

Thapar Institute of Engineering and Technology

Department of Computer Science and Engineering

Fashion Store Management System

Garvita Bhatnagar (102303538) Siddharth Malik (102303541) Kashvi Aggarwal (102303564) Siddhita Madan (102483082)

Supervisor: Mr. Abdul Kadir

A report submitted in partial fulfilment of the requirements for the degree of Bachelor of Technology in Computer Engineering

Declaration

Siddhita Madan (102483082), Garvita Bhatnagar (102303538), Kashvi Aggarwal (102303564) and Siddharth Malik (102303541), of the Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala, Punjab 147001, confirm that this is our work. Figures, tables, equations, code snippets, artworks, and illustrations are original unless explicitly acknowledged, quoted, or referenced. We understand that failing to do so will be considered a case of plagiarism, which is a form of academic misconduct and will be penalized accordingly. We consent for a copy of this report to be shared with future students as an exemplar and for the work to be made available more widely to TIET members and the public interested in teaching, learning, and research.

Siddhita Madan (102483082) Garvita Bhatnagar (102303538) Kashvi Aggarwal (102303564) Siddharth Malik (102303541)

April 28, 2025

Abstract

The Fashion Store Management System is designed to streamline the operations of a retail clothing business by providing efficient management of customer data, employee records, supplier inventories, sales, and purchase transactions. The primary objective of this project is to develop a lightweight backend system using Python, integrating MySQL for secure and reliable data storage. The system establishes multiple normalized relational tables to maintain data integrity and minimize redundancy, allowing for quick re-trieval and updates.

The project involves the development of modular Python scripts that connect to the MySQL database, automate the creation of tables, and offer interactive operations such as adding customers, recording sales, managing supplier inventory, and tracking purchases.

The results show that the system successfully handles all core backend functionalities expected from a retail management platform. It demonstrates effective transaction management, relational mapping, and normalization compliance to ensure data consistency.

In conclusion, the project achieves its aim of building a basic yet ex- tendable backend system for fashion store management, providing a strong foundation for future scaling or integration with a frontend user interface.

Keywords: Fashion Store Management, Backend Development, Python,

MySQL, Database Normalization

Word Count: 225 words

GitLab Link: https://github.com/SiddharthM2416/Clothing-Store-Inventory-

Management

Acknowledgements

We would like to express our heartfelt gratitude to all those who supported us during the completion of this project, "Clothing Store Management System."

First and foremost, we sincerely thank our project supervisor, Mr. Abdul Kadir, for his invaluable guidance, encouragement, and insightful suggestions throughout the project work.

We also extend our gratitude to the Department of Computer Science Engineering and Thapar Institute of Engineering and Technology for providing the necessary facilities and a supportive learning environment.

A special thanks to our friends and family for their continuous encouragement and motivation during this journey. Without the support, advice, and facilities provided by these individuals and institutions, the successful completion of this project would not have been possible.

Contents

1	Problem Statement 7 1.1 Summary	7
2	Entity Relationship Model of the Project 8 2.1 Summary	
3	Methodology 9 3.1 Summary	9
4	E-R Model to Relational Model 10 4.1 Summary	10
5	Normalized Tables and Discussion 11 5.1 Summary	11
6	PL/SQL Snapshots 17	
7	SQL/PL/SQL Outputs and Discussion 24	

List of Figures

Figure No.	Title	Page No.
Fig 1.1	E-R Diagram of Fashion Store Management System	2
Fig 1.2	Relational Table - CUSTOMER	6
Fig 1.3	Relational Table - SUPPLIER	7
Fig 1.4	Relational Table - GARMENTS	7
Fig 1.5	Relational Table - EMPLOYEE	8
Fig 1.6	Relational Table - SALES	9
Fig 1.7	Relational Table - PURCHASE	9
Fig 1.8	Normalization Diagram (up to 3NF)	10

List of Tables

Table Name	Description
CUSTOMER	Stores customer information like name, address, and contact details.
SUPPLIER	Stores supplier details including contact and material provided.
GARMENTS	Contains information about garments like type, quantity, price, and supplier.
EMPLOYEE	Stores employee records including name, address, age, salary, and commission.
SALES	Records customer purchases handled by employees along with transaction amounts.
PURCHASE	Records purchases made from suppliers, including garments, price, and quantity.

List of Abbreviations

Abbreviation	Full Form
DBMS	Database Management System
SQL	Structured Query Language
ER	Entity-Relationship
PK	Primary Key
FK	Foreign Key
CRUD	Create, Read, Update, Delete
GUI	Graphical User Interface (if used)
MySQL	My Structured Query Language
ID	Identification Number

Chapter 1

Problem Statement

1.1 Summary

The fashion industry has grown rapidly, resulting in the need for efficient management of customers, suppliers, employees, garments, sales, and purchase operations. Traditional paper-based management systems are outdated, prone to errors, and inefficient in handling bulk operations.

The main objectives of the Fashion Store Management System are:

- To store and manage customer details such as name, address, and contact information.
- To manage suppliers and the materials they provide.
- To maintain an inventory of garments, including their type, quantity, and price.
- To manage employee information, including their salaries and commissions.
- To record sales transactions between employees and customers.
- To maintain purchase records from suppliers for inventory replenishment.

The system will use a relational database management system (RDBMS) approach to manage all the information efficiently.

Chapter 2 Entity Relationship Model of the Project

2.1 Summary

The ER diagram models a Garment Business Management System including entities:

- Customer (custid, name, contact, address)
- Employee (empid, ename, address, age, salary, comm)
- Supplier (sid, sname, address, contact, material)
- Garments (gid, type, price, sid)
- Sales (saleid, amt, custid, empid)
- Purchase (pid, price, qty, gid, sid)

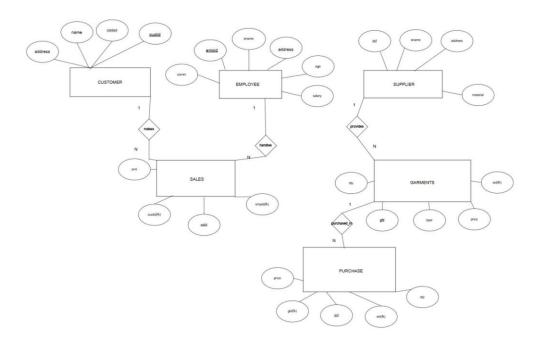
Relationships:

- An Employee handles many Sales.
- A Customer makes many Sales.
- A Supplier provides many Garments.
- Garments are purchased through Purchase transactions.

Chapter 2.1

Entity Relationship Model of the Project

Diagram



Chapter 3 Methodology

3.1 Summary

- 1. Requirement Analysis: Identified the core requirements.
- 2. System Design: Designed an E-R Diagram and relational schema.
- 3. Database Normalization: Structured according to 1NF, 2NF, and 3NF.
- 4. Database Implementation: Created tables using MySQL.
- 5. Backend Development: Developed Python scripts with MySQL connector.
- 6. Testing and Validation: CRUD operations were performed and tested.
- 7. Final Review and Documentation: Prepared final report.

Chapter 4

E-R Model to Relational Model

4.1 Summary

Sample table creation scripts:

Customer= "CREATE TABLE IF NOT EXIST
CUSTOMER(custid int auto_increment,
name varchar(20),
address varchar(15),
contact char(10),
primary key(custid,contact));"

1. CUSTOMER

Column Name	Data Type	Constraints
custid	INT	AUTO_INCREMENT, PK
name	VARCHAR(20)	
address	VARCHAR(15)	
contact	CHAR(10)	PK

Primary Key → (custid , contact) (composite key)

supplier="CREATE TABLE if not exists SUPPLIER(sid int auto_increment primary key, sname varchar(20), address varchar(15), material varchar(10));"

2. SUPPLIER

Column Name	Data Type	Constraints
sid	INT	AUTO_INCREMENT, PK
sname	VARCHAR(20)	
address	VARCHAR(15)	
material	VARCHAR(10)	

garments="CREATE TABLE if not exists GARMENTS(
gid int auto_increment primary key,
type varchar(20),
qty int,
price int,
sid int references supplier.sid);"

3. GARMENTS

Column Name	Data Type	Constraints
gid	INT	AUTO_INCREMENT, PK
type	VARCHAR(20)	
qty	INT	
price	INT	
sid	INT	$FK \rightarrow SUPPLIER(sid)$

employee="CREATE TABLE if not exists EMPLOYEE(empid int auto_increment primary key, ename varchar(20), address varchar(15), age int, salary int, comm decimal(2,2));"

4. EMPLOYEE

Column Name	Data Type	Constraints
empid	INT	AUTO_INCREMENT, PK
ename	VARCHAR(20)	
address	VARCHAR(15)	
age	INT	
salary	INT	
comm	DECIMAL(2,2)	

sales="CREATE TABLE if not exists SALES(salid int auto_increment primary key, custid int references customer.custid, empid int references employee.empid, amt int);"

5. SALES

Column Name	Data Type	Constraints
salid	INT	AUTO_INCREMENT, PK
custid	INT	$FK \rightarrow CUSTOMER(custid)$
empid	INT	$FK \rightarrow EMPLOYEE(empid)$
amt	INT	

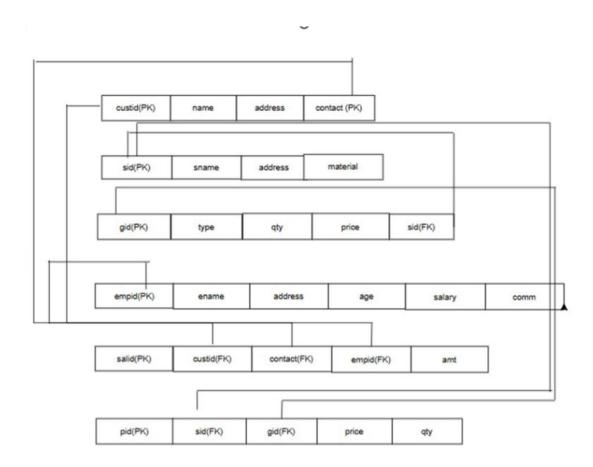
purchase="CREATE TABLE if not exists PURCHASE(\ pid int auto_increment primary key,\ sid int references supplier.sid,\ gid int references garments.gid,\ price int,\ qty int);"

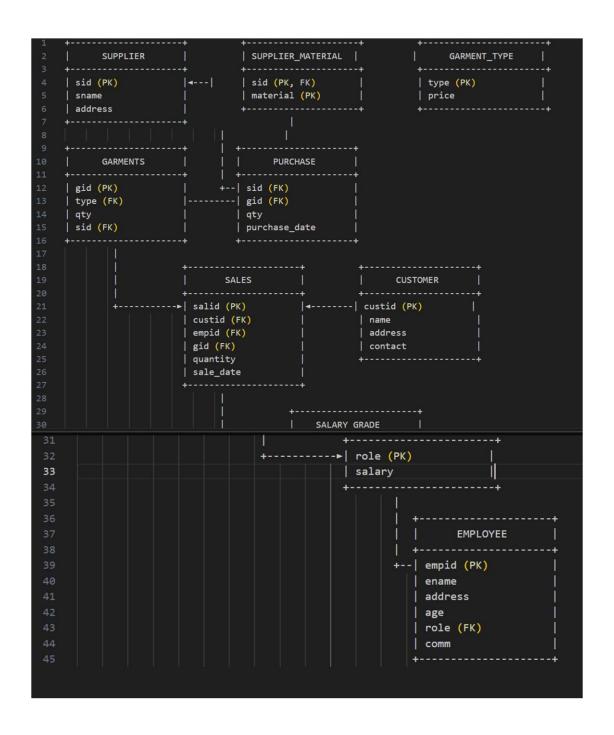
6. PURCHASE		
Column Name	Data Type	Constraints
pid	INT	AUTO_INCREMENT, PK
sid	INT	$FK \rightarrow SUPPLIER(sid)$
gid	INT	$FK \rightarrow GARMENTS(gid)$
price	INT	
qty	INT	

Chapter 5 Normalized Tables and Discussion

5.1 Summary

The database has been normalised up to Third Normal Form (3nf). All tables have atomic attributes (1NF), full functional dependency on the primary key (2NF), and no transitive dependencies (3NF). Foreign keys maintain referential integrity across related tables. As a result, the database structure avoids redundancy, ensures data consistency, and prevents update, insert, and delete anomalies.





Chapter 6 PL/SQL Snapshots

```
import mysql.connector
mydb=mysql.connector.connect(host='localhost',user='root'\
                             ,passwd='admin',auth plugin="mysql native password")
mycursor=mydb.cursor()
mycursor.execute("USE cltstr;")
#To Add a new customer
def addcust(name,address,contact):
    val=(name, address, contact)
    ins="INSERT INTO customer(name,address,contact) VALUES (%s,%s,%s);"
    mycursor.execute(ins, val)
    mydb.commit()
#To add a new supplier
def addsupp(name, address, material):
   val=(name, address, material)
   ins="INSERT INTO supplier(sname,address,material) VALUES (%s,%s,%s);"
   mycursor.execute(ins, val)
   mydb.commit()
#To add a new type of Garment
def addgarment(t,qty,price,sid):
   a=(t,qty,price,sid)
   mycursor.execute("INSERT INTO garments(type,qty,price,sid) VALUES(%s,%s,%s,%s);", a)
   mydb.commit()
#To add a new employee
def addemp(name,address,age,sal,comm):
    ins="INSERT INTO employee (ename, address, age, salary, comm) VALUES (%s, %s, %s, %s, %s);"
    val=(name, address, age, sal, comm)
    mycursor.execute(ins, val)
   mydb.commit()
#To record new sales
def newsale(11, contact, empid):
    mycursor.execute("SELECT gid FROM garments;")
    12 = mycursor.fetchall()
    11 = [i for i in l1 if (i,) in l2] # Validate garment IDs
    mycursor.execute("SELECT contact FROM customer;")
    c = mycursor.fetchall()
    if (contact,) not in c:
        raise Exception("Customer not found. Please add customer first.")
    mycursor.execute("SELECT empid FROM employee;")
    e = mycursor.fetchall()
    if (empid,) not in e:
        raise Exception("Invalid Employee ID.")
    mycursor.execute("SELECT custid FROM customer WHERE contact = %s;", (contact,))
    cid = mycursor.fetchone()[0]
```

```
amt = 0
  for i in l1:
     mycursor.execute("SELECT price FROM garments WHERE gid = %s;", (i,))
     p = mycursor.fetchone()[0]
     amt += p
     mycursor.execute("UPDATE garments SET qty = qty - 1 WHERE gid = %s;", (i,))
     mydb.commit()
  vals = (cid, empid, amt)
  q = "INSERT INTO sales(custid, empid, amt) VALUES (%s, %s, %s);"
 mycursor.execute(q, vals)
  mydb.commit()
#To record a new purchase from suppliers to store
def newpurchase(sid, gid, price, qty):
   mycursor.execute("SELECT sid FROM supplier;")
   s = mycursor.fetchall()
   if (sid,) not in s:
       raise Exception("Invalid Supplier ID.")
   mycursor.execute("SELECT gid FROM garments WHERE sid = %s;", (sid,))
   g = mycursor.fetchall()
if (gid,) not in g:
       raise Exception("Invalid Garment ID for given Supplier.")
   ins = "INSERT INTO purchase(sid, gid, price, qty) VALUES (%s, %s, %s, %s);"
   val = (sid, gid, price, qty)
   mycursor.execute(ins, val)
   mydb.commit()
   mycursor.execute("UPDATE garments SET qty = qty + (%s) WHERE gid = (%s);", (qty, gid))
   mydb.commit()
def nets():#To check Total sales
    mycursor.execute("select sum(amt) from sales;")
    amt=mycursor.fetchone()[0]
    return amt
def netp():#To check Total purchases
    mycursor.execute("select sum(price) from purchase;")
    amt=mycursor.fetchone()[0]
    return amt
def allsales(): #To open a list of all sales
    mycursor.execute("Select sales.*, employee.ename from sales JOIN \
employee ON sales.empid=employee.empid;")
    s=mycursor.fetchall()
    return s
def allpurchase(): #To open a list of all purchases
    mycursor.execute("Select purchase.*, supplier.sname from purchase \
JOIN supplier ON purchase.sid=supplier.sid;")
    s=mycursor.fetchall()
    return s
def rememp(empid): #To remove an employee
    mycursor.execute("delete from employee where empid=(%s);", (empid,))
    mydb.commit()
```

```
Frontend
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import tables
import processes # Backend connection
root = tk.Tk()
root.title("Clothing Store Management System")
root.geometry("800x600")
root.config(bq="white")
# Function to clear the frame
def clear frame():
        for widget in root.winfo children():
                widget.destroy()
# Home Page
def home():
    clear_frame()
title = tk.Label(root, text="Clothing Store Management", font=("Arial", 24, "bold"), bg="white")
    title.pack(pady=20)
   button_frame = tk.Frame(root, bg="white")
button_frame.pack()
    options = [
        ions = [
("Add Customer", add_customer_page),
("Add Employee", add_employee_page),
("Add Supplier", add_supplier_page),
("Add Garment", add_garment_page),
("New Sale", new_sale_page),
("New Purchase", new_purchase_page),
("Net Sales", view_net_sales),
("Net Purchases", view_net_purchases),
("View All Sales", view_all_sales),
("View All Purchases", view_all_purchases),
("Semove Employee", remove employee_page).
         ("Remove Employee", remove_employee_page),
    ]
    for idx, (text, command) in enumerate(options):
        btn = tk.Button(button_frame, text=text, command=command, width=20, height=2,\bg="lightblue", font=("Arial", 12))
        btn.grid(row=idx//2, column=idx%2, padx=20, pady=10)
# Go to Home Button
def go_home_button():
   home_btn = tk.Button(root, text="Go to Home", command=home, bg="orange", font=("Arial", 12, "bold"))
   home_btn.pack(pady=10)
# Add Customer Page
def add_customer_page():
    clear_frame()
    tk.Label(root, text="Add Customer", font=("Arial", 20), bg="white").pack(pady=20)
    name = tk.Entry(root, width=30)
   address = tk.Entry(root, width=30)
contact = tk.Entry(root, width=30)
    tk.Label(root, text="Name", font=("Arial", 12), bg="white").pack()
    tk.Label(root, text="Address", font=("Arial", 12), bg="white").pack()
    address.pack()
    tk.Label (root, text="Contact (10 digits)", font=("Arial", 12), bg="white").pack()
    contact.pack()
```

```
def submit():
            try:
        except Exception as e:
             messagebox.showerror("Error", str(e))
    tk.Button(root, text="Submit", command=submit, bq="lightqreen", font=("Arial", 12)).pack(pady=20)
    go_home_button()
# Add Employee Page
def add employee page():
    clear frame()
    tk.Label(root, text="Add Employee", font=("Arial", 20), bg="white").pack(pady=20)
    name = tk.Entry(root, width=30)
    address = tk.Entry(root, width=30)
    age = tk.Entry(root, width=30)
    salary = tk.Entry(root, width=30)
comm = tk.Entry(root, width=30)
    for label, entry in fields:
        tk.Label(root, text=label, font=("Arial", 12), bg="white").pack()
        entry.pack()
    def submit():
        try:
             processes.addemp(name.get(), address.get(), int(age.get()), int(salary.get()),\
float(comm.get())) # Pass data to backend
messagebox.showinfo("Success", "Employee added successfully!")
             home ()
        except Exception as e:
             messagebox.showerror("Error", str(e))
    tk.Button(root, text="Submit", command=submit, bg="lightgreen", font=("Arial", 12)).pack(pady=20)
    go_home_button()
# Add Supplier Page
def add_supplier_page():
    clear frame()
    tk.Label(root, text="Add Supplier", font=("Arial", 20), bg="white").pack(pady=20)
    name = tk.Entry(root, width=30)
    address = tk.Entry(root, width=30)
    material = tk.Entry(root, width=30)
    for label, entry in [("Name", name), ("Address", address), ("Material Type", material)]: tk.Label(root, text=label, font=("Arial", 12), bg="white").pack()
        entry.pack()
    def submit():
        try:
             processes.addsupp(name.get(), address.get(), material.get()) # Pass data to backend
messagebox.showinfo("Success", "Supplier added successfully!")
             home()
        except Exception as e:
             messagebox.showerror("Error", str(e))
    tk.Button(root, text="Submit", command=submit, bg="lightgreen", font=("Arial", 12)).pack(pady=20)
    go home button()
# Add Garment Page
def add_garment_page():
    clear_frame()
    tk.Label(root, text="Add Garment", font=("Arial", 20), bq="white") pack(pady=20)
    gtype = tk.Entry(root, width=30)
    qty = tk.Entry(root, width=30)
    price = tk.Entry(root, width=30)
sid = tk.Entry(root, width=30)
```

```
entry.pack()
  def submit():
     try:
         home()
     except Exception as e:
         messagebox.showerror("Error", str(e))
  tk.Button(root, text="Submit", command=submit, bg="lightgreen", font=("Arial", 12)).pack(pady=20)
  go home button()
# New Sale Page
def new_sale_page():
    clear_frame()
   tk.Label(root, text="New Sale", font=("Arial", 20), bg="white").pack(pady=20)
   gids = tk.Entry(root, width=50)
   contact = tk.Entry(root, width=30)
empid = tk.Entry(root, width=30)
   tk.Label(root, text="Enter Garment IDs (comma separated)", font=("Arial", 12), bg="white").pack()
   gids.pack()
   tk.Label(root, text="Customer Contact No.", font=("Arial", 12), bg="white") pack()
   contact.pack()
   tk.Label(root, text="Employee ID", font=("Arial", 12), bg="white").pack()
   empid.pack()
   def submit():
       try:
          id_list = [int(i.strip()) for i in gids.get().split(",")]
          processes.newsale(id_list, contact.get(), int(empid.get())) # Pass data to backend messagebox.showinfo("Success", "Sale recorded successfully!")
          home ()
     except Exception as e:
         messagebox.showerror("Error", str(e))
 tk.Button(root, text="Submit", command=submit, bq="lightqreen", font=("Arial", 12)).pack(pady=20)
 go home button()
# New Purchase Page
def new_purchase_page():
   clear frame()
   tk.Label(root, text="New Purchase", font=("Arial", 20), bg="white").pack(pady=20)
   sid = tk.Entry(root, width=30)
   gid = tk.Entry(root, width=30)
   price = tk.Entry(root, width=30)
   qty = tk.Entry(root, width=30)
   entry.pack()
   def submit():
          home ()
       except Exception as e:
          messagebox.showerror("Error", str(e))
   tk.Button(root, text="Submit", command=submit, bg="lightgreen", font=("Arial", 12)).pack(pady=20)
   go_home_button()
```

```
# View Net Sales
def view net sales():
   clear_frame()
   amt = processes.nets()
tk.Label(root, text=f"Total Net Sales: ₹{amt}", font=("Arial", 20), bg="white").pack(pady=40)
   go_home_button()
# View Net Purchases
def view net purchases():
   clear frame()
   amt = processes.netp()
    tk.Label(root, text=f"Total Net Purchases: ₹{amt}", font=("Arial", 20), bg="white").pack(pady=40)
   go_home_button()
# View All Sales
def view_all_sales():
    clear frame()
   tk.Label(root, text="All Sales", font=("Arial", 20), bg="white").pack(pady=20)
   sales = processes.allsales()
   table_frame = tk.Frame(root)
   table_frame.pack()
   columns = ("Sale ID", "Customer ID", "Employee ID", "Amount", "Employee")
   tree = ttk.Treeview(table frame, columns=columns, show="headings", height=15)
   for col in columns:
        tree.heading(col, text=col)
        tree.column(col, width=120, anchor="center")
   for sale in sales:
        tree.insert("", "end", values=sale)
   tree.pack()
   go home button()
# View All Purchases
def view_all_purchases():
    clear frame()
    tk.Label(root, text="All Purchases", font=("Arial", 20), bg="white").pack(pady=20)
    purchases = processes.allpurchase()
    table_frame = tk.Frame(root)
    table frame.pack()
    columns = ("Purchase ID", "Supplier ID", "Garment ID", "Price", "Quantity", "Supplier")
    tree = ttk.Treeview(table_frame, columns=columns, show="headings", height=15)
    for col in columns:
        tree.heading(col, text=col)
tree.column(col, width=120, anchor="center")
    for purchase in purchases:
        tree.insert("", "end", values=purchase)
    tree.pack()
    go home button()
```

```
# Remove Employee
def remove_employee_page():
    clear frame()
    tk.Label(root, text="Remove Employee", font=("Arial", 20), bg="white").pack(pady=20)

empid = tk.Entry(root, width=30)
    tk.Label(root, text="Employee ID to Remove", font=("Arial", 12), bg="white").pack()
    empid.pack()

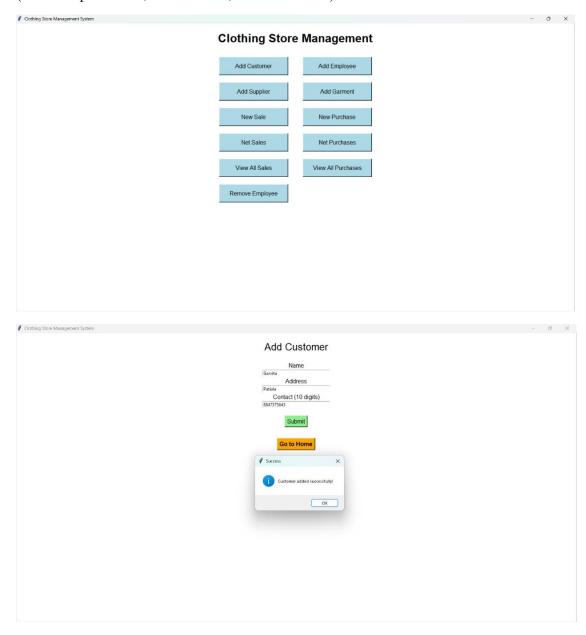
def submit():
    try:
        processes.rememp(int(empid.get()))  # Pass data to backend
        messagebox.showinfo("Success", "Employee removed successfully!")
        home()
    except Exception as e:
        messagebox.showerror("Error", str(e))

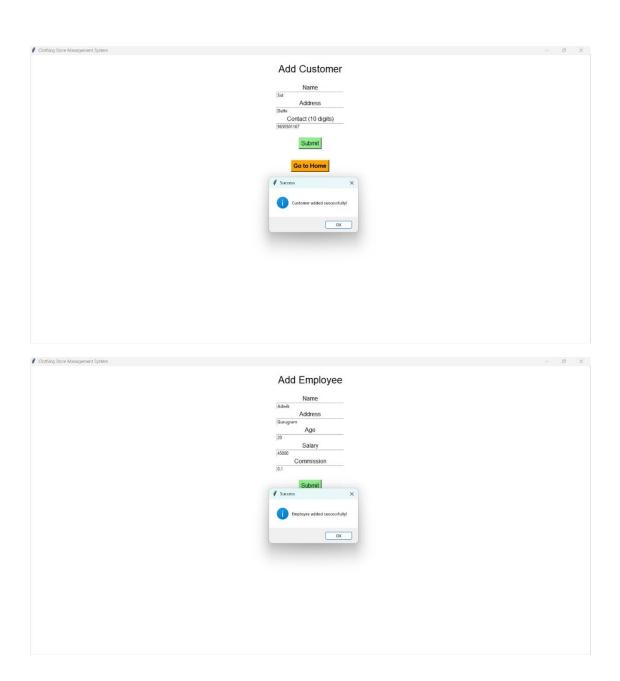
tk.Button(root, text="Submit", command=submit, bg="red", font=("Arial", 12)).pack(pady=20)
    go_home_button()

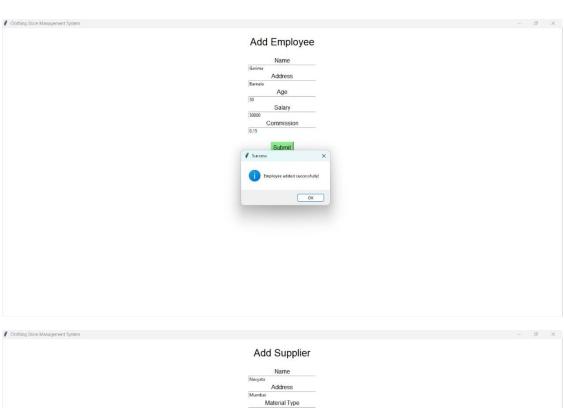
# Start with Home directly
home()
root.mainloop()
```

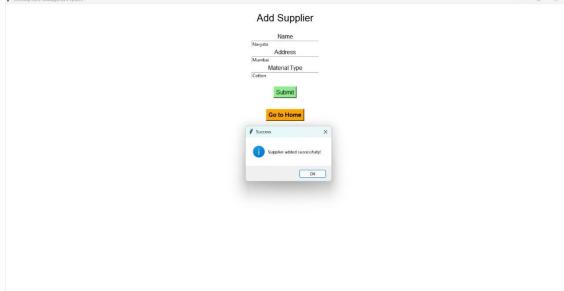
Chapter 7 SQL/PL/SQL Outputs and Discussion

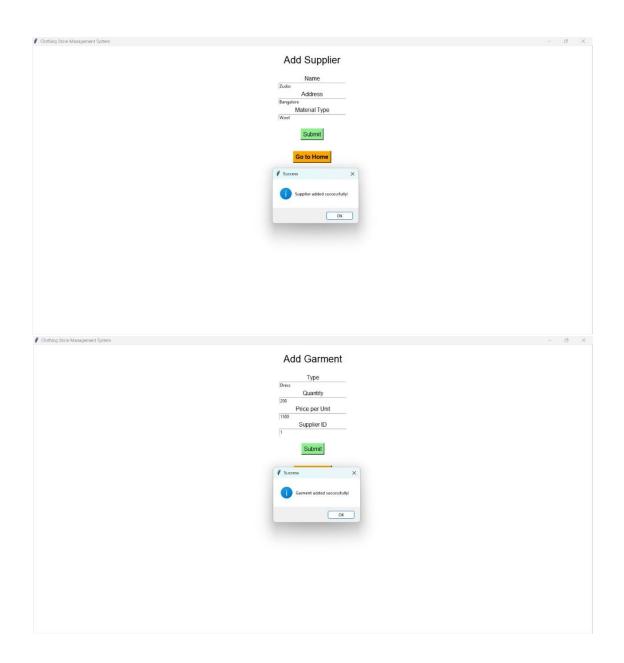
(Insert output results, observations, and discussion.)

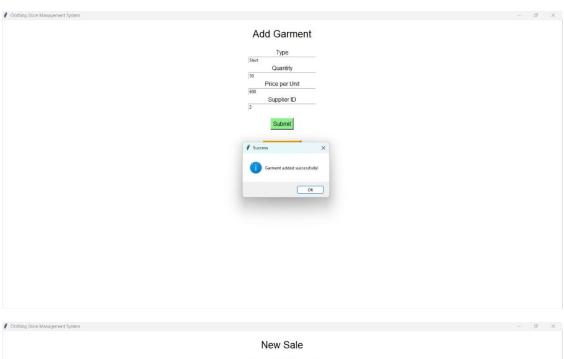


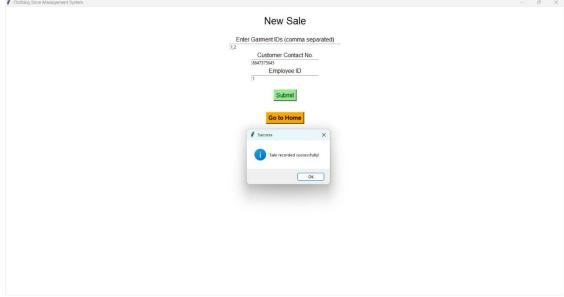


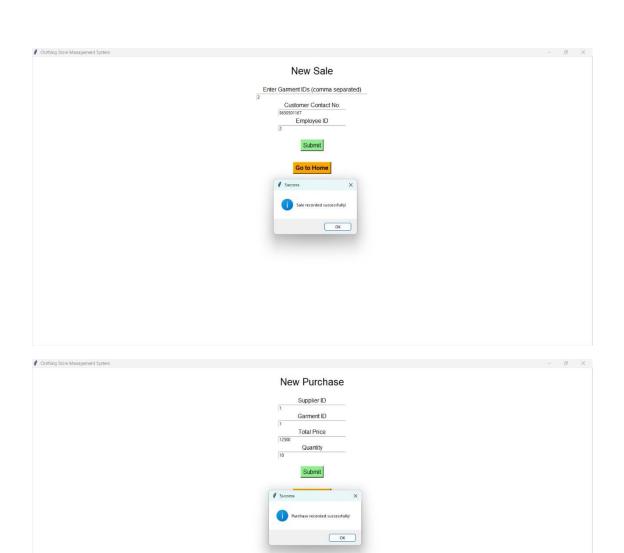


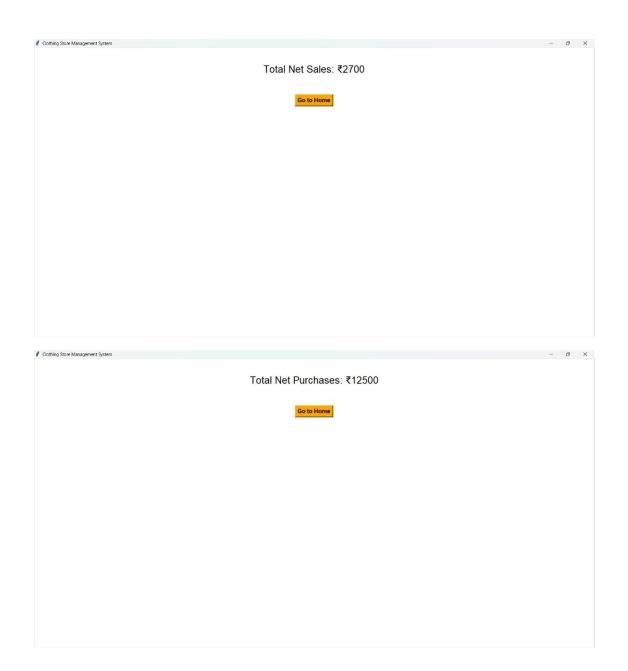


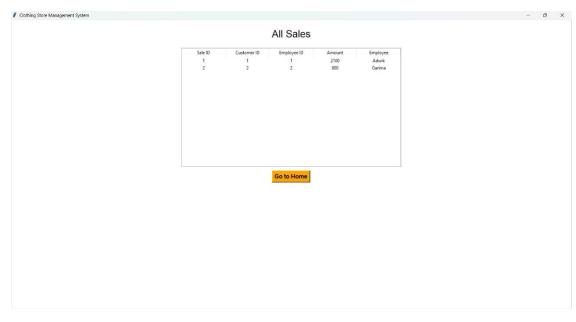


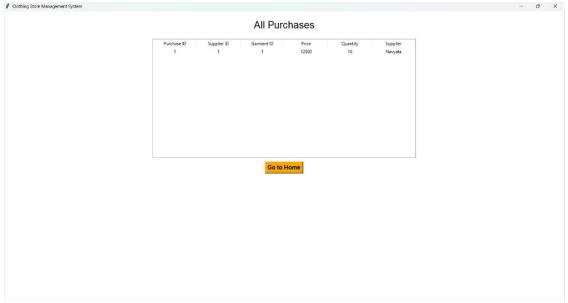


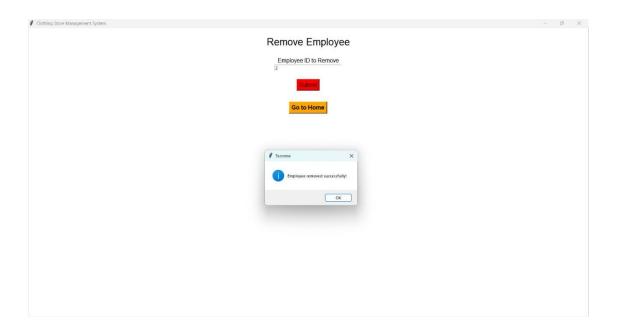












References

- MySQL Documentation: https://dev.mysql.com/doc/
- Python MySQL Connector Documentation: https://pypi.org/project/ mysql-connector-python/
- Database System Concepts by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan
- Fundamentals of Database Systems by Ramez Elmasri and Shamkant B. Navathe
- W3Schools SQL and MySQL Tutorials: https://www.w3schools.com/sql/
- GeeksforGeeks Python MySQL: https://www.geeksforgeeks.org/ python-database-connection-mysql/
- Class notes and instructor lectures
- Personal understanding and practical implementation