13, 'bold'), fg='white',command=register).pack()

screen.mainloop()

main\_screen()