|  |
| --- |
| **<<** **Fashion Store Management>>**  **21CSS101J – PROGRAMMING FOR PROBLEM-SOLVING**  **Mini Project Report**  *Submitted by*  **Shreyas Manchanda RA2311003011185**  **B.Tech. CSE - Core**  **Jayant Munde RA2311003011206**  **B.Tech. CSE - Core**  **SRMIST-01.jpg**  **SCHOOL OF COMPUTING**  **COLLEGE OF ENGINEERING AND TECHNOLOGY**  **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**  **(Under Section 3 of UGC Act, 1956)**  S.R.M. NAGAR, KATTANKULATHUR – 603 203  CHENGALPATTU DISTRICT  **November 2023**  **COLLEGE OF ENGINEERING AND TECHNOLOGY**  **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**  **(Under Section 3 of UGC Act, 1956)**  S.R.M. NAGAR, KATTANKULATHUR – 603 203  **SRMIST-01.jpg**  **BONAFIDE CERTIFICATE**  Certified that Mini project report titled **Fashion Store Management System** is the bonafide work of **Shreyas Manchanda (RA2311003011185) and Jayant Munde (RA2311003011206)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate  **SIGNATURE SIGNATURE**  **(GUIDE) (HEAD OF THE DEPARTMENT)** |
|  |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S No.** | **Title** | **Page No.** |
| 1 | Problem Statement | 5 |
| 2 | Methodology / Procedure/ Algorithm | 6 |
| 3 | Flowchart | 8 |
| 4 | Coding (C/Python) | 9 |
| 5 | Front-end code (HTML, CSS, Javascript) [Optional] | 16 |
| 6 | Modules of the proposed work | 17 |
| 7 | Results/Screenshots | 18 |
| 8 | Conclusion | 21 |
| 9 | References | 22 |

1. **Problem Statement**

Fashion retail stores often face challenges in effectively managing their inventory, sales, and overall business operations. Manual tracking and recording of product details, purchases, and sales transactions can be time-consuming, error-prone, and inefficient. To address these issues, there is a need for an automated "Fashion Store Management Program" that streamlines processes such as adding, editing, deleting, and viewing product information, as well as handling purchase and sales operations. The lack of a structured and automated system can lead to inaccuracies in inventory levels, hinder decision-making, and result in a suboptimal shopping experience for customers. Therefore, the development of a robust and user-friendly program is crucial to enhance the operational efficiency of fashion stores, providing a comprehensive solution for efficient data management and reporting

1. **Methodology / Procedure/ Algorithm**
2. **Initialize Database Connection:**

Connect to the MySQL database named "fashion."

Create a cursor for database operations.

1. **Define Functions:**

Create functions for specific tasks: AddProduct, EditProduct, DelProduct, ViewProduct, PurchaseProduct, ViewPurchase, ViewStock, SaleProduct, ViewSales, MenuSet, and runAgain.

1. **Add Product Function (AddProduct):**

Accept user input for product details (ID, name, brand, type, season, rate).

Execute SQL queries to insert product details into the 'product' and 'stock' tables.

Print a confirmation message.

1. **Edit Product Function (EditProduct):**

Accept user input for product ID to edit.

Display current product details.

Accept user input for the field and value to be edited.

Execute SQL query to update the product details.

Print a confirmation message.

1. **Delete Product Function (DelProduct):**

Accept user input for the product ID to be deleted.

Execute SQL queries to delete corresponding records from 'sales', 'purchase', 'stock', and 'product' tables.

Print a deletion confirmation message.

1. **View Product Function (ViewProduct):**

Display a menu for different viewing options (all details, name, brand, type, season, ID).

Accept user input for the viewing option.

Execute SQL queries based on the user's choice and display the results.

1. **Purchase Product Function (PurchaseProduct):**

Generate a purchase ID based on the current date and time.

Accept user input for product ID, quantity, and discount.

Execute SQL queries to insert purchase details into the 'purchase' table and update stock information.

Print a purchase confirmation message.

1. **View Purchase Function (ViewPurchase):**

Accept user input for the product name.

Execute an SQL query to retrieve purchase details based on the product name.

Display the results.

1. **View Stock Function (ViewStock):**

Accept user input for the product name.

Execute an SQL query to retrieve stock details based on the product name.

Display the results.

1. **Sale Product Function (SaleProduct):**

Generate a sale ID based on the current date and time.

Accept user input for product ID, quantity, and discount.

Execute SQL queries to insert sale details into the 'sales' table and update stock information.

Print a sale confirmation message.

1. **View Sales Function (ViewSales):**

Accept user input for the product name.

Execute an SQL query to retrieve sales details based on the product name.

Display the results.

1. **Menu Set Function (MenuSet):**

Display a menu with options for various operations.

Accept user input for the desired operation.

Call the corresponding function based on the user's choice.

1. **Run Again Function (runAgain):**

Prompt the user if they want to run the program again.

If yes, clear the console and call the MenuSet function again.

1. **Program Execution:**

Display a welcome message.

Call the MenuSet function to initiate the program.

After each operation, call the runAgain function to check if the user wants to perform additional tasks.

**Software Requirements**

The "Fashion Store Management Program" is developed using Python and MySQL, with additional libraries for enhanced functionality. The code is written and executed in a Python development environment. The following software components are essential for the proper functioning of the program:

1. **Programming Language:**
   * Python 3.x
2. **Database Management System (DBMS):**
   * MySQL - A widely used open relational database management system.
3. **Python Libraries:**
   * **mysql-connector**: Required for MySQL database connectivity.
   * **pandas**: Optional, for data manipulation (though not extensively used in the provided code).
   * **datetime**: Standard Python library for working with dates and times.
4. **Integrated Development Environment (IDE):**
   * Any Python-compatible IDE, such as PyCharm, for coding and execution.

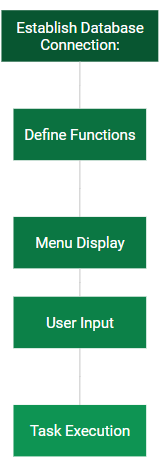
**Hardware Requirements**

To ensure optimal performance, the system requires the following hardware specifications:

1. **Operating System:**
   * Windows 10 or above.
2. **RAM:**
   * Minimum 4GB.
3. **Storage:**
   * Minimum 500GB hard disk space.

These software and hardware requirements are necessary for the successful execution of the "Fashion Store Management Program." Users should install the specified software components and ensure their hardware configurations meet or exceed the recommended specifications for optimal performance.

**3. Flow chart**

****

**4. Coding (Python)**

import os

import platform

import mysql.connector

import pandas as pd

import datetime

mydb=mysql.connector.connect(host="localhost", user="root", passwd="root", database="fashion")

mycursor=mydb.cursor()

def AddProduct():

L=[]

stk=[]

pid=input("Enter the Product ID : ")

L.append(pid)

IName=input("Enter the Product Name : ")

L.append(IName)

brnd=input("Enter the Product Brand Name : ")

L.append(brnd)

fr=input("Enter Male/Female/Kids : ")

L.append(fr)

sn=input("Enter Winter/Summer : ")

L.append(sn)

rate=int(input("Enter the Rates for Product :"))

L.append(rate)

product=(L)

sql="Insert into product (product\_id,PName,brand,Product\_for,Season,rate)values(%s,%s,%s,%s,%s,%s)"

mycursor.execute(sql,product)

mydb.commit()

stk.append(pid)

stk.append(0)

stk.append("No")

st=(stk)

sql="insert into stock(item\_id, Instock, status) values(%s,%s,%s)"

mycursor.execute(sql,st)

mydb.commit()

print("One Product inserted ")

def EditProduct():

pid=input("Enter product ID to be edited : ")

sql="select \* from product where product\_id=%s"

ed=(pid,)

mycursor.execute(sql,ed)

res=mycursor.fetchall()

for x in res:

print(x)

print("")

fld=input("Enter the field which you want to edit : ")

val=input("Enter the value you want to set : ")

sql="Update product set " + fld +"='" + val + "' where product\_id='" + pid + "'"

sq=sql

mycursor.execute(sql)

print("Editing Don : ")

print("After correction the record is : ")

sql="select \* from product where product\_id=%s"

ed=(pid,)

mycursor.execute(sql,ed)

res=mycursor.fetchall()

for x in res:

print(x)

mydb.commit()

def DelProduct():

pid=input("Enter the Product)id to be deleted : ")

sql="delete from sales where item\_id=%s"

id=(pid,)

mycursor.execute(sql,id)

mydb.commit()

sql="delete from purchase where item\_id=%s"

mycursor.execute(sql,id)

mydb.commit()

sql="delete from stock where item\_id=%s"

mycursor.execute(sql,id)

mydb.commit()

sql="delete from product where product\_id=%s"

mycursor.execute(sql,id)

mydb.commit()

print("One Item Deleted")

def ViewProduct():

print("Display Menu: Select the category to display the data")

print("1. All Details")

print("2. Product Name:")

print("3. Product Brand:")

print("4. Product For:")

print("5. Product Season:")

print("6. Product ID:")

x=0

ch=int(input("Enter your choice to display : "))

if ch==1:

sql="select \* from product"

mycursor.execute(sql)

res=mycursor.fetchall()

for x in res:

print(x)

x=1

elif ch==2:

var='PName'

val=input("Enter the name of Product : ")

elif ch==3:

var='brand'

val=input("Enter the name of Brand : ")

elif ch==4:

var='Product\_for'

val=input("Enter Male/Femal/Kids : ")

elif ch==5:

var='season'

val=input("Enter the Season : ")

elif ch==6:

var='product\_id'

val=input("Enter the Product\_id : ")

if x==0:

sql="select \* from product where " + var + " = %s"

sq=sql

tp=(val,)

mycursor.execute(sq,tp)

res=mycursor.fetchall()

for x in res:

print(x)

def PurchaseProduct():

mn=""

dy=""

now=datetime.datetime.now()

purchaseID="P"+str(now.year)+str(now.month)+str(now.day)+str(now.hour)+str(now.minute)+str(now.second)

L=[]

Lst=[]

L.append(purchaseID)

itemId=input("Enter Product ID : ")

L.append(itemId)

itemNo=int(input("Enter the number of Items : "))

L.append(itemNo)

sql="select rate from product where product\_id=%s"

pid=(itemId,)

mycursor.execute(sql,pid)

res=mycursor.fetchone()

for x in res:

print("rate is : ", x)

amount=x\*itemNo

print("Amount is :", amount)

L.append(amount)

mnth=now.month

if mnth<=9:

mn="0"+str(mnth)

else:

mn=str(mnth)

day=now.day

if day<=9:

dy="0"+str(day)

else:

dy=str(day)

dt=str(now.year)+"-"+mn+"-"+dy

L.append(dt)

tp=(L)

sql="insert into purchase(purchase\_id,item\_id,no\_of\_items,amount,Purchase\_date)values(%s,%s,%s,%s,%s)"

mycursor.execute(sql,tp)

mydb.commit()

sql="Select Instock from stock where item\_id=%s"

mycursor.execute(sql,pid)

res=mycursor.fetchall()

status="No"

for x in res:

print(x)

instock=x[0]+itemNo

if instock>0:

status="Yes"

Lst.append(instock)

Lst.append(status)

Lst.append(itemId)

tp=(Lst)

sql="update stock set instock=%s,status=%s where item\_id=%s"

mycursor.execute(sql,tp)

mydb.commit()

print("1 Item purchased and saved in Database")

def ViewPurchase():

item=input("Enter Product Name : ")

sql="select product.product\_id,

product.PName,product.brand,purchase.no\_of\_items,purchase.purchase\_date,purchase.amount from product

INNER JOIN purchase ON product.product\_id=purchase.item\_id and product.PName=%s"

itm=(item,)

mycursor.execute(sql,itm)

res=mycursor.fetchall()

for x in res:

print(x)

def ViewStock():

item=input("Enter Product Name : ")

sql="select product.product\_id,product.PName,stock.Instock,\

stock.status from stock, product where \

product.product\_id=stock.item\_id and product.PName=%s"

itm=(item,)

mycursor.execute(sql,itm)

res=mycursor.fetchall()

for x in res:

print(x)

def SaleProduct():

now=datetime.datetime.now()

saleID="S"+str(now.year)+str(now.month)+str(now.day)+str(now.hour)+str(now.minute)+str(now.second)

L=[]

L.append(saleID)

itemId=input("Enter Product ID : ")

L.append(itemId)

itemNo=int(input("Enter the number of Items : "))

L.append(itemNo)

sql="select rate from product where product\_id=%s"

pid=(itemId,)

mycursor.execute(sql,pid)

res=mycursor.fetchall()

for x in res:

print("The rate of item is :",x)

dis=int(input("Enter the discount : "))

saleRate=x[0]-(x[0]\*dis/100)

L.append(saleRate)

amount=itemNo\*saleRate

L.append(amount)

mnth=now.month

if mnth<=9:

mn="0"+str(mnth)

else:

mn=str(mnth)

day=now.day

if day<=9:

dy="0"+str(day)

else:

dy=str(day)

dt=str(now.year)+"-"+mn+"-"+dy

L.append(dt)

tp=(L)

sql="insert into sales (sale\_id, item\_id,no\_of\_item\_sold,\

sale\_rate,amount,date\_of\_sale) values(%s,%s,%s,%s,%s,%s)"

mycursor.execute(sql,tp)

mydb.commit()

sql="Select Instock from stock where item\_id=%s"

mycursor.execute(sql,pid)

res=mycursor.fetchall()

for x in res:

print("Total Items in Stock are : ",x)

instock=x[0]-itemNo

if instock>0:

status="Yes"

tp=(instock,status,itemId)

sql="update stock set instock=%s,status=%s where item\_id=%s"

print("Remaining Items in Stock are : ",instock)

mycursor.execute(sql,tp)

mydb.commit()

def ViewSales():

item=input("Enter Product Name : ")

sql="select product.product\_id, product.PName,product.brand,\

sales.no\_of\_item\_sold,sales.date\_of\_sale,sales.amount \

from sales, product where product.product\_id=sales.item\_id \

and product.PName=%s"

itm=(item,)

mycursor.execute(sql,itm)

res=mycursor.fetchall()

for x in res:

print(x)

def MenuSet(): #Function For The SFashion Store System

print("Enter 1 : To Add Product ")

print("Enter 2 : To Edit Product ")

print("Enter 3 : To Delete Product ")

print("Enter 4 : To View Product ")

print("Enter 5 : To Purchase Product")

print("Enter 6 : To View Purchases")

print("Enter 7 : To View Stock Detials")

print("Enter 8 : To Sale the item")

print("Enter 9 : To View Sales Detials")

try: #Using Exceptions For Validation

userInput = int(input("Please Select An Above Option: ")) #Will Take Input From User

except ValueError:

exit("\nHy! That's Not A Number") #Error Message

else:

print("\n") #Print New Line

if(userInput == 1):

AddProduct()

elif(userInput == 2):

EditProduct()

elif (userInput==3):

DelProduct()

elif (userInput==4):

ViewProduct()

elif (userInput==5):

PurchaseProduct()

elif (userInput==6):

ViewPurchase()

elif (userInput==7):

ViewStock()

elif (userInput==8):

SaleProduct()

elif (userInput==9):

ViewSales()

else:

print("Enter correct choice. . . ")

print("\*"\*80)

print("\* \* \* \* \* \* \* Welcome to the Project of Fashion Store \* \* \* \* \* \* \* ")

print("\* \* \* \* Developed by: Neha Tyagi, PGT CS, KV no. 5 Jaipur : \* \* \* \* ")

print("\*"\*80)

print("")

MenuSet()

def runAgain():

runAgn = input("\nwant To Run Again Y/n: ")

while(runAgn.lower() == 'y'):

if(platform.system() == "Windows"):

print(os.system('cls'))

else:

print(os.system('clear'))

MenuSet()

runAgn = input("\nwant To Run Again Y/n: ")

runAgain()

**5. Front-end code (HTML, CSS, Javascript) (Optional)**

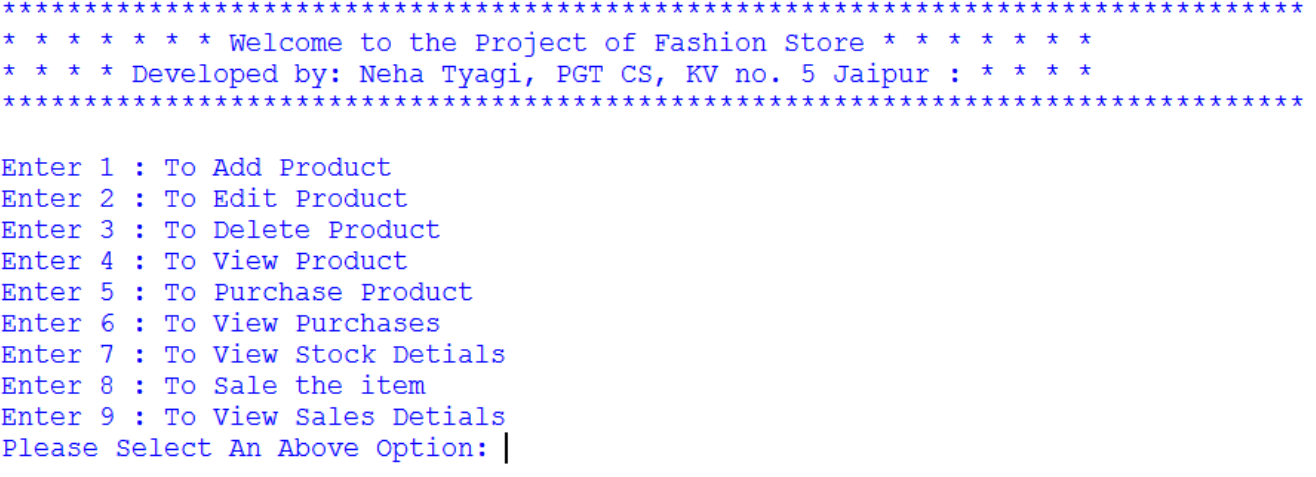
**6. Modules of the proposed work**

1. **Database Connection Module:**
   * Establishes a connection to the MySQL database using the **mysql-connector** library.
   * Handles database cursor initialization for executing queries.
2. **Product Management Module:**
   * **Add Product Function:**
     + Accepts user input to add a new product to the system.
     + Inserts product details into the 'product' table in the database.
     + Updates the 'stock' table with the product's initial stock information.
   * **Edit Product Function:**
     + Allows users to modify existing product details.
     + Retrieves and displays the current product details based on the provided product ID.
     + Updates the specified product field with the user's input.
   * **Delete Product Function:**
     + Deletes a product along with its associated records from 'sales,' 'purchase,' 'stock,' and 'product' tables.
   * **View Product Function:**
     + Displays product details based on user-selected criteria (all details, name, brand, type, season, ID).
3. **Purchase and Sale Module:**
   * **Purchase Product Function:**
     + Generates a unique purchase ID based on the current date and time.
     + Accepts user input for product ID, quantity, and calculates the purchase amount.
     + Updates the 'purchase' table and adjusts stock information.
   * **Sale Product Function:**
     + Generates a unique sale ID based on the current date and time.
     + Accepts user input for product ID, quantity, and discount.
     + Updates the 'sales' table and adjusts stock information.
4. **View Details Module:**
   * **View Purchase Function:**
     + Displays purchase details for a specific product based on user input.
   * **View Stock Function:**
     + Displays stock details for a specific product based on user input.
   * **View Sales Function:**
     + Displays sales details for a specific product based on user input.
5. **Menu Interface Module:**
   * **MenuSet Function:**
     + Displays a user-friendly menu with options for different operations.
     + Takes user input and calls corresponding functions based on the choice.
6. **System Flow Control Module:**
   * **runAgain Function:**
     + Prompts the user to run the program again.
     + Clears the console and calls the MenuSet function for additional operations.

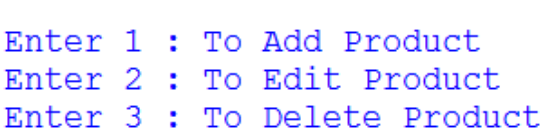
These modules collectively form the "Fashion Store Management Program," providing functionalities for product management, purchase and sale transactions, and detailed information viewing. The code is developed using Python and MySQL, ensuring compatibility with the specified software requirements.

**7. Results/Screenshots**

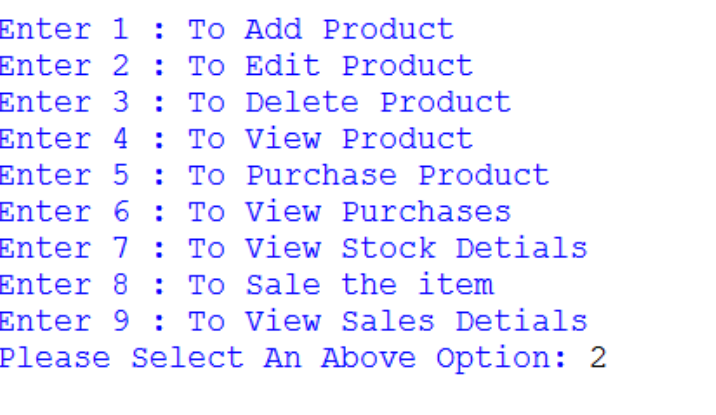
**Main Table**

****

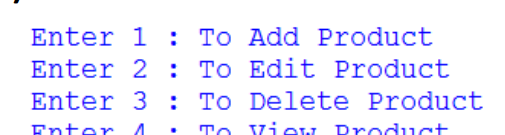
**Add Product**

****

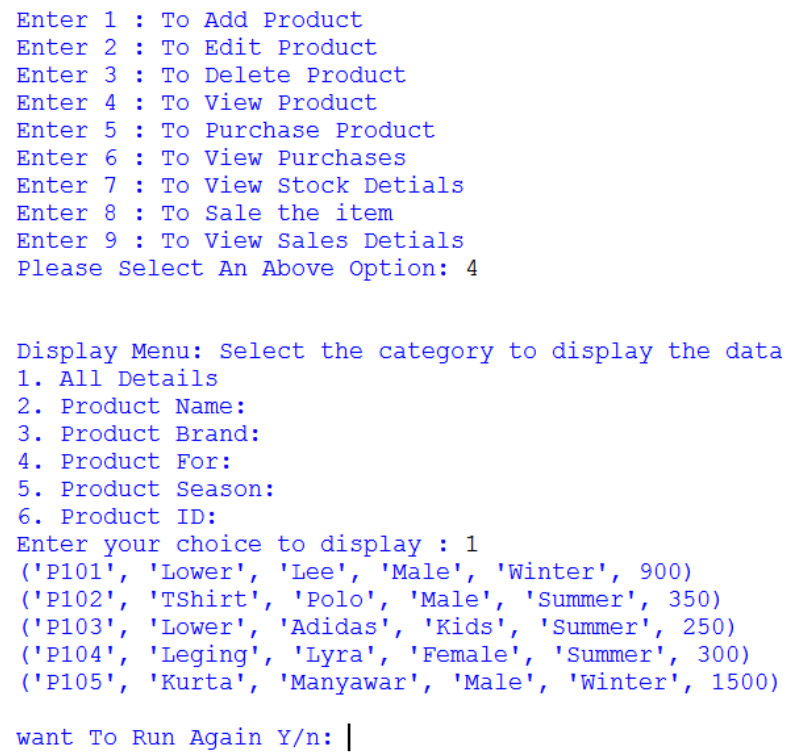
**Edit Product**

****

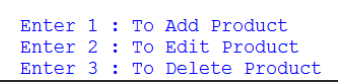
**Delete Product**

****

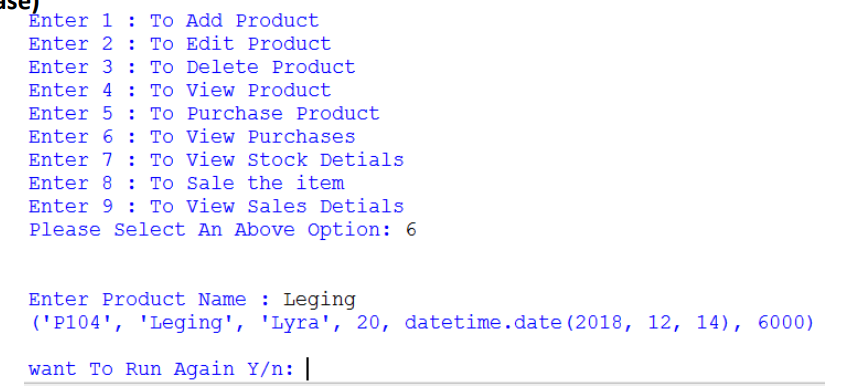
**View Product**

****

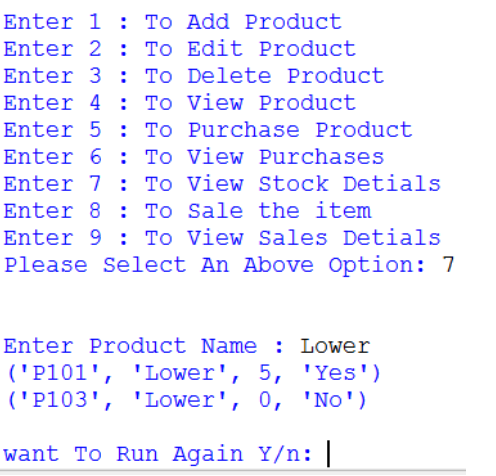
**Purchase Product**

****

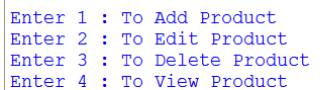
**View Purchase**

****

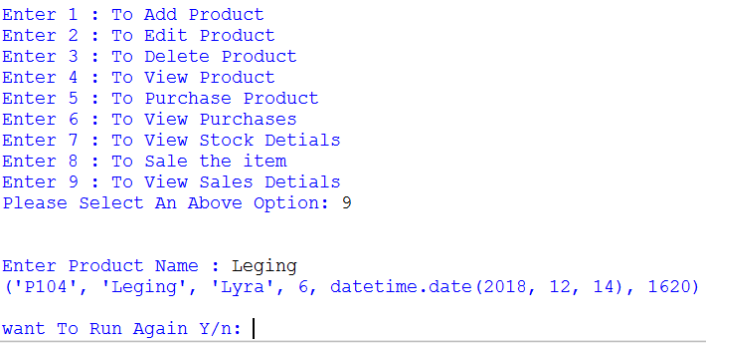
**View Stock details**

****

**Sale Item**

****

**View Sale Details**

****

**8. Conclusion**

In conclusion, the "Fashion Store Management Program" offers a versatile and efficient solution for managing various aspects of a fashion retail store. With the ability to add, edit, delete, and view product details, as well as facilitate purchase and sales operations, this program provides a comprehensive framework for fashion store management. The user-friendly menu system ensures easy navigation and execution of tasks, while the continuous operation option allows for seamless task repetition. By establishing a connection to a MySQL database, the program enables real-time data management and reporting. To enhance this system, it's important to address indentation issues and implement robust error handling. Overall, this program demonstrates the potential for technology to optimize fashion store operations and serve as a valuable tool for business management.

**9. References**

<https://www.w3schools.com/python/python_mysql_getstarted.asp>