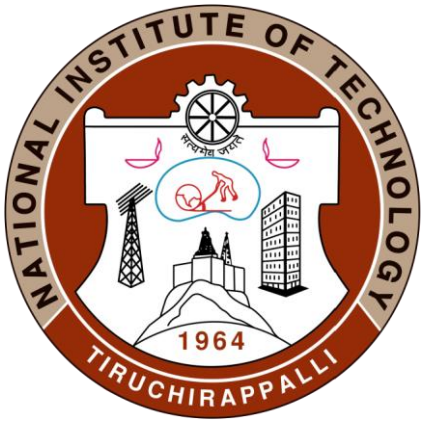
NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI

Tamil Nadu-620015



'Database Management System'

PROJECT REPORT

STORE MANAGEMENT

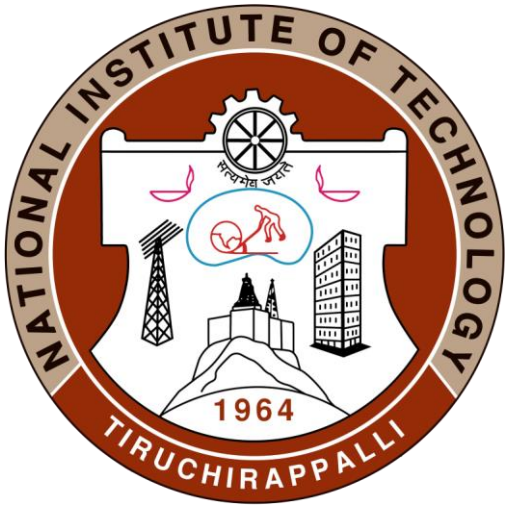
Submitted To:

Dr. U. Vignesh

Submitted By:

Akshay Jain
Roll No. – 205119007
MCA - II Semester 'A'

NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI-15



CERTIFICATE

*This is to certify that **Mr. AKSHAY JAIN**, student of 2nd semester MCA (batch 2019-2022) of National Institute of Technology, Tiruchirappalli has successfully completed the project **STORE MANAGEMENT** in Tkinter(Python)/MySQL under the guidance of **Dr. U. Vignesh.***

Signature

Dr. U. Vignesh

Abstract

The main aim of **Store Management** project is to keep a track of sales, purchases and their effect on inventory. We aim to demonstrate the use of create, read, select, update and delete MySQL operations through this project. The project starts by add items to the inventory by the seller, then a customer, purchase some products from seller (dealer) and note the changes to inventory. The purchased goods can be modified later which demonstrates the update functionality of project. Finally, we can record a sale and note the changes to product quantity in inventory page. The Application is built using Tkinter (Python) and MySQL technologies.

ACKNOWLEDGEMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to NIT, Trichy for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my gratitude towards **Dr. U. Vignesh** for his kind co-operation and encouragement which help me in completion of this project.

Dr. U. Vignesh
(Department of Computer Applications)

INTRODUCTION

A database management system (DBMS) refers to the technology for creating and managing databases. Basically DBMS is a software tool to organize (create, retrieve, update and manage) data in a database.

The main aim of a DBMS is to supply a way to store up and retrieve database information that is both convenient and efficient. By data, we mean known facts that can be recorded and that have embedded meaning. Normally people use software such as DBASE IV or V, Microsoft ACCESS, or EXCEL to store data in the form of database.

Database systems are meant to handle large collection of information. Management of data involves both defining structures for storage of information and providing mechanisms that can do the manipulation of those stored information. Moreover, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access.

This project is aim at computerizing the manual process of wedding system. Front end and backend are implemented using Tkinter and MySQL respectively. The project consists of different forms(entity) namely Add, Update, Billing which are used for maintaining stock of store. The forms have number of entries. As well as each entry will be used to hold the information of items in the inventory.

The services of a Store Management System can include:

- Holding information about the items in stock.
- Adding information of new stocks.
- Updating information of current stocks.
- Searching information of item with the ID.
- Generating Invoice of items purchased by the customer.
- Keeping records of daily transactions.

Database Management System

DBMS stands for Database Management System. We can break it like this DBMS = Database + Management System. Database is a collection of data and Management System is a set of programs to store and retrieve those data. Based on this we can define DBMS like this: DBMS is a collection of inter-related data and set of programs to store and access those data in an easy and effective manner.

Database system are basically developed for large amount of data. When dealing with huge amount of data, there are two things that require optimization: Storage of data and retrieval of data. According to the principles of database systems, the data is stored in such a way that it acquires a lot less space as the redundant data(duplicate data) has been removed before storage.

Along with storing the data in an optimized and systematic manner, It is also important that we retrieve the data quickly when needed. Database system ensures that data is retrieved as quickly as possible.

Applications of DBMS

The development of computer graphics has been driven both by the needs of the user community and by the advances in hardware and software. The applications of database are many and varied; it can be divided into four major areas

1. Hierarchical and network system
2. Flexibility with relational database.
3. Object oriented application.
4. Interchanging the data on the web for e-commerce.

Display information

In this particular project, we have taken Tkinter as a front end in order to display the information which are stored in the backend database called MySQL.

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

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the Tkinter module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

User Interfaces

Our interactions with computers has become dominated by a visual paradigm that includes windows, buttons, menus, pointing device, such as a mouse. Although we are familiar with the syntax of MySQL, advances in MySQL have made possible other forms of advantages.

What is MySQL?

MySQL is multithreaded, multi user SQL database management System (DBMS). The basic program run as server providing multiuser access to a number of databases. The project's source code is available under terms of the GNU(General Public Union), as well as under a variety of

property arguments. MySQL is a database. The data in a MySQL is stored in a Database objects called tables. A table is a collection of related data entries and it consists of columns and rows. The databases are useful when storing information categorically.

MySQL is a central components of the LAMP open source web application software stack (and other “AMP” stacks). LAMP is an acronym for Linux, Apache, MySQL, Perl/PHP/Python. Application that use the MySQL database include PyCharm, TYP03, MODx, Joomla, WordPress, PHPBB, MyBB and Drupal. MySQL is also used in many high profile, large scale web sites, including Google(Though not for the searches).

MySQL Command Syntax

As you might have observed from the simple program in the previous section, MySQL uses mainly uses six commands in which SELECT is used to retrieve rows selected from one or more tables. FROM refers to the table from which we need to select the attributes. WHERE clause, if given, indicates condition or conditions that rows must satisfy to be selected. where_ condition is expression that evaluates to true for each row to be selected. This statement selects all rows if there is no where clause. GROUP BY clause used to group the values of the attributes provided that values must be same. HAVING clause is applied nearly last, just before items are sent to the client, with no optimization. If the HAVING clause refers to a column that is ambiguous, warning occurs. ORDER BY clause is used for the purpose of sorting the values of the attributes in a result. If you use GROUP BY, output rows are sorted according to GROUP BY columns as if you had an ORDER BY for the same columns.

Purpose

The purpose of this project is to outline Inventory data and to recommend data management solutions and to provide a information regarding the stock. The purpose of this project is to develop a data management system to consolidate, organize, document, store and distribute information related to Store Management System.

A centralized database created to consolidate data, allowing integrated, long term analyses, and dynamic search ability with user friendly query tools to be performed to support adaptive management. Many data collection, analysis and presentation software programs that are currently being used must be able to interface with any new data management system. Continuity with consistent data collection methodology is enforced by a common database system, allowing for standardized format for forms ad reports between projects.

Scope

The scope of the project is managing a consistency and storage of data by dedicated data administrator. It provides most of the features that a Database Management System should have. It is developed by using MySQL database. It has been implemented in WINDOWS platform.

Hardware specification

Processor	:	i3 Core Processor
Clock speed	:	2.5GHz
Monitor	:	1024 * 768 Resolution Color
Keyboard	:	QWERTY
RAM	:	1 GB

Input Output Console for interaction

Software specification

MySQL Libraries
MySQL Client Server
PhyCham

Operating system	:	Windows7

DESIGN OF THE PROJECT

This project has been developed using MySQL software which is queries oriented. Changes at the queries and the way in which it uses a system state may cause anticipated changes in the behaviour of other result.

Schema and Tables Description

Following tables are used in the project:-

1. Inventory:

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
name	varchar(50)	NO		NULL	
stock	int	NO		NULL	
cp	int	YES		NULL	
sp	int	YES		NULL	
totalcp	int	YES		NULL	
totalsp	int	YES		NULL	
assumed_pofit	int	YES		NULL	
vendor	varchar(50)	YES		NULL	
vendor_phoneno	bigint	YES		NULL	

2. Transactions:

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
product_name	varchar(50)	NO		NULL	
quantity	int	NO		NULL	
amount	int	YES		NULL	
tdate	date	YES		NULL	

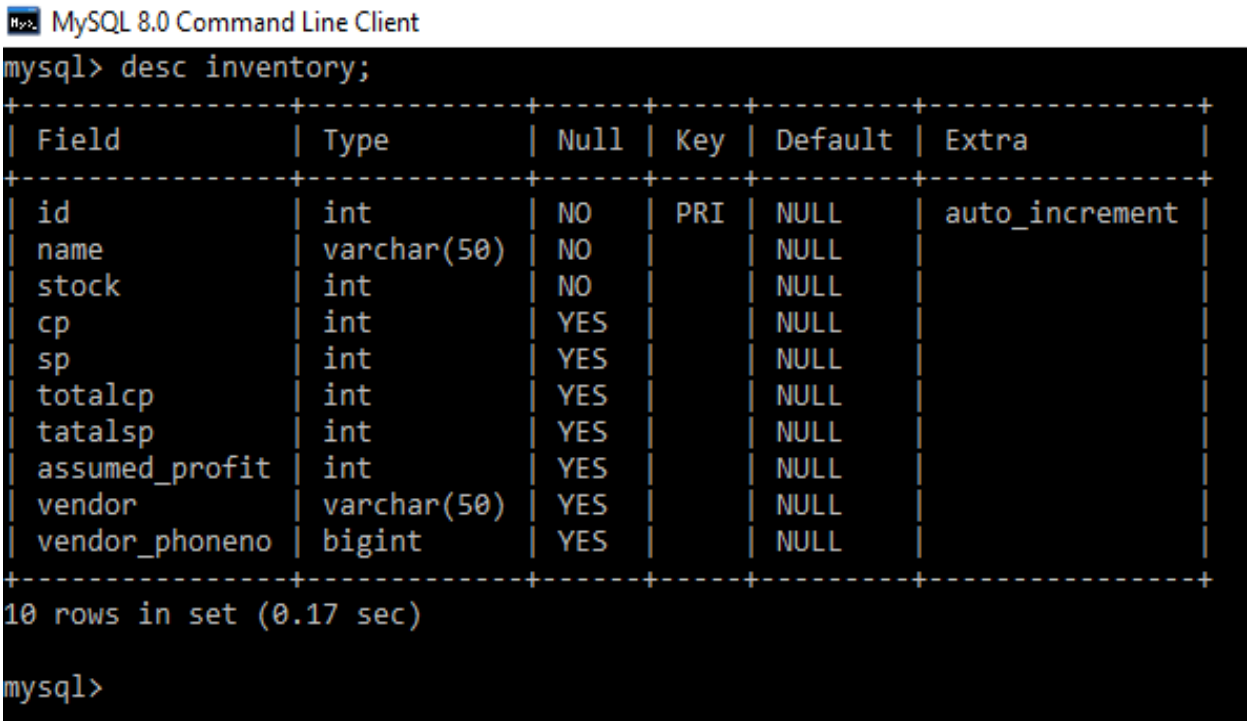
IMPLEMENTATION

The project is implemented using MySQL database along with Tkinter.

Implementation of Table Creation:

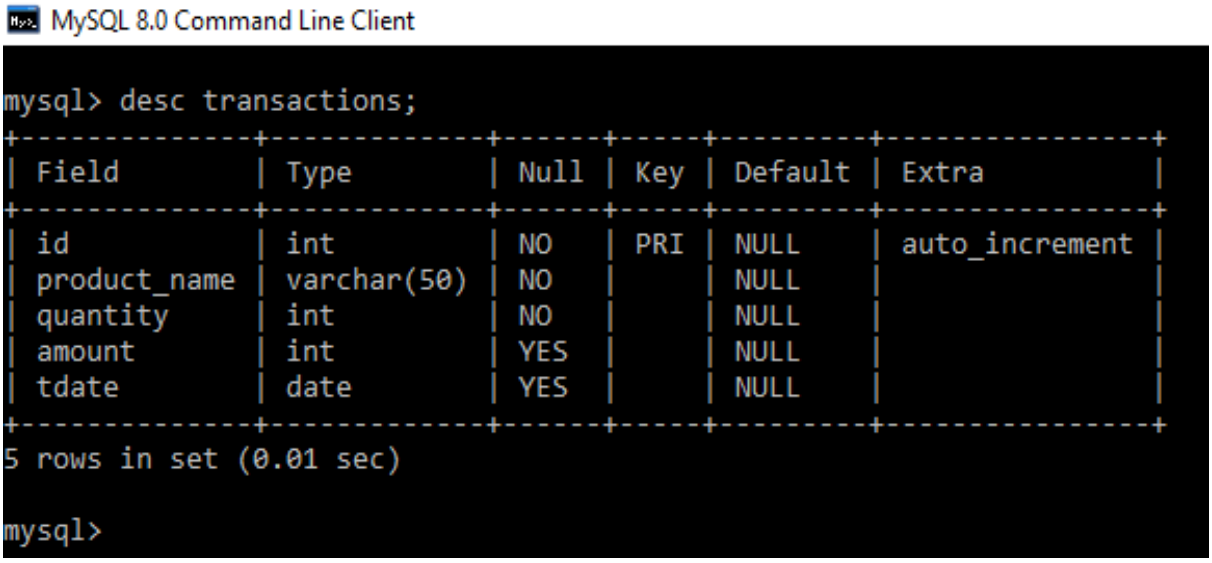
1. Inventory –

create table if not exists inventory
(id int not null auto_increment, name varchar(50) not null,
stock int not null, cp int, sp int, totalcp int, tatalsp int,
assumed_profit int, vendor varchar(50), vendor_phoneno bigint,
primary key(id));



2. Transactions –

create table if not exists transactions
(id int not null auto_increment, product_name varchar(50) not null, quantity int not null, amount int, tdate date, primary key(id));



Project Description:

DBMS Modules –

1. Inventory: As the name suggests, records of items are hold in this table with the help of Primary key(id).

2. Transactions: Each and every transactions happening in the store are placed in this table with their respective date of selling.

GUI Modules –

1. Add items to Database: This form is used to add each and every item in the store to the inventory table of the store database by using insert query with their name, stock, cost price, selling price, total cost price, total selling price, vendor, vendor phone number.

2. Update information of items to database: This form is used to update information of any item present in the inventory table of store database with the help of it's item's id (Primary Key).

3. Search items: This form is used to search information about any item present in the database with it's item's id.

4. Generate Bill: This form is used to Generate Bill of all the items that a customer wants to buy and to calculate the total cost which is to be paid by the customer.

5. Change Button: This button is used to find how much change is to be returned to the customer if he paid more than the bill amount.

6. Print Invoice: By clicking generate a bill a process is initiated which will contact to the printing device so the invoice can be printed.

7. Clear Button: This Button is used to clear all entry field of the form.

Source Code

GIT Link:

https://github.com/rudeakshay/205119007_DBMS_Project_StoreManagement

Note: Source codes are in text format not snapshots.

//source code for **add_to_db.py**

```
# import all the modules
import tkinter
from tkinter import *
import mysql.connector
import tkinter.messagebox

conn = mysql.connector.connect(host='localhost', user='root', passwd='7729', database='store',use_pure=True )
con = conn.cursor()

s = "create table if not exists inventory(id int not null auto_increment, name varchar(50) not null, stock int not null," \
    "cp int, sp int, totalcp int, tatalsp int, assumed_profit int, vendor varchar(50), vendor_phoneno bigint, primary key(id))"

con.execute(s)
conn.commit()

con.execute("SELECT Max(id) FROM inventory")
result = con.fetchall()
if result:
    for r in result:
        id = r[0]

class Database:
    def __init__(self, master, *args, **kwargs):

        self.master = master
        self.heading = Label(master, text="Add To The Database", font=('arial 40 bold'), fg='steelblue')
```

```
        self.heading.place(x=450, y=0)

        # labels for the window
        self.name_l = Label(master, text="Enter Product Name", font=('arial 18 bo
ld'))
        self.name_l.place(x=0, y=70)

        self.stock_l = Label(master, text="Enter Stocks", font=("arial 18 bold"))
        self.stock_l.place(x=0, y=120)

        self.cp_l = Label(master, text="Enter Cost Price", font=("arial 18 bold")
)
        self.cp_l.place(x=0, y=170)

        self.sp_l = Label(master, text="Enter Selling Price", font=("arial 18 bol
d"))
        self.sp_l.place(x=0, y=220)

        self.vendor_l = Label(master, text="Enter Vendor Name", font=("arial 18 b
old"))
        self.vendor_l.place(x=0, y=270)

        self.vendor_phone_l = Label(master, text="Enter Vendor Phone Number", fon
t=("arial 18 bold"))
        self.vendor_phone_l.place(x=0, y=320)

        self.id_l = Label(master, text="Enter ID", font=("arial 18 bold"))
        self.id_l.place(x=0, y=370)

        # entries for lables
        self.name_e = Entry(master, width=25, font=('arial 18 bold'))
        self.name_e.place(x=400, y=70)

        self.stock_e = Entry(master, width=25, font=('arial 18 bold'))
        self.stock_e.place(x=400, y=120)

        self.cp_e = Entry(master, width=25, font=('arial 18 bold'))
        self.cp_e.place(x=400, y=170)

        self.sp_e = Entry(master, width=25, font=('arial 18 bold'))
        self.sp_e.place(x=400, y=220)

        self.vendor_e = Entry(master, width=25, font=('arial 18 bold'))
        self.vendor_e.place(x=400, y=270)
```

```
        self.vendor_phone_e = Entry(master, width=25, font=('arial 18 bold'))
        self.vendor_phone_e.place(x=400, y=320)

        self.id_e = Entry(master, width=25, font=('arial 18 bold'))
        self.id_e.place(x=400,y=370)

        # button to add to the database
        self.btn_clear = Button(master, text="Clear All Fields", width=25, height
=2, bg='red', fg='white', command=self.clear_all)
        self.btn_clear.place(x=320, y=420)

        self.btn_add = Button(master, text="Add To Database", width=25, height=2,
 bg='steelblue', fg='white', command = self.get_items)
        self.btn_add.place(x=520, y=420)

        # text box for the logs
        self.tBox = Text(master, width=60, height=20)
        self.tBox.place(x=800, y=70)
        self.tBox.insert(END, "ID has reached upto: " + str(id))

    def get_items(self, *args, **kwaargs):
        conn = mysql.connector.connect(host='localhost', user='root', passwd='772
9', database='store',use_pure=True )
        con = conn.cursor()
        # get data from entries
        self.name = self.name_e.get()
        self.stock = self.stock_e.get()
        self.cp = self.cp_e.get()
        self.sp = self.sp_e.get()
        self.vendor = self.vendor_e.get()
        self.vendor_phone = self.vendor_phone_e.get()

        # dynamic entries
        self.totalcp = float(self.cp) * float(self.stock)
        self.totalsp = float(self.sp) * float(self.stock)
        self.assumed_profit = float(self.totalsp - self.totalcp)

        if self.name == '' or self.stock == '' or self.cp == '' or self.sp == '':
            tkinter.messagebox.showinfo("Error", "Please Fill All Entries.")
        else:
            sql = "INSERT INTO inventory (name, stock, cp, sp, totalcp, tatalsp,
assumed_profit, vendor, vendor_phoneno) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %
s)"
            con.execute(sql, (self.name, self.stock, self.cp, self.sp, self.total
cp, self.totalsp, self.assumed_profit, self.vendor, self.vendor_phone))
```

```
            conn.commit()
            self.tBox.insert(END, "\n\nInserted " + str(self.name) + " into the d
atabase with code " + str(self.id_e.get()))
            tkinter.messagebox.showinfo("Success", "Successfully added to the Dat
abase.")
        conn.close()

    def clear_all(self, *args, **kwargs):
        # num=id+1
        self.name_e.delete(0, END)
        self.stock_e.delete(0, END)
        self.cp_e.delete(0, END)
        self.sp_e.delete(0, END)
        self.vendor_e.delete(0, END)
        self.vendor_phone_e.delete(0, END)
        self.id_e.delete(0, END)

root = Tk()
b = Database(root)

root.geometry("1366x768+0+0")
root.title("Add To Database")
root.mainloop()
```

//source code for **update.py**

```
# import all the modules
import tkinter
from tkinter import *
import mysql.connector
import tkinter.messagebox

conn = mysql.connector.connect(host='localhost', user='root', passwd='7729', dat
abase='store', use_pure=True)
con = conn.cursor()

s = "create table if not exists inventory(id int not null auto_increment, name v
archar(50) not null, stock int not null," \
    "cp int, sp int, totalcp int, tatalsp int, assumed_profit int, vendor varcha
r(50), vendor_phoneno bigint, primary key(id))"

con.execute(s)
conn.commit()

con.execute("SELECT Max(id) FROM inventory")
result = con.fetchall()
if result:
    for r in result:
        id = r[0]

class Database:
    def __init__(self, master, *args, **kwargs):
        self.master = master
        self.heading = Label(master, text="Update To The Database", font=('arial
 40 bold'), fg='steelblue')
        self.heading.place(x=400, y=0)

        # label and entry for id
        self.id_le = Label(master, text="Enter ID", font=("arial 18 bold"))
        self.id_le.place(x=0, y=70)

        self.id_leb = Entry(master, width=10, font=('arial 20 bold'))
        self.id_leb.place(x=400, y=70)

        self.btn_search = Button(master, text="Search", width=15, height=2, bg='
orange', command=self.search)
        self.btn_search.place(x=580, y=70)

        # labels for the window
```

```
        self.name_l = Label(master, text="Enter Product Name", font=('arial 18 b
old'))
        self.name_l.place(x=0, y=120)

        self.stock_l = Label(master, text="Enter Stocks", font=("arial 18 bold")
)
        self.stock_l.place(x=0, y=170)

        self.cp_l = Label(master, text="Enter Cost Price", font=("arial 18 bold"
))
        self.cp_l.place(x=0, y=220)

        self.sp_l = Label(master, text="Enter Selling Price", font=("arial 18 bo
ld"))
        self.sp_l.place(x=0, y=270)

        self.totalcp_l = Label(master, text="Enter Total Cost Price", font=("ari
al 18 bold"))
        self.totalcp_l.place(x=0, y=320)

        self.totalsp_l = Label(master, text="Enter Total Selling Price", font=("
arial 18 bold"))
        self.totalsp_l.place(x=0, y=370)

        self.vendor_l = Label(master, text="Enter Vendor Name", font=("arial 18
bold"))
        self.vendor_l.place(x=0, y=420)

        self.vendor_phone_l = Label(master, text="Enter Vendor Phone Number", fo
nt=("arial 18 bold"))
        self.vendor_phone_l.place(x=0, y=470)

        # entries for lables
        self.name_e = Entry(master, width=25, font=('arial 18 bold'))
        self.name_e.place(x=400, y=120)

        self.stock_e = Entry(master, width=25, font=('arial 18 bold'))
        self.stock_e.place(x=400, y=170)

        self.cp_e = Entry(master, width=25, font=('arial 18 bold'))
        self.cp_e.place(x=400, y=220)

        self.sp_e = Entry(master, width=25, font=('arial 18 bold'))
        self.sp_e.place(x=400, y=270)
```

```
        self.totalcp_e = Entry(master, width=25, font=('arial 18 bold'))
        self.totalcp_e.place(x=400, y=320)

        self.totalsp_e = Entry(master, width=25, font=('arial 18 bold'))
        self.totalsp_e.place(x=400, y=370)

        self.vendor_e = Entry(master, width=25, font=('arial 18 bold'))
        self.vendor_e.place(x=400, y=420)

        self.vendor_phone_e = Entry(master, width=25, font=('arial 18 bold'))
        self.vendor_phone_e.place(x=400, y=470)

        # button to add to the database
        self.btn_clear = Button(master, text="Clear All Fields", width=25, heigh
t=2, bg='red', fg='white')
        self.btn_clear.place(x=320, y=520)

        self.btn_update = Button(master, text="Update Database", width=25, heigh
t=2, bg='steelblue', fg='white', command=self.update)
        self.btn_update.place(x=520, y=520)

        # text box for the logs
        self.tBox = Text(master, width=60, height=20)
        self.tBox.place(x=800, y=70)
        self.tBox.insert(END, "")

    def search(self, *args, **kwargs):
        sql = "SELECT * FROM inventory WHERE id=%s"
        con.execute(sql, (self.id_leb.get(), ))
        result = con.fetchall()

        for r in result:
            self.n1 = r[1]  # name
            self.n2 = r[2]  # stock
            self.n3 = r[3]  # cp
            self.n4 = r[4]  # sp
            self.n5 = r[5]  # totalcp
            self.n6 = r[6]  # tatalsp
            self.n7 = r[7]  # assumed_profit
            self.n8 = r[8]  # vendor
            self.n9 = r[9]  # vendor_phoneno
        conn.commit()

        # insert into the entries to update
        self.name_e.delete(0, END)
```

```
        self.name_e.insert(0, str(self.n1))

        self.stock_e.delete(0, END)
        self.stock_e.insert(0, str(self.n2))

        self.cp_e.delete(0, END)
        self.cp_e.insert(0, str(self.n3))

        self.sp_e.delete(0, END)
        self.sp_e.insert(0, str(self.n4))

        self.totalcp_e.delete(0, END)
        self.totalcp_e.insert(0, str(self.n5))

        self.totalsp_e.delete(0, END)
        self.totalsp_e.insert(0, str(self.n6))

        self.vendor_e.delete(0, END)
        self.vendor_e.insert(0, str(self.n8))

        self.vendor_phone_e.delete(0, END)
        self.vendor_phone_e.insert(0, str(self.n9))

    def update(self, *args, **kwargs):
        con = conn.cursor()
        # get all updated values
        self.u1 = self.name_e.get()
        self.u2 = self.stock_e.get()
        self.u3 = self.cp_e.get()
        self.u4 = self.sp_e.get()
        self.u5 = self.totalcp_e.get()
        self.u6 = self.totalsp_e.get()
        self.u7 = self.vendor_e.get()
        self.u8 = self.vendor_phone_e.get()

        query = "UPDATE inventory SET name=%s, stock=%s, cp=%s, sp=%s, totalcp=%
s, tatalsp=%s, vendor=%s, vendor_phoneno=%s WHERE id=%s"
        con.execute(query, (self.u1, self.u2, self.u3, self.u4, self.u5, self.u6
, self.u7, self.u8, self.id_leb.get()))
        conn.commit()
        tkinter.messagebox.showinfo("Success","Database Updated!!!")


root = Tk()
b = Database(root)
```

```
root.geometry("1366x768+0+0")
root.title("Update To Database")
root.mainloop()
```

//source code for **main.py**

```
# import all the modules
from tkinter import *
import mysql.connector
import tkinter.messagebox
import datetime
import os
import random

conn = mysql.connector.connect(host='localhost', user='root', passwd='7729', dat
abase='store',use_pure=True )
con = conn.cursor()

# creating transactions table
s = "create table if not exists transactions(id int not null auto_increment, pro
duct_name varchar(50) not null, quantity int not null," \
    "amount int, tdate date, primary key(id))"

con.execute(s)
conn.commit()

# date
date = datetime.datetime.now().date()

# temporary lists like sessions
products_list = []
product_price = []
product_quantity = []
product_id =[]

# labels list
lables_list = []

class Application:
    def __init__(self, master, *args, **kwargs):

        self.master = master
        # frames
        self.left = Frame(master, width=700, height=768, bg='white')
        self.left.pack(side=LEFT)

        self.right = Frame(master, width=666, height=768, bg='lightblue')
        self.right.pack(side=RIGHT)
```

```
        # components
        self.heading = Label(self.left, text="2k Market", font=('arial 40 bold')
, bg='white')
        self.heading.place(x=0,y=0)

        self.date_l = Label(self.right, text="Today's Date: " + str(date), font=
('arial 18 bold'), bg='lightblue', fg='white')
        self.date_l.place(x=0, y=0)

        # table invoice ==========================================================
==========
        self.tproduct = Label(self.right, text="Products", font=('arial 18 bold'
), bg='lightblue', fg='white')
        self.tproduct.place(x=0,y=60)

        self.tquantity = Label(self.right, text="Quantity", font=('arial 18 bold
'), bg='lightblue', fg='white')
        self.tquantity.place(x=300, y=60)

        self.tamount = Label(self.right, text="Amount", font=('arial 18 bold'),
bg='lightblue', fg='white')
        self.tamount.place(x=500, y=60)

        # enter stuff
        self.enterid = Label(self.left, text="Enter Product's ID", font=('arial
18 bold'), bg='white')
        self.enterid.place(x=0, y=80)

        self.enteride = Entry(self.left, width=25, font=('arial 18 bold'), bg='l
ightblue')
        self.enteride.place(x=230, y=80)
        self.enteride.focus()

        self.search_btn = Button(self.left, text="Search", width=22, height=2, b
g='orange', command=self.ajax)
        self.search_btn.place(x=350, y=120)

        # fill it by the function ajax
        self.productname = Label(self.left, text="", font=('arial 27 bold'), bg=
'white', fg='steelblue')
        self.productname.place(x=0, y=250)

        self.pprice = Label(self.left, text="", font=('arial 27 bold'), bg='whit
e', fg='steelblue')
        self.pprice.place(x=0, y=290)
```

```
        # total label
        self.total_l= Label(self.right, text="", font=('arial 40 bold'), bg='lig
htblue', fg='white')
        self.total_l.place(x=0, y=600)

        self.master.bind("<Return>", self.ajax)
        self.master.bind("<Up>", self.add_to_cart)
        self.master.bind("<space>", self.generate_bill)

    def ajax(self, *args, **kwargs):
        self.get_id=self.enteride.get()
        # get the productsinfo with there id and fill in the labels above
        query="SELECT * FROM inventory WHERE id=%s"
        con.execute(query,(self.get_id, ))
        result=con.fetchall()
        for self.r in result:
            self.get_id= self.r[0]
            self.get_name= self.r[1]
            self.get_price= self.r[4]
            self.get_stock= self.r[2]
        self.productname.configure(text="Product's Name: " + str(self.get_name))
        self.pprice.configure(text="Price: Rs. " + str(self.get_price))

        # create the quatity and discount label
        self.quantity_l=Label(self.left, text="Enter Quantity", font=('arial 18
bold'), bg='white')
        self.quantity_l.place(x=0, y=370)

        self.quantity_e= Entry(self.left, width=25,font=('arial 18 bold'), bg='l
ight blue')
        self.quantity_e.place(x=190, y=370)
        self.quantity_e.focus()

        self.discount_l = Label(self.left, text="Enter Discount", font=('arial 1
8 bold'), bg='white')
        self.discount_l.place(x=0, y=410)

        self.discount_e = Entry(self.left, width=25, font=('arial 18 bold'), bg=
'light blue')
        self.discount_e.place(x=190, y=410)
        self.discount_e.insert(END, 0)

        # add to cart button
```

```
        self.add_to_cart_btn = Button(self.left, text="Add To Cart", width=22, h
eight=2, bg='orange', command=self.add_to_cart)
        self.add_to_cart_btn.place(x=350, y=450)

        #generate bills and change
        self.change_l=Label(self.left,text="Given Amount", font=('arial 18 bold'
), bg='white')
        self.change_l.place(x=0, y=550)

        self.change_e= Entry(self.left, width =25, font=('arial 18 bold'), bg='l
ightblue')
        self.change_e.place(x=190, y=550)

        #button change
        self.change_btn = Button(self.left, text="Calculate Change", width=22, h
eight=2, bg='orange', command=self.change_fun)
        self.change_btn.place(x=350, y=590)

        #generate bill button
        self.generate_bill_btn = Button(self.left, text="Generate Bill", width=8
0, height=2, bg='red', command=self.generate_bill)
        self.generate_bill_btn.place(x=60, y=640)

    def add_to_cart(self, *args, **kwargs):
        #get the quantity value from the database
        self.quantity_value= int(self.quantity_e.get())
        if self.quantity_value > int(self.get_stock):
            tkinter.messagebox.showinfo("Error","Not that many Products in our I
nventory.")
        else:
            #calculate the price
            self.final_price= float(self.quantity_value)*float(self.get_price)-
(float(self.discount_e.get()))

            products_list.append(self.get_name)
            product_price.append(self.final_price)
            product_quantity.append(self.quantity_value)
            product_id.append(self.get_id)

            self.x_index=0
            self.y_index=100
            self.counter=0
            for self.p in products_list:
                self.tempname= Label(self.right, text=str(products_list[self.cou
nter]), font =('arial 18 bold') , bg='lightblue', fg='white')
```

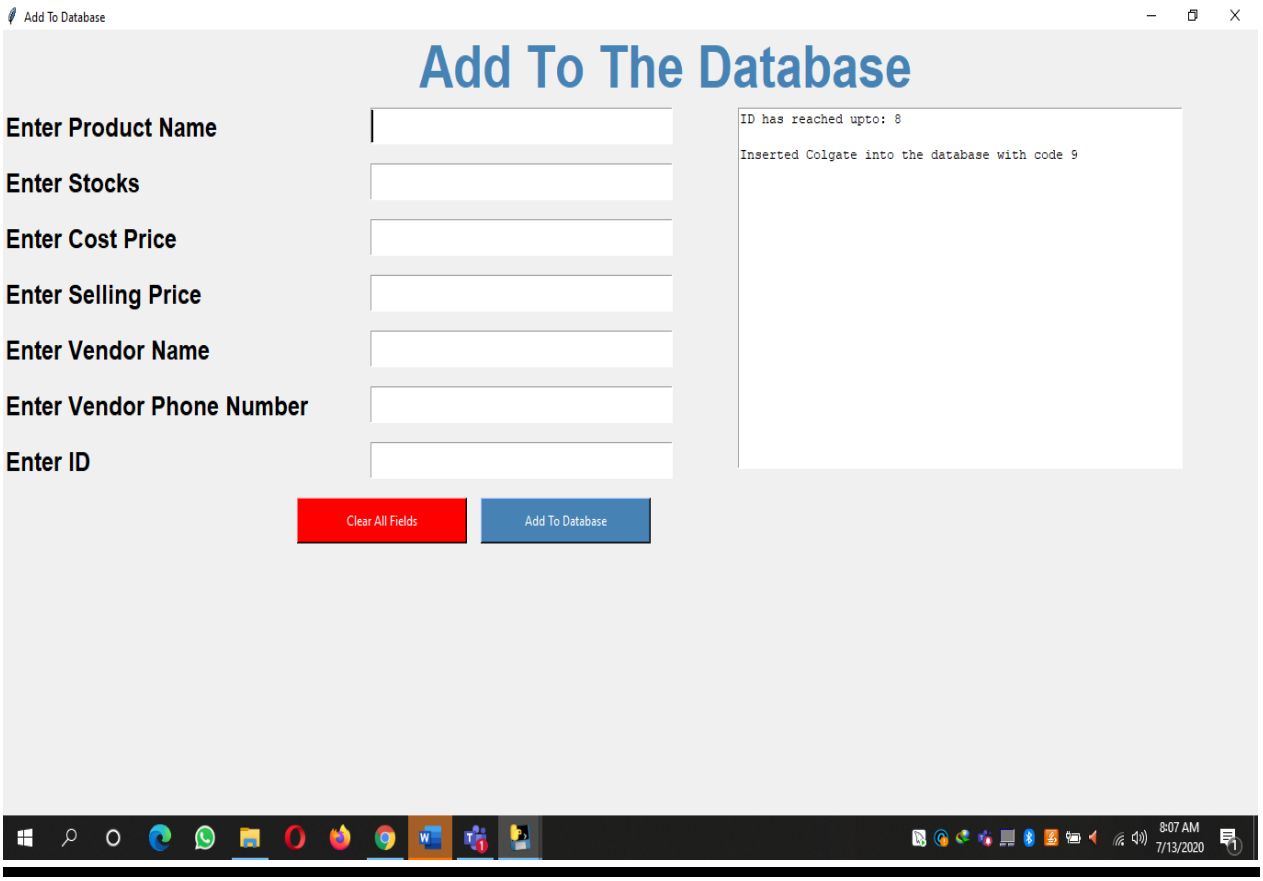
```
                self.tempname.place(x=0, y=self.y_index)
                lables_list.append(self.tempname)

                self.tempqt = Label(self.right, text=str(product_quantity[self.c
ounter]), font=('arial 18 bold'), bg='lightblue',fg='white')
                self.tempqt.place(x=300, y=self.y_index)
                lables_list.append(self.tempqt)

                self.tempprice = Label(self.right, text=str(product_price[self.c
ounter]), font=('arial 18 bold'), bg='lightblue',fg='white')
                self.tempprice.place(x=500, y=self.y_index)
                lables_list.append(self.tempprice)

                self.y_index+=40
                self.counter+=1

                # total configure
                self.total_l.configure(text="Total Rs. " + str(sum(product_price
)))

                # delete
                self.productname.configure(text="")
                self.pprice.configure(text="")
                self.quantity_l.place_forget()
                self.quantity_e.place_forget()
                self.discount_l.place_forget()
                self.discount_e.place_forget()
                self.add_to_cart_btn.destroy()

                # autofocus to the enter id
                self.enteride.focus()
                self.enteride.delete(0, END)

    def change_fun(self):
        # get the amount given by the customer and the amount generated by the c
omputer
        self.amount_given = float(self.change_e.get())
        self.our_total = float(sum(product_price))

        self.to_give = self.amount_given - self.our_total

        #label change
        self.c_amount = Label(self.left, text="Change: Rs. "+str(self.to_give),
font=('arial 18 bold'), fg='red', bg='white')
        self.c_amount.place(x=0, y=600)
```

```
    def generate_bill(self, *args, **kwargs):
        # create the bill before updating the database
        directory = "C:/New folder/StoreManagement/Invoice/" + str(date) + "/"
        if not os.path.exists(directory):
            os.makedirs(directory)

        # templates for the bill
        company = "\t\t\t\t\t2K Market\n"
        address = "\t\tNational Institute of Technology, Tiruchirappalli\n"
        phone = "\t\t\t\t\t9876543210\n"
        sample = "\t\t\t\t\tInvoice\n"
        dt = "\t\t\t\t\t" + str(date)

        table_header = "\n\n\t-------------------------------------------------
---------\n\t\tSNo.\t\tProducts\t\tQty\t\tAmount\n\t-----------------------------
-------------------------------"
        final = company + address + phone + sample + dt + "\n" + table_header

        # open a file to write it to
        file_name = str(directory) + str(random.randrange(5000,10000)) + ".rtf"
        f = open(file_name, 'w')
        f.write(final)
        # fill dynamics
        r = 1
        i = 0
        for t in products_list:
            f.write("\n\t\t" + str(r) + "\t\t" + str(products_list[i] + "     ")
[:10] + "\t\t" + str(product_quantity[i]) +"\t\t" + str(product_price[i]))
            r += 1
            i += 1
        f.write("\n\n\t\t\t\tTotal Amount Rs. " + str(sum(product_price)))
        f.write("\n\t\t\t\tThanks for Visiting.")
        os.startfile(file_name, "print")
        f.close()

        # decrease the stock
        self.x = 0

        for i in products_list:
            initial = "SELECT * FROM inventory WHERE id=%s"
            con.execute(initial, (product_id[self.x],))
            result = con.fetchall()

            for r in result:
```

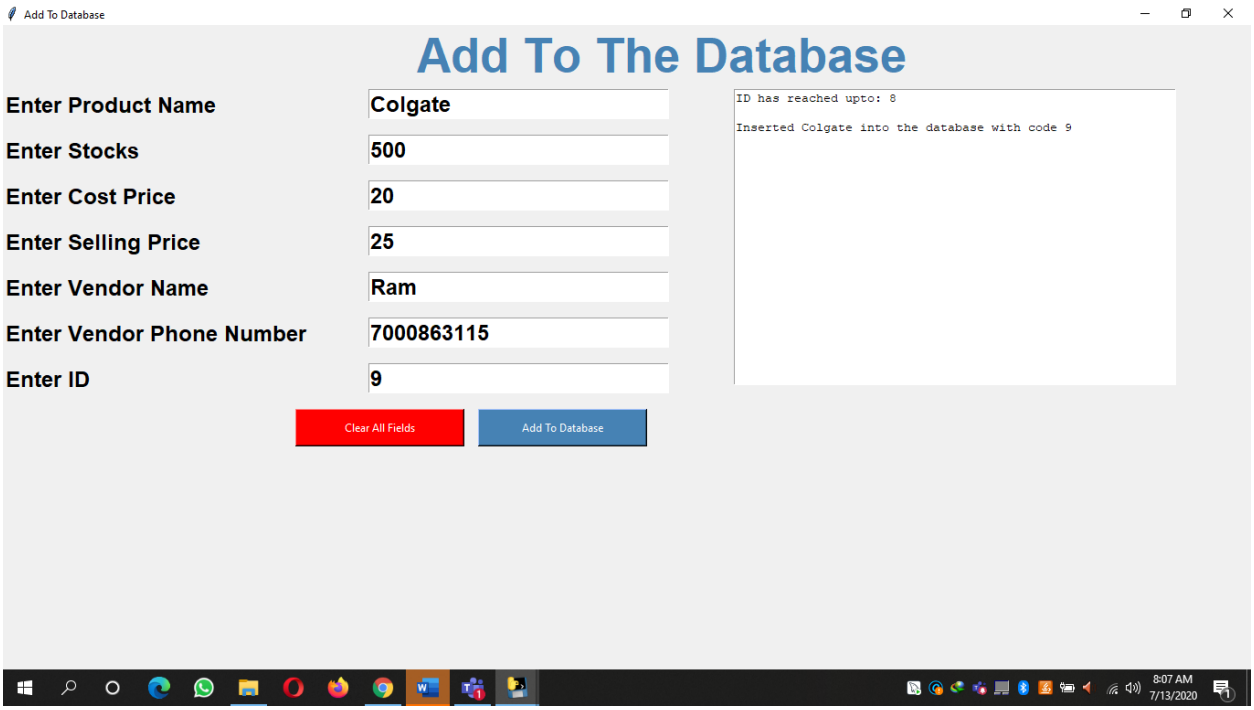
```
                self.old_stock = r[2]

            self.new_stock = int(self.old_stock)-int(product_quantity[self.x])

            # updating the stock
            sql="UPDATE inventory SET stock=%s WHERE id=%s"
            con.execute(sql, (self.new_stock, product_id[self.x]))
            conn.commit()

            # insert into the transaction
            sql2 = "INSERT INTO transactions(product_name, quantity, amount, tda
te) VALUES (%s,%s,%s,%s)"
            con.execute(sql2,(products_list[self.x], product_quantity[self.x], p
roduct_price[self.x], date))
            conn.commit()

            self.x += 1

        for a in lables_list:
            a.destroy()

        del(product_id[:])
        del(products_list[:])
        del(product_quantity[:])
        del(product_price[:])

        self.total_l.configure(text="")
        self.c_amount.configure(text="")
        self.change_e.delete(0, END)
        self.enteride.focus()

        tkinter.messagebox.showinfo("Success", "Done Everything Smoothly.")

root = Tk()
b = Application(root)

root.geometry("1366x768+0+0")
root.mainloop()
```

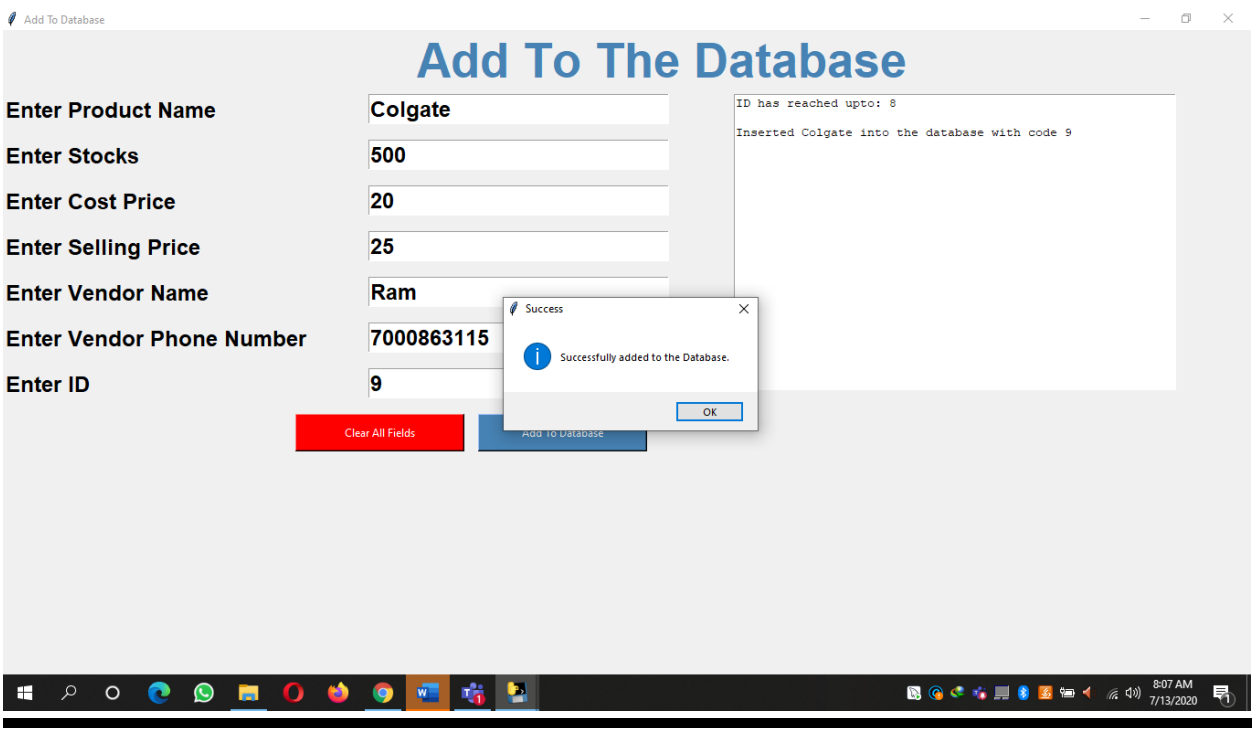
Snapshots

1. Form for Adding Item's Information's to Database:

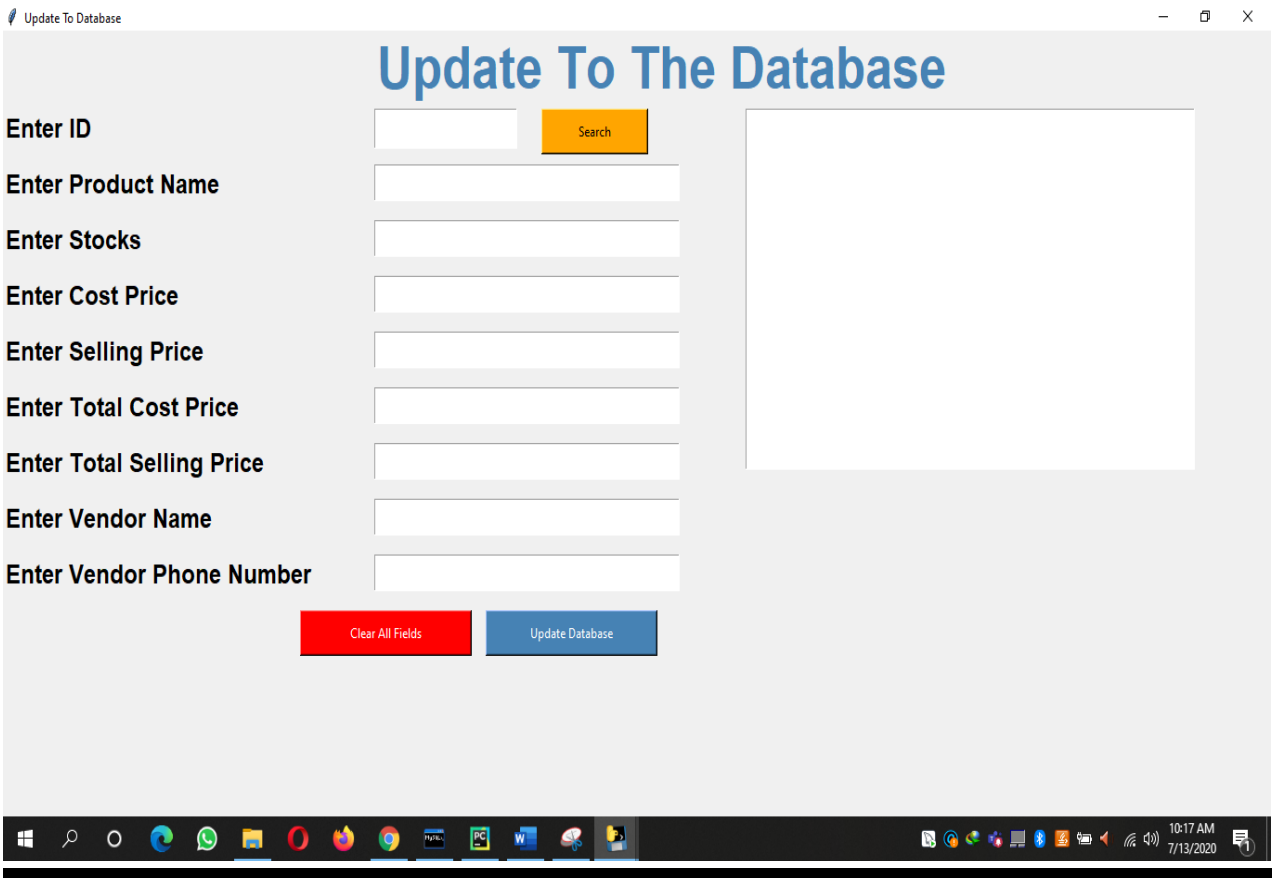


2. Adding Item's Information's to Database:

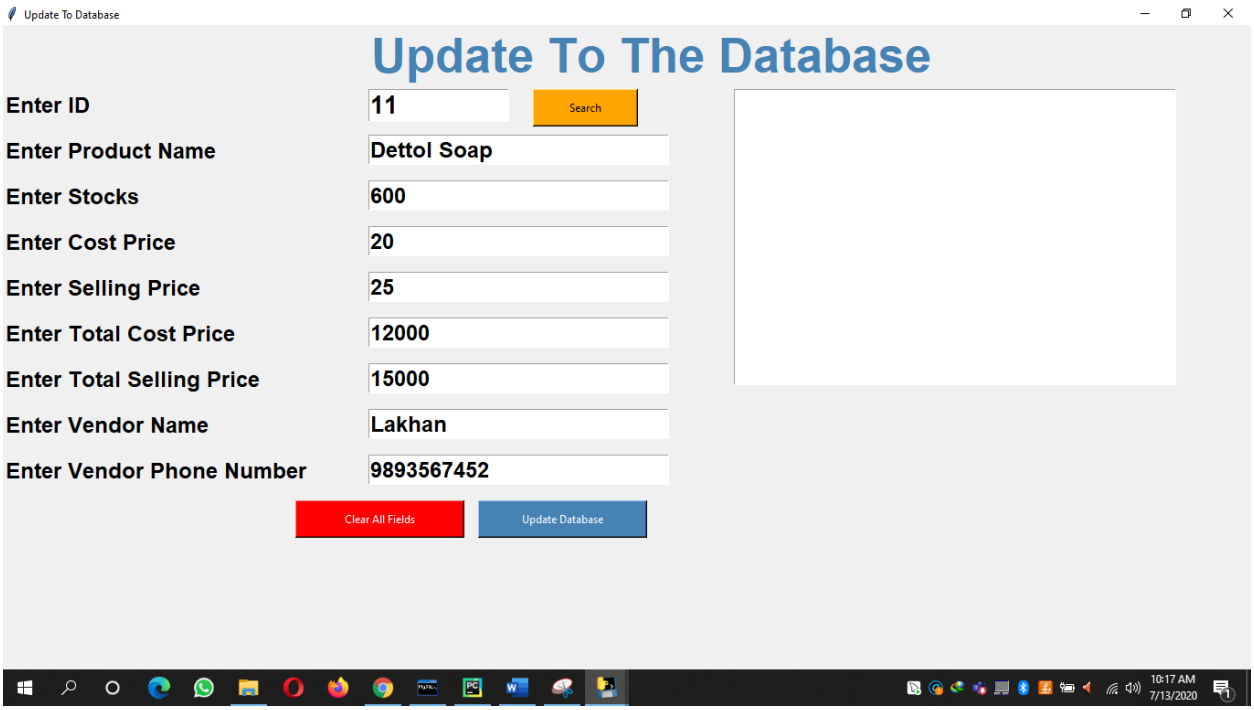


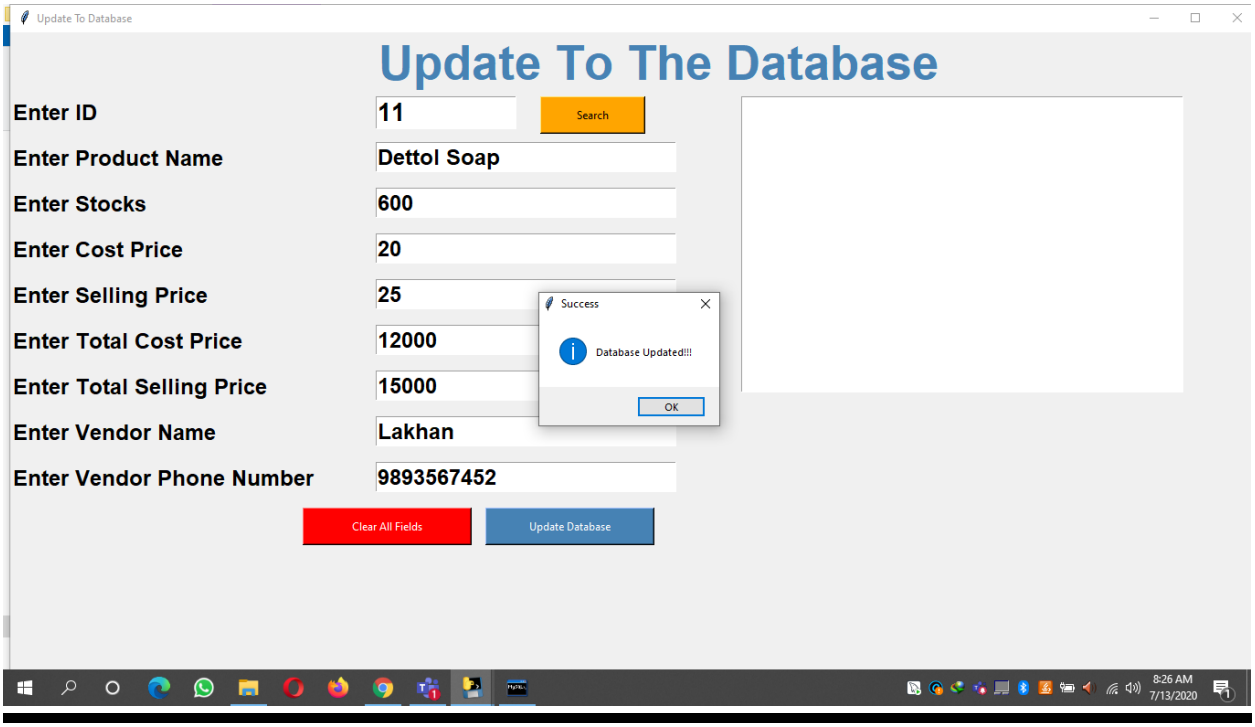


3. Form for Updating Item's Information's to Database:

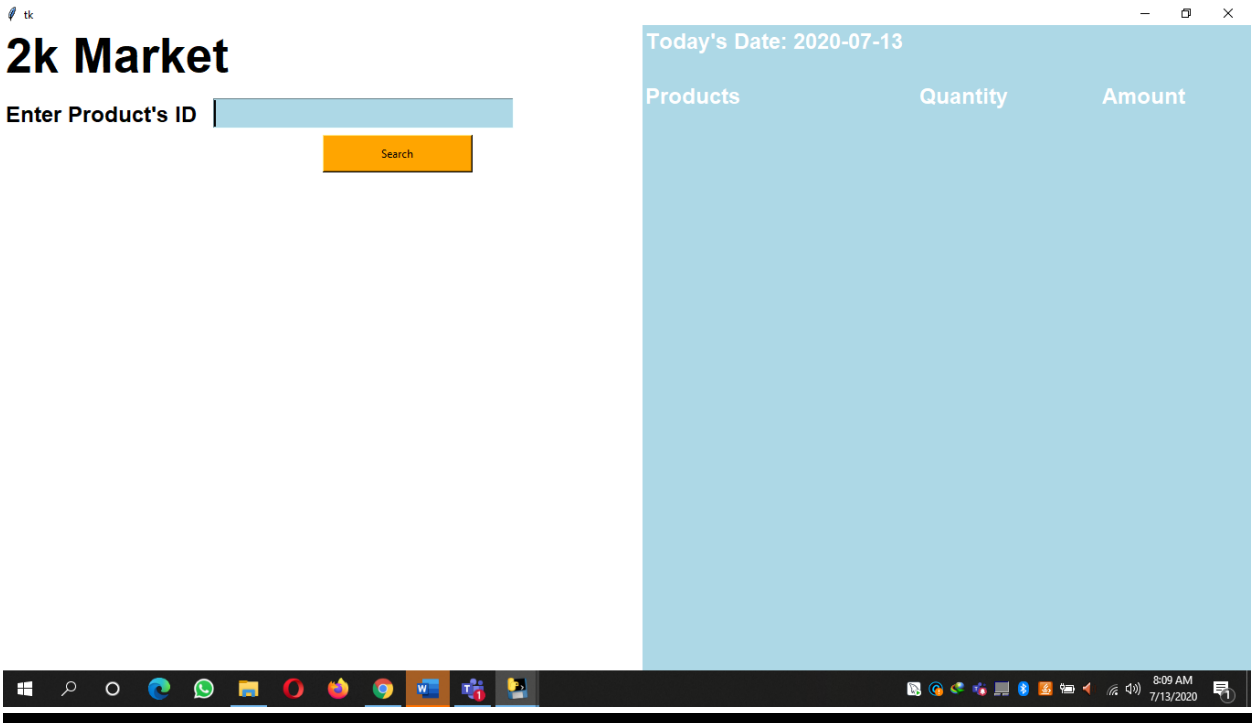


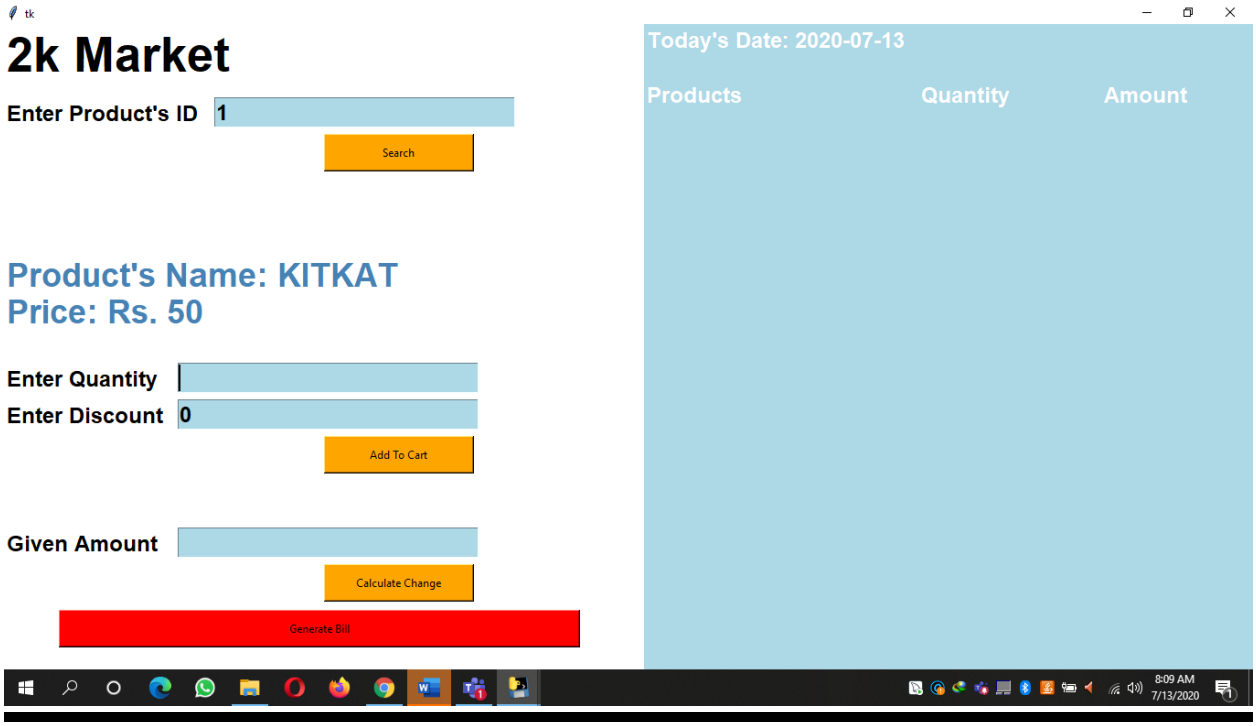
4. Updating Item's Information's to Database:

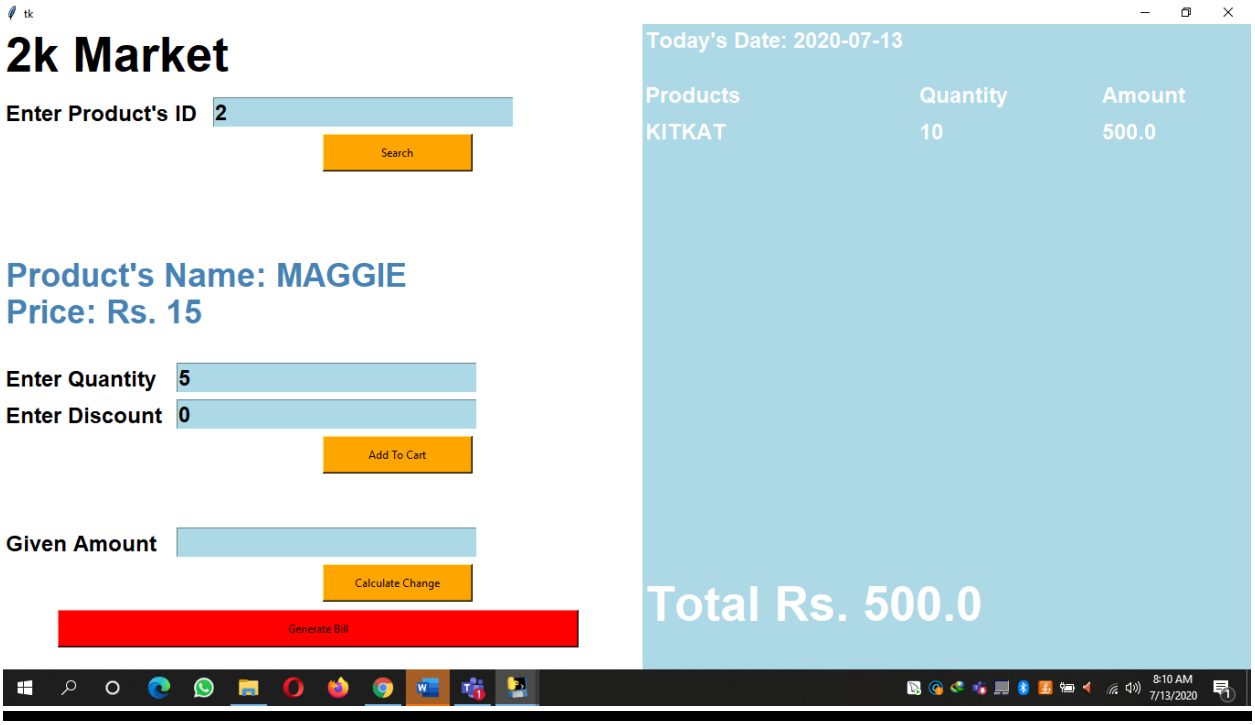


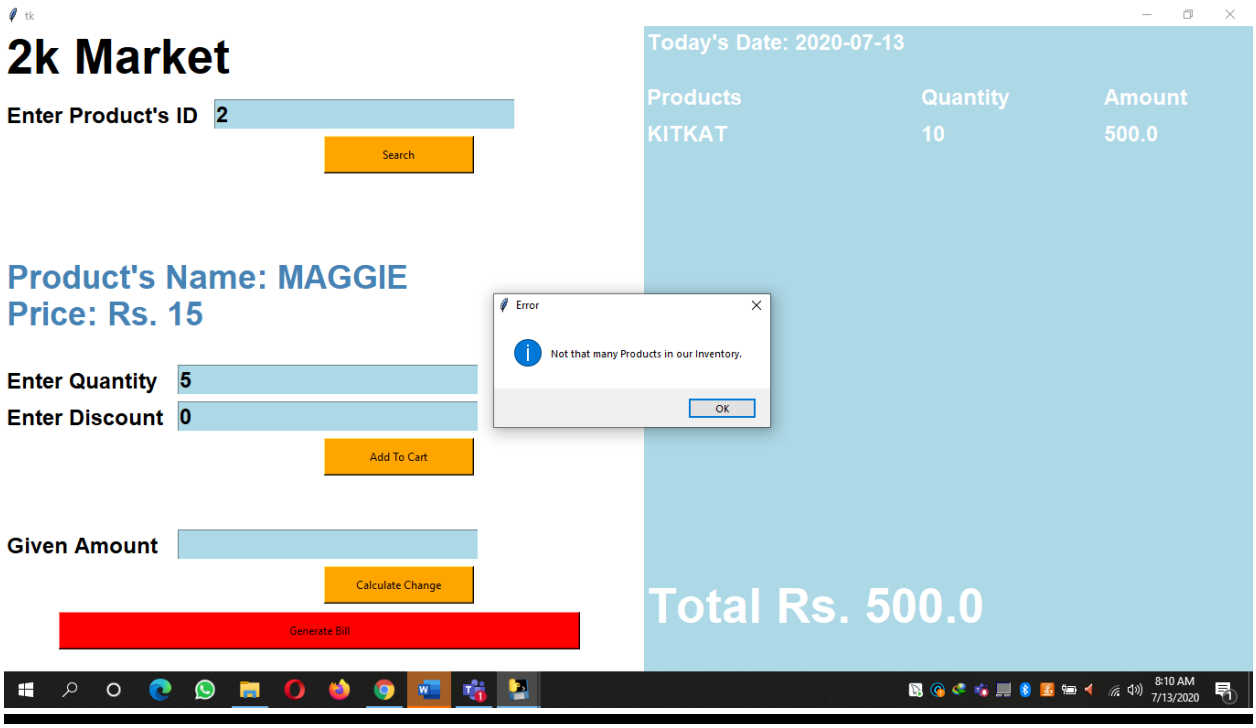


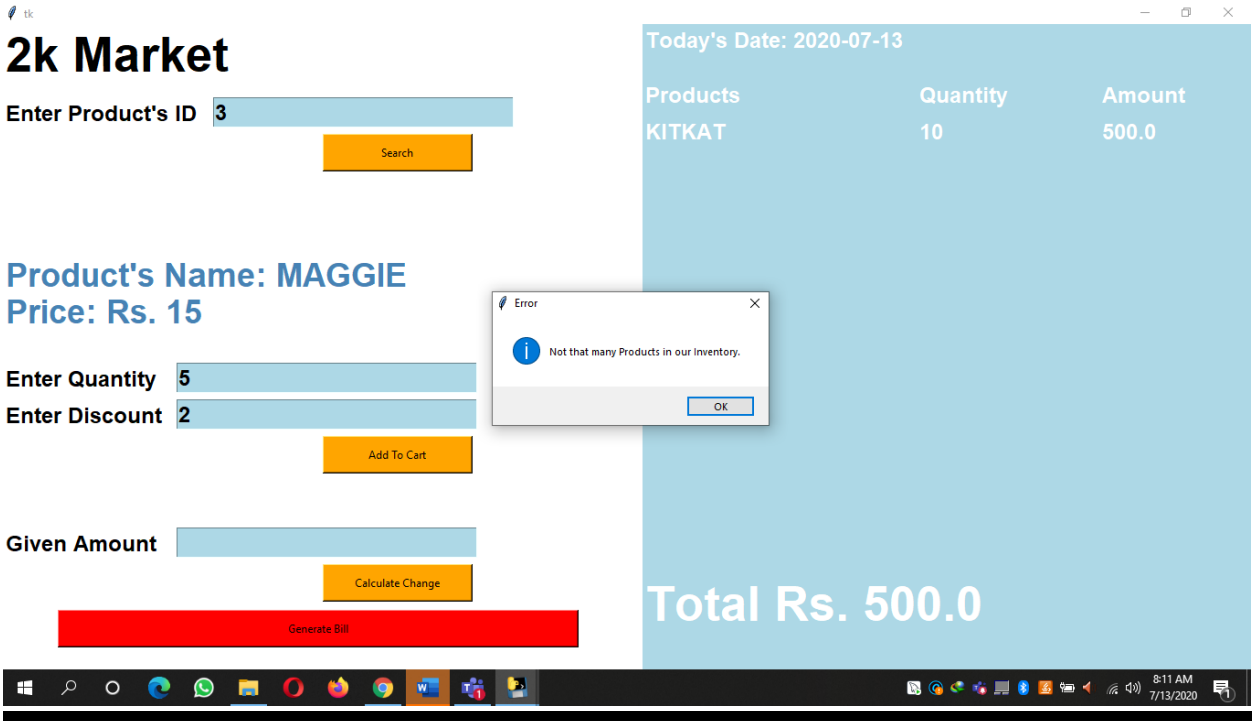
5. 2k Market Form:

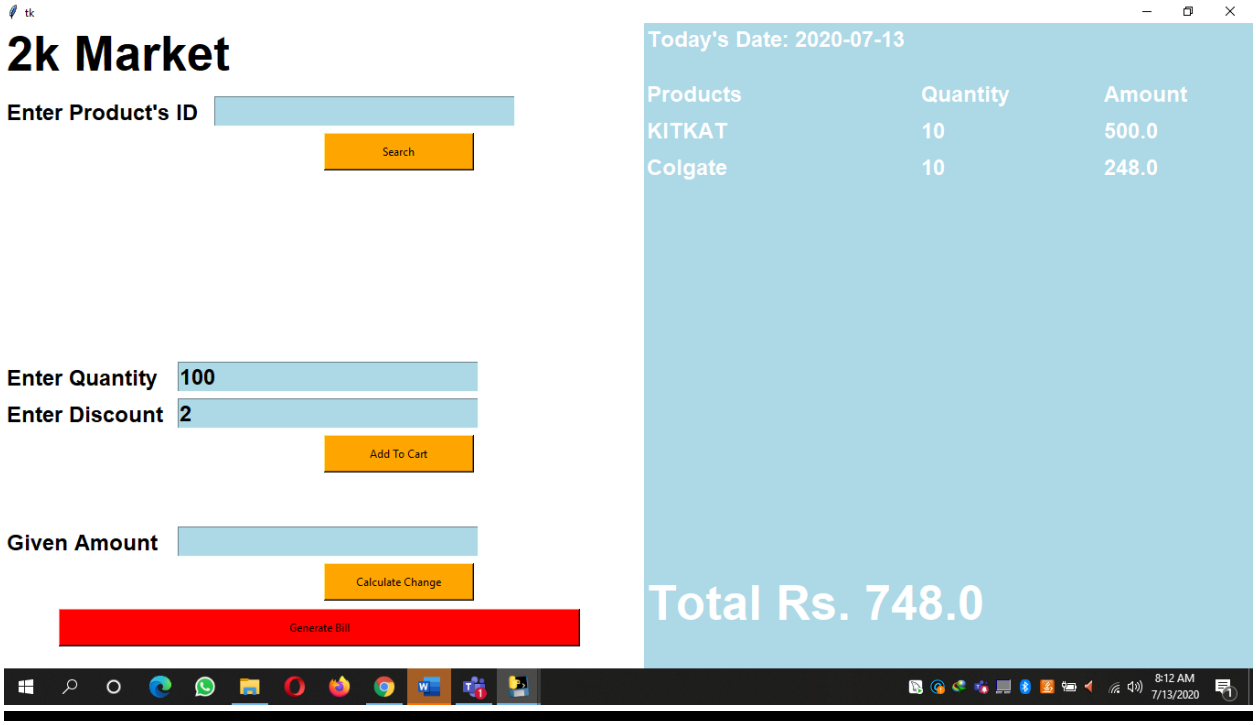


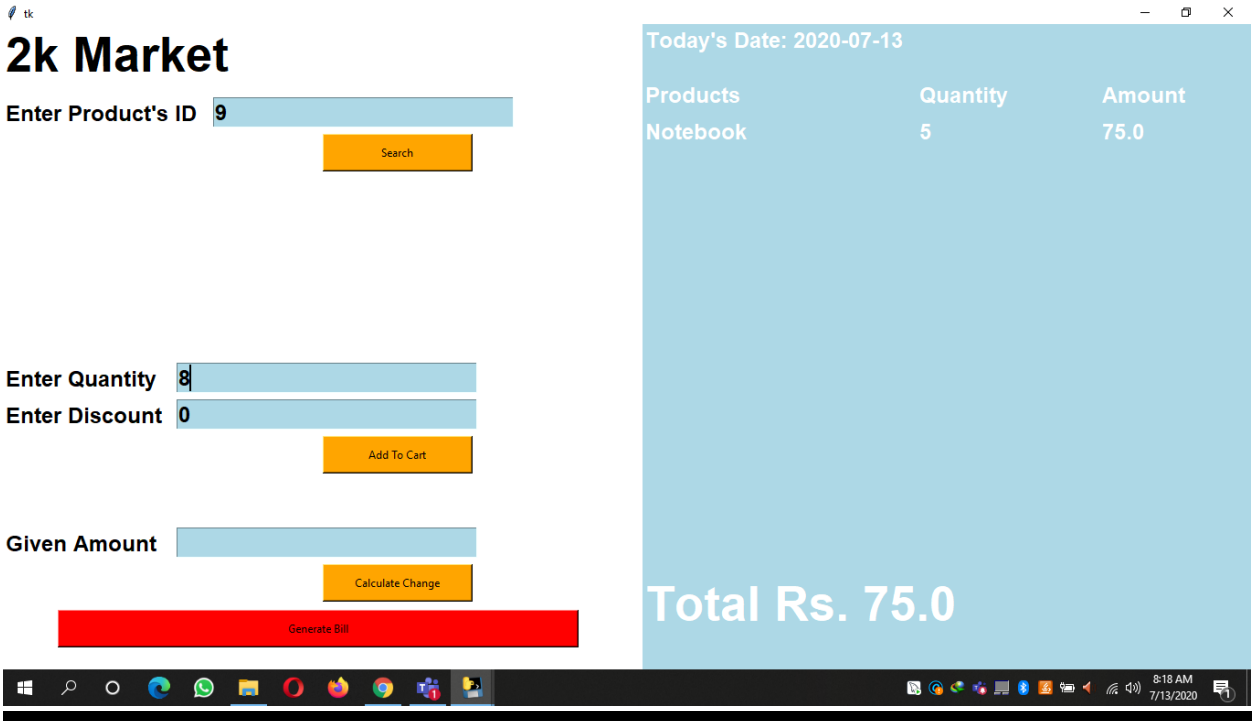


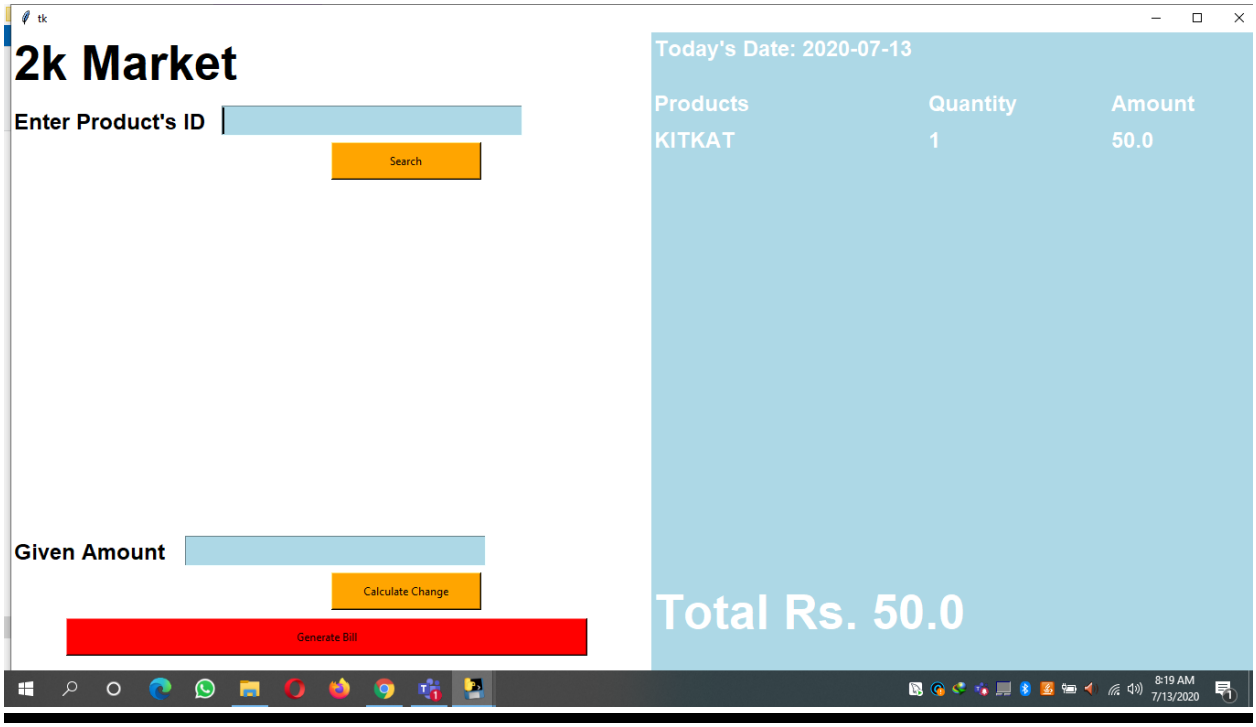

tk
2k Market
Enter Product's ID 2
Search
Product's Name: MAGGIE
Price: Rs. 15
Enter Quantity 5
Enter Discount 0
Add To Cart
Given Amount
Calculate Change
Generate Bill
Today's Date: 2020-07-13
Products Quantity Amount
KITKAT 10 500.0
Total Rs. 500.0
8:10 AM
7/13/2020

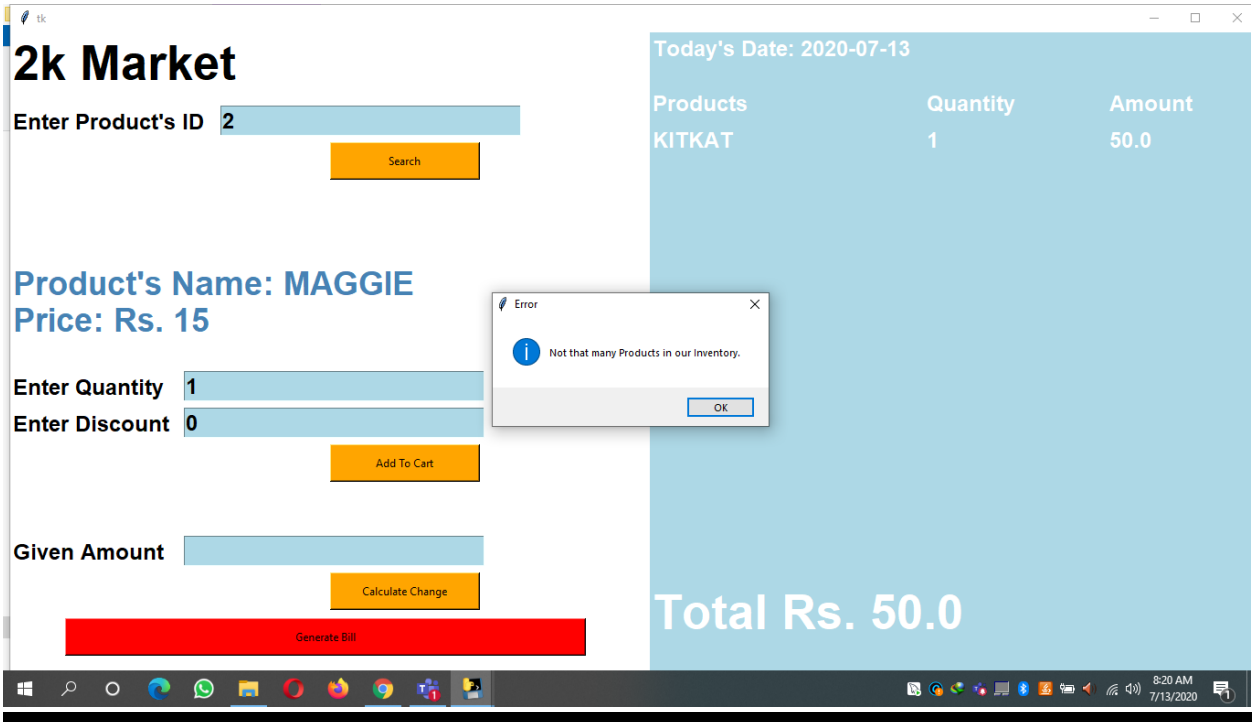

tk
2k Market
Enter Product's ID 2
Search
Product's Name: MAGGIE
Price: Rs. 15
Enter Quantity 5
Enter Discount 0
Add To Cart
Given Amount
Calculate Change
Generate Bill
Error
Not that many Products in our Inventory.
OK
Today's Date: 2020-07-13
Products Quantity Amount
KITKAT 10 500.0
Total Rs. 500.0
8:10 AM
7/13/2020

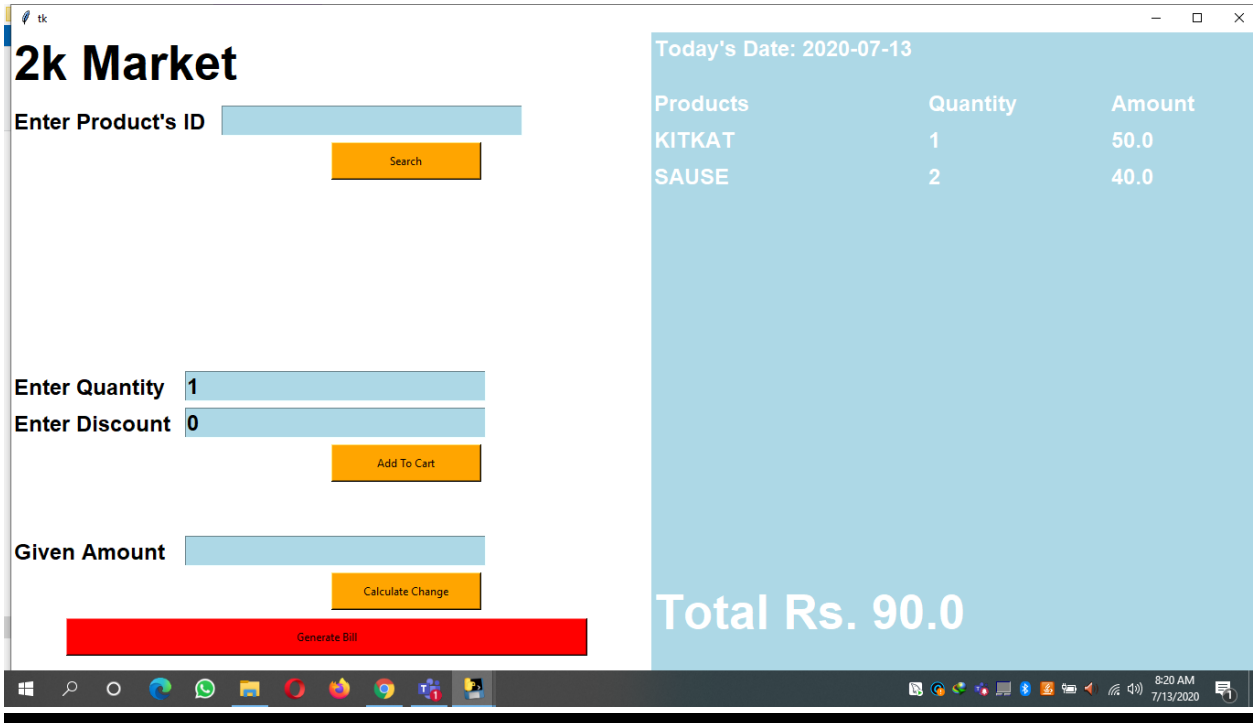

tk
2k Market
Enter Product's ID
3
Search
Product's Name: MAGGIE
Price: Rs. 15
Enter Quantity
5
Enter Discount
2
Add To Cart
Given Amount
Calculate Change
Generate Bill
Today's Date: 2020-07-13
Products
Quantity
Amount
KITKAT
10
500.0
Total Rs. 500.0
Error
Not that many Products in our Inventory.
OK
8:11 AM
7/13/2020

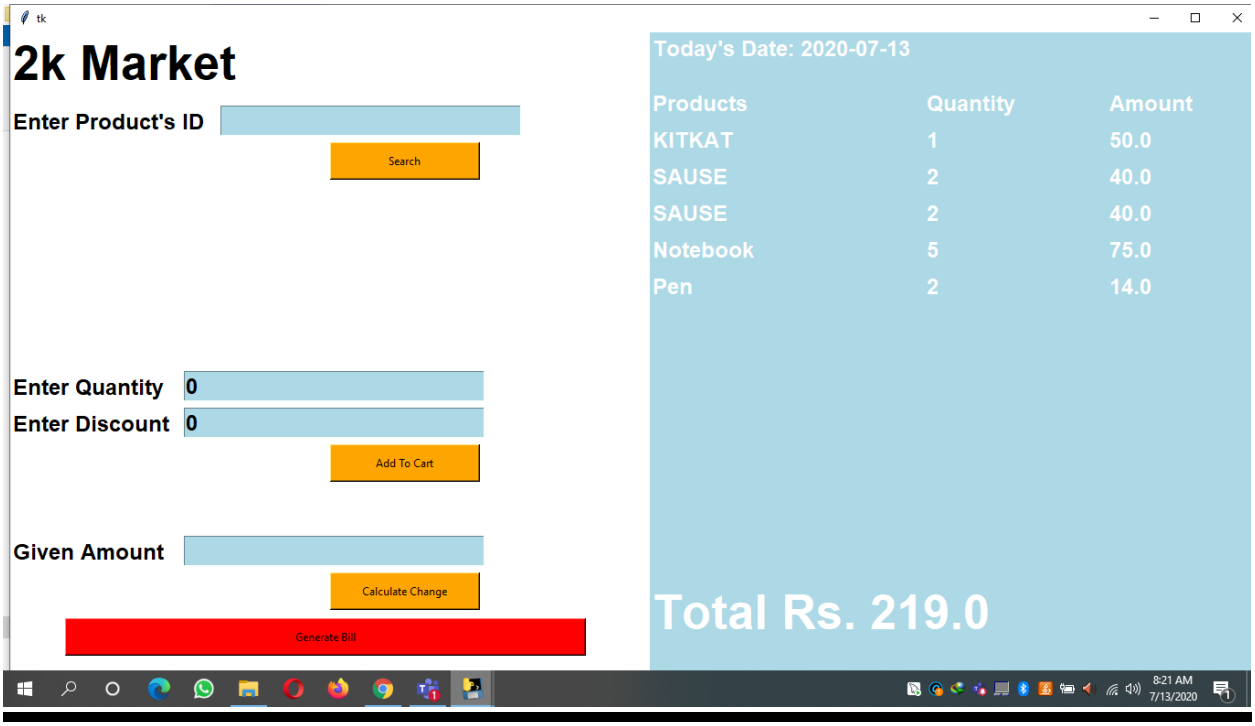

tk
2k Market
Enter Product's ID
Search
Enter Quantity
100
Enter Discount
2
Add To Cart
Given Amount
Calculate Change
Generate Bill
Today's Date: 2020-07-13
Products
Quantity
Amount
KITKAT
10
500.0
Colgate
10
248.0
Total Rs. 748.0
8:12 AM
7/13/2020

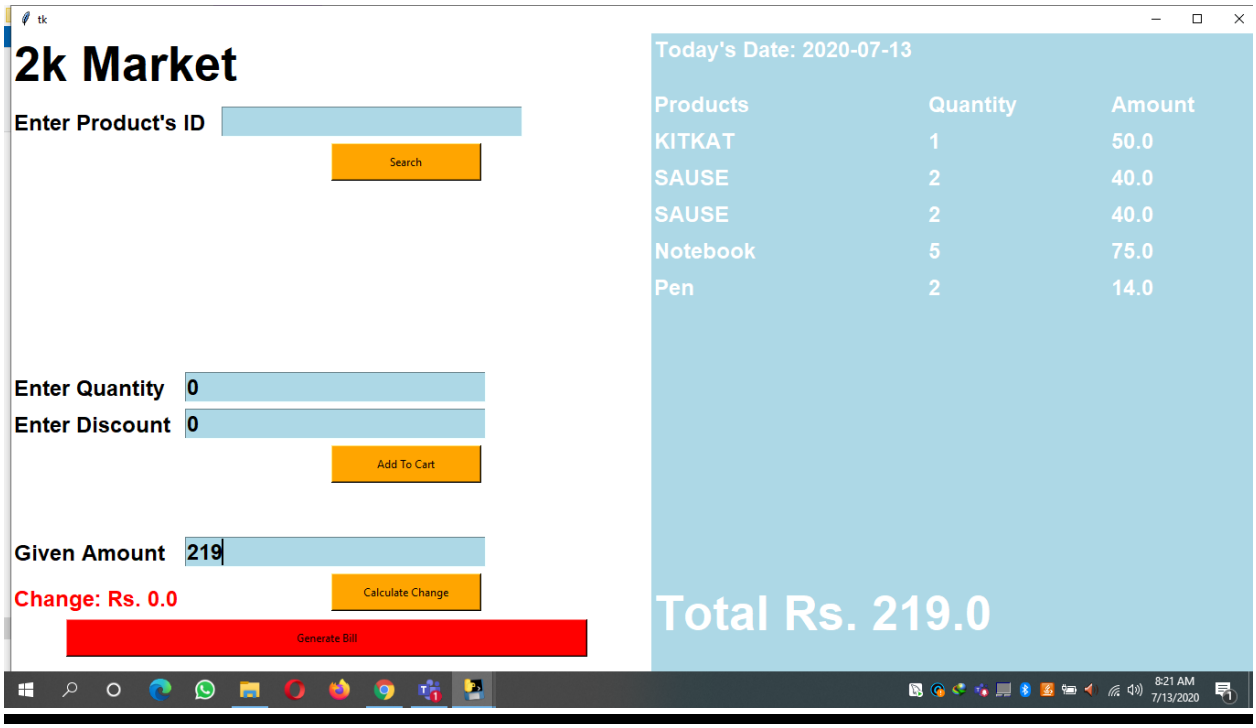

tk
2k Market
Enter Product's ID
9
Search
Enter Quantity
8
Enter Discount
0
Add To Cart
Given Amount
Calculate Change
Generate Bill
Today's Date: 2020-07-13
Products
Quantity
Amount
Notebook
5
75.0
Total Rs. 75.0
8:18 AM
7/13/2020

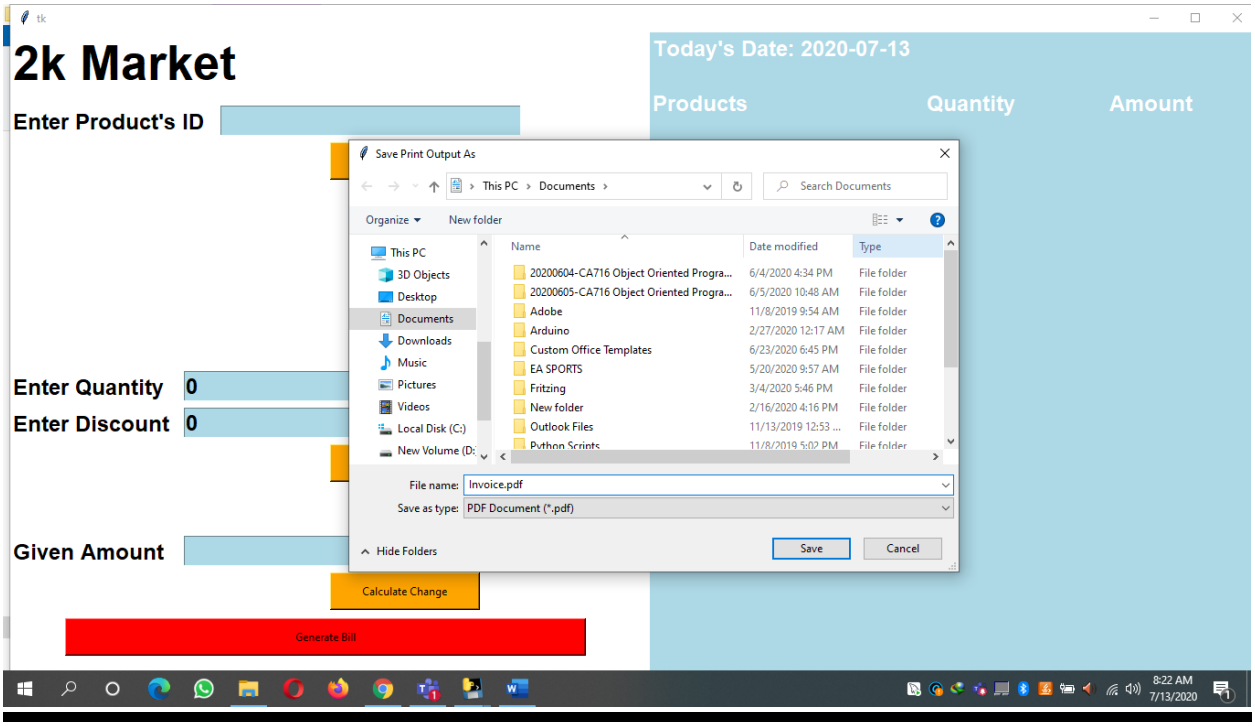

tk
2k Market
Enter Product's ID
Search
Given Amount
Calculate Change
Generate Bill
Today's Date: 2020-07-13
Products
Quantity
Amount
KITKAT
1
50.0
Total Rs. 50.0
8:19 AM
7/13/2020

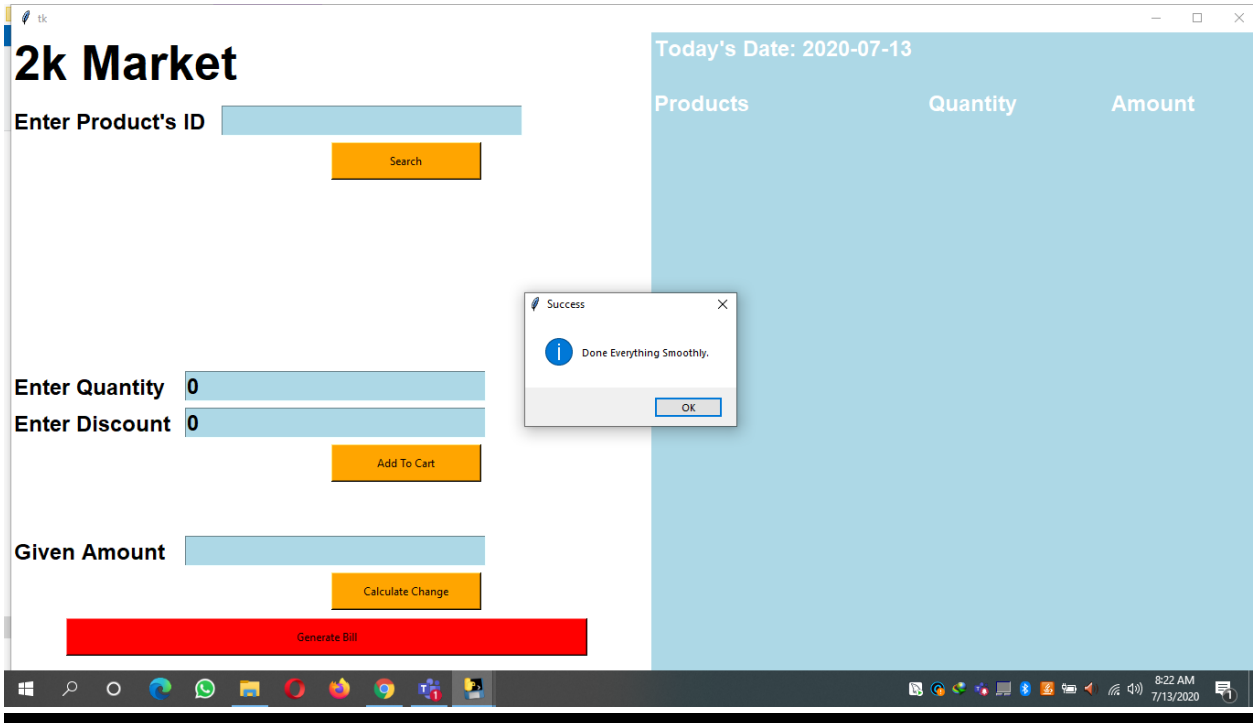

tk
2k Market
Enter Product's ID
2
Search
Product's Name: MAGGIE
Price: Rs. 15
Enter Quantity
1
Enter Discount
0
Add To Cart
Given Amount
Calculate Change
Generate Bill
Today's Date: 2020-07-13
Products
Quantity
Amount
KITKAT
1
50.0
Total Rs. 50.0
Error
Not that many Products in our Inventory.
OK
8:20 AM
7/13/2020


tk
2k Market
Enter Product's ID
Search
Enter Quantity
1
Enter Discount
0
Add To Cart
Given Amount
Calculate Change
Generate Bill
Today's Date: 2020-07-13
Products
Quantity
Amount
KITKAT
1
50.0
SAUSE
2
40.0
Total Rs. 90.0
8:20 AM
7/13/2020

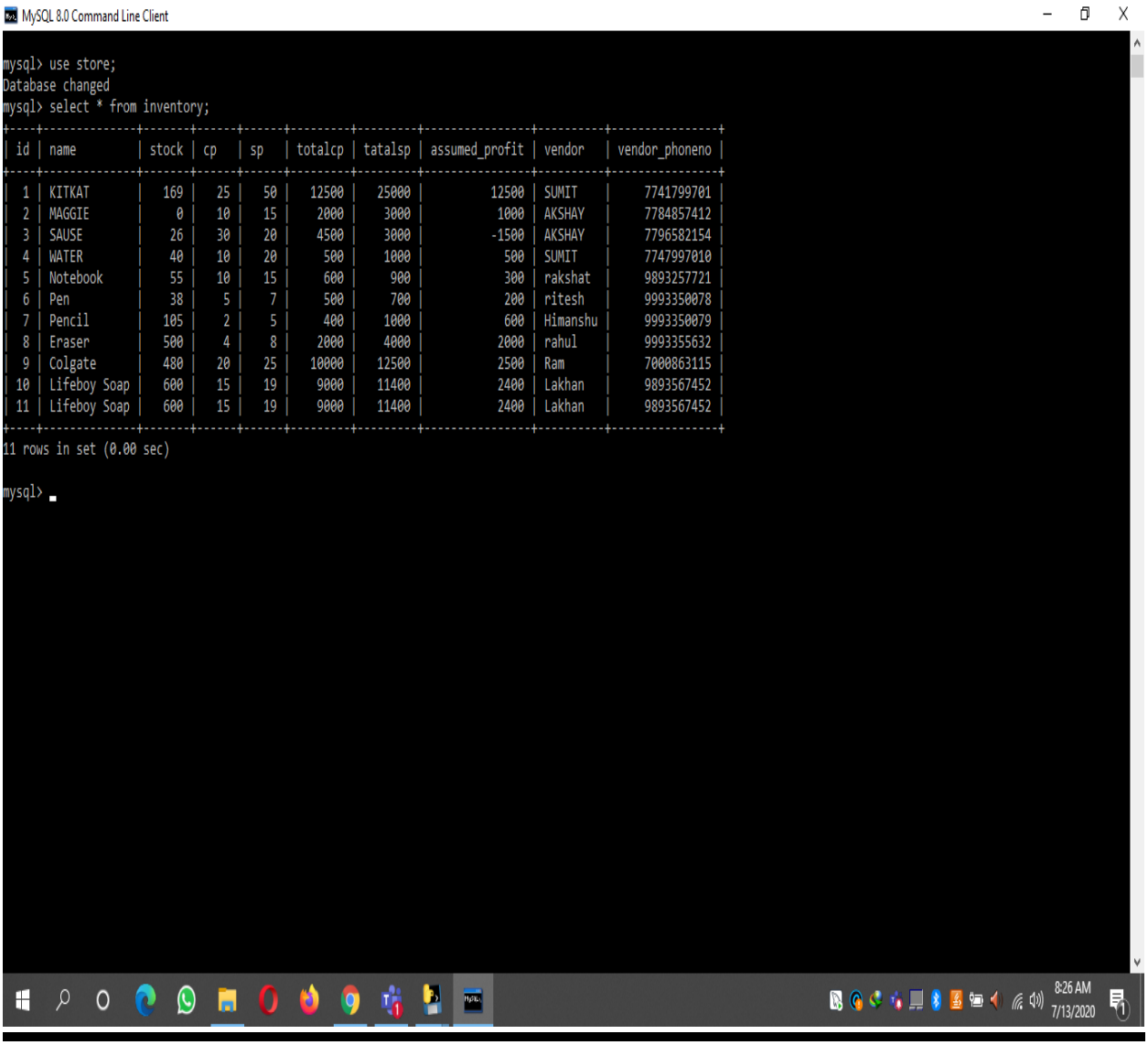

tk
2k Market
Enter Product's ID
Search
Enter Quantity 0
Enter Discount 0
Add To Cart
Given Amount
Calculate Change
Generate Bill
Today's Date: 2020-07-13
Products Quantity Amount
KITKAT 1 50.0
SAUSE 2 40.0
SAUSE 2 40.0
Notebook 5 75.0
Pen 2 14.0
Total Rs. 219.0
8:21 AM
7/13/2020


tk
2k Market
Enter Product's ID
Search
Enter Quantity 0
Enter Discount 0
Add To Cart
Given Amount 219
Change: Rs. 0.0
Calculate Change
Generate Bill
Today's Date: 2020-07-13
Products Quantity Amount
KITKAT 1 50.0
SAUSE 2 40.0
SAUSE 2 40.0
Notebook 5 75.0
Pen 2 14.0
Total Rs. 219.0
8:21 AM
7/13/2020

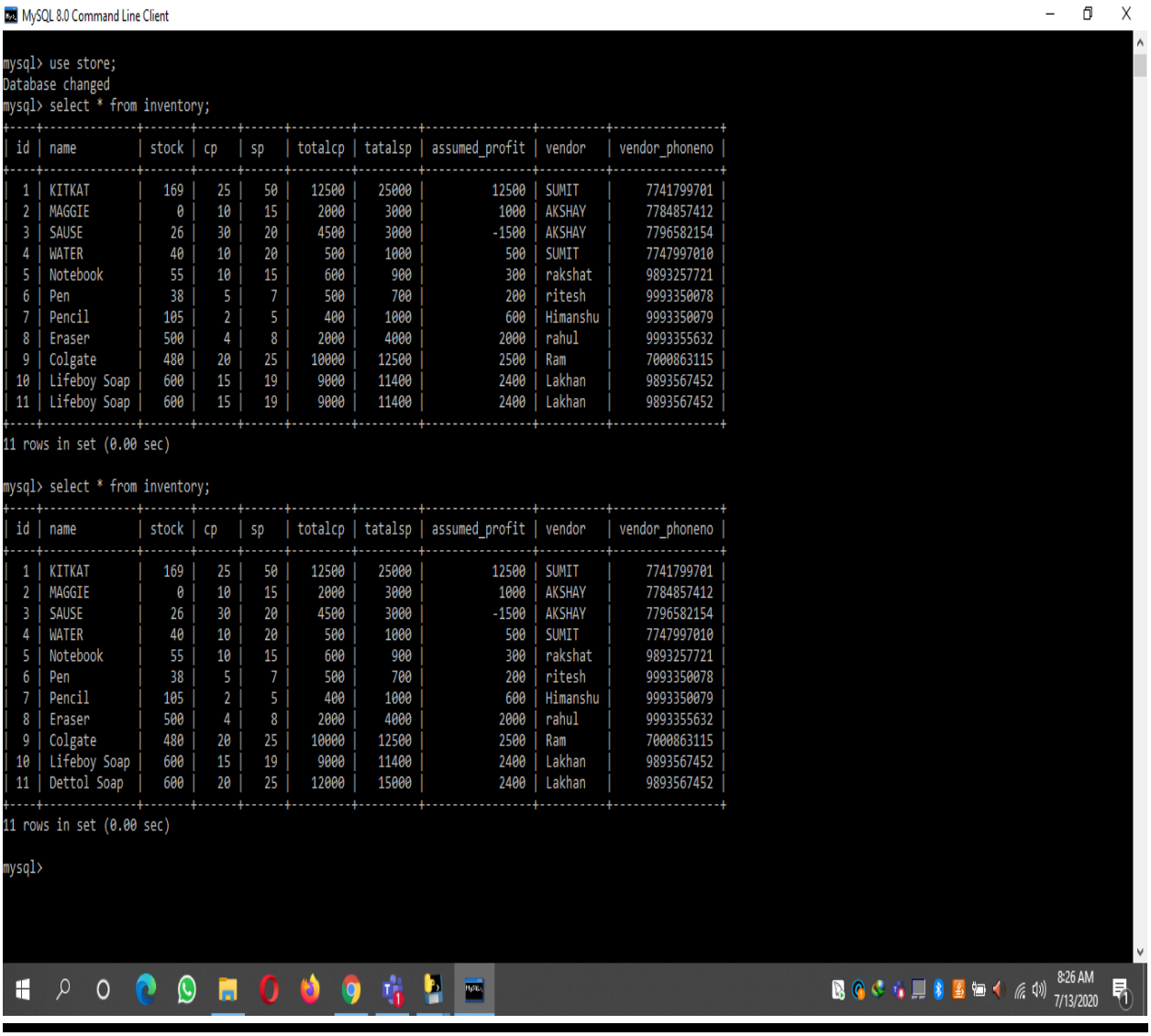

tk
2k Market
Today's Date: 2020-07-13
Products
Quantity
Amount
Enter Product's ID
Save Print Output As
This PC › Documents ›
Search Documents
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
New Volume (D:
Name
Date modified
Type
20200604-CA716 Object Oriented Progra... 6/4/2020 4:34 PM File folder
20200605-CA716 Object Oriented Progra... 6/5/2020 10:48 AM File folder
Adobe 11/8/2019 9:54 AM File folder
Arduino 2/27/2020 12:17 AM File folder
Custom Office Templates 6/23/2020 6:45 PM File folder
EA SPORTS 5/20/2020 9:57 AM File folder
Fritzing 3/4/2020 5:46 PM File folder
New folder 2/16/2020 4:16 PM File folder
Outlook Files 11/13/2019 12:53 ... File folder
Python Scripts 11/8/2019 5:02 PM File folder
File name: Invoice.pdf
Save as type: PDF Document (*.pdf)
Hide Folders
Save
Cancel
Enter Quantity 0
Enter Discount 0
Given Amount
Calculate Change
Generate Bill
8:22 AM
7/13/2020


tk
2k Market
Today's Date: 2020-07-13
Products
Quantity
Amount
Enter Product's ID
Search
Success
Done Everything Smoothly.
OK
Enter Quantity 0
Enter Discount 0
Add To Cart
Given Amount
Calculate Change
Generate Bill
8:22 AM
7/13/2020

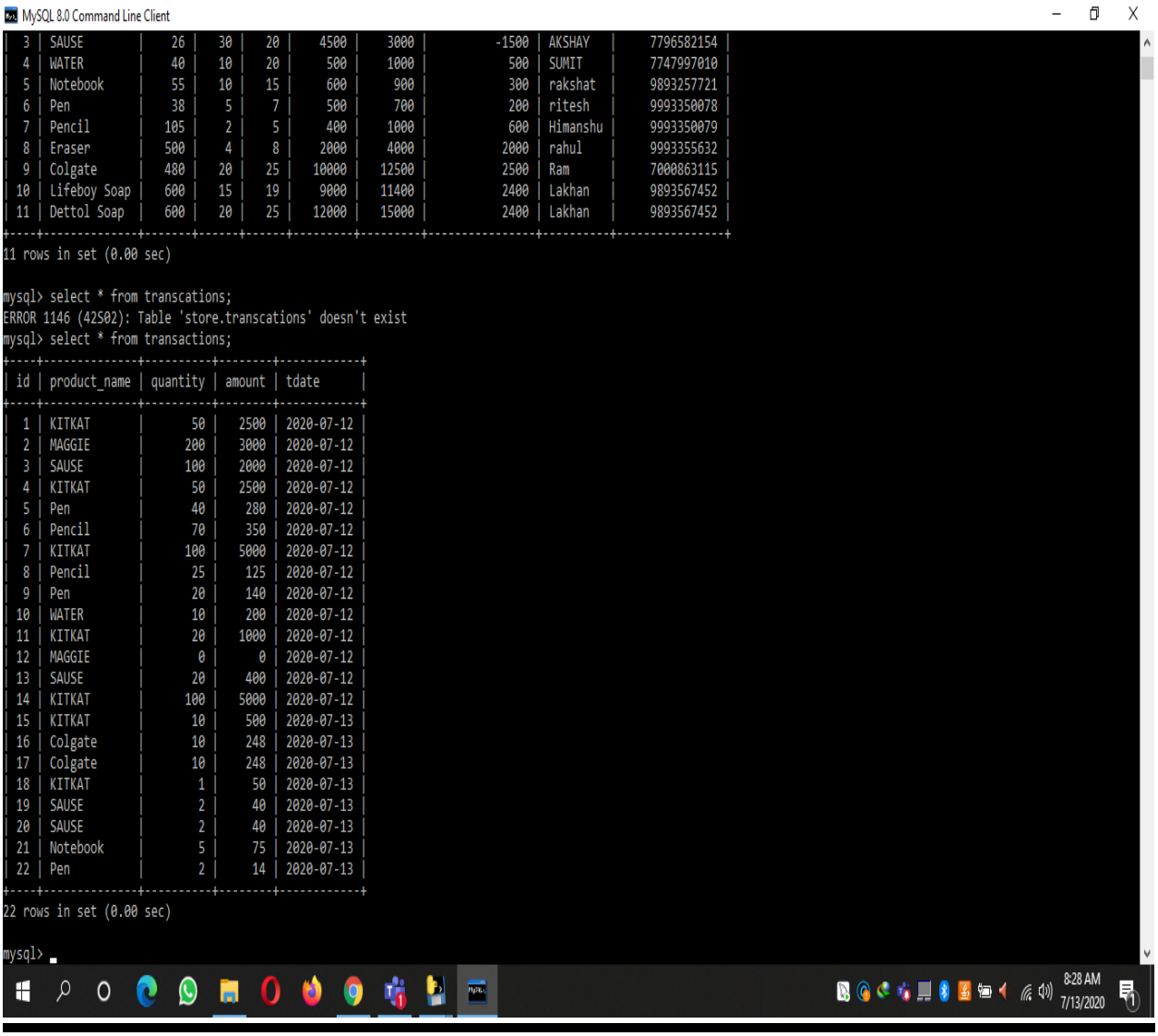
6. Inventory Table After Adding Data:



7. Inventory Table After Updating Data:



8. Transactions Table After Generating Bills:



9. Generated Invoice:

2K Market
National Institute of Technology, Tiruchirappalli
9876543210
Invoice
2020-07-13

SNo.	Products	Qty	Amount
1	KITKAT	5	250.0
2	SAUSE	2	40.0
3	WATER	2	40.0
4	Notebook	2	30.0
5	Pen	1	7.0
6	Pencil	5	25.0
7	Eraser	25	200.0
8	Colgate	13	325.0
9	Lifeboy So	5	95.0
10	Dettol Soa	5	125.0

Total Amount Rs. 1137.0
Thanks for Visiting.

References

- https://www.w3schools.com/sql/sql_intro.asp
- https://docs.python.org/3/library/tkinter.html
- https://stackoverflow.com/
- https://www.jetbrains.com/pycharm/
- https://www.youtube.com/
- https://www.tutorialspoint.com/python/python_gui_programming.htm
- https://www.tutorialspoint.com/dbms/index.htm
- https://www.javatpoint.com/dbms-tutorial
- https://realpython.com/python-gui-tkinter/

-Thank You-

--END--