Nupur Deshpande

Batch – 1039

Project - Inventory Management System

import mysql.connector

from datetime import datetime

# Connect to MySQL

db = mysql.connector.connect(

    host="localhost",

    user="root",

    password="1234",  # Change if needed

    database="python\_1234"

)

cursor = db.cursor()

# print(" Connected to database.")

# # Create products table

# cursor.execute("""

#     CREATE TABLE IF NOT EXISTS products (

#         id INT PRIMARY KEY,

#         name VARCHAR(50),

#         quantity INT,

#         price FLOAT

#     )

# """)

# # Create purchases table

# cursor.execute("""

#     CREATE TABLE IF NOT EXISTS purchases (

#         id INT AUTO\_INCREMENT PRIMARY KEY,

#         product\_id INT,

#         quantity INT,

#         purchase\_price FLOAT,

#         purchase\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

#     )

# """)

# # Create sales table

# cursor.execute("""

#     CREATE TABLE IF NOT EXISTS sales (

#         id INT AUTO\_INCREMENT PRIMARY KEY,

#         product\_id INT,

#         quantity INT,

#         sale\_price FLOAT,

#         sale\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

#     )

# """)

# db.commit()

# print("table created sucessfully")

# ------------------ Product Functions ------------------

def add\_product(id, name, quantity, price):

    sql = "INSERT INTO products (id, name, quantity, price) VALUES (%s, %s, %s, %s)"

    cursor.execute(sql, (id, name, quantity, price))

    db.commit()

    print("Product added successfully.")

def view\_products():

    cursor.execute("SELECT \* FROM products")

    rows = cursor.fetchall()

    print("\nID | Name       | Quantity | Price")

    print("-" \* 40)

    for row in rows:

        print(f"{row[0]} | {row[1]} | {row[2]}       | ${row[3]:.2f}")

def update\_product(id, quantity):

    cursor.execute("UPDATE products SET quantity = %s WHERE id = %s", (quantity, id))

    db.commit()

    print("Product updated successfully.")

def delete\_product(id):

    cursor.execute("DELETE FROM products WHERE id = %s", (id,))

    db.commit()

    print("Product deleted successfully.")

def search\_product():

    term = input("Enter Product ID or Name to search: ")

    if term.isdigit():

        cursor.execute("SELECT \* FROM products WHERE id = %s", (int(term),))

    else:

        cursor.execute("SELECT \* FROM products WHERE name LIKE %s", ('%' + term + '%',))

    rows = cursor.fetchall()

    if rows:

        for row in rows:

            print(f"{row[0]} | {row[1]} | {row[2]} | ${row[3]:.2f}")

    else:

        print("No product found.")

def low\_stock\_alert():

    cursor.execute("SELECT \* FROM products WHERE quantity < 5")

    rows = cursor.fetchall()

    print("\nLow Stock Products (Qty < 5):")

    if rows:

        for row in rows:

            print(f"{row[0]} | {row[1]} | {row[2]} | ${row[3]:.2f}")

    else:

        print("All products are well stocked.")

def total\_inventory\_value():

    cursor.execute("SELECT SUM(quantity \* price) FROM products")

    total = cursor.fetchone()[0]

    print(f"Total Inventory Value: ${total:.2f}" if total else "Inventory is empty.")

def sort\_products():

    print("""

Sort By:

1. Price (Low to High)

2. Price (High to Low)

3. Quantity (Low to High)

4. Quantity (High to Low)

""")

    opt = int(input("Enter your option: "))

    order\_map = {

        1: "price ASC", 2: "price DESC",

        3: "quantity ASC", 4: "quantity DESC"

    }

    if opt in order\_map:

        cursor.execute(f"SELECT \* FROM products ORDER BY {order\_map[opt]}")

        rows = cursor.fetchall()

        for row in rows:

            print(f"{row[0]} | {row[1]} | {row[2]} | ${row[3]:.2f}")

    else:

        print("Invalid sort option.")

# ------------------ Purchases and Sales ------------------

def record\_purchase(product\_id, quantity, purchase\_price):

    cursor.execute("INSERT INTO purchases (product\_id, quantity, purchase\_price) VALUES (%s, %s, %s)",

                   (product\_id, quantity, purchase\_price))

    cursor.execute("UPDATE products SET quantity = quantity + %s WHERE id = %s",

                   (quantity, product\_id))

    db.commit()

    print("Purchase recorded successfully.")

def record\_sale(product\_id, quantity, sale\_price):

    cursor.execute("SELECT quantity FROM products WHERE id = %s", (product\_id,))

    result = cursor.fetchone()

    if result and result[0] >= quantity:

        cursor.execute("INSERT INTO sales (product\_id, quantity, sale\_price) VALUES (%s, %s, %s)",

                       (product\_id, quantity, sale\_price))

        cursor.execute("UPDATE products SET quantity = quantity - %s WHERE id = %s",

                       (quantity, product\_id))

        db.commit()

        print("Sale recorded successfully.")

    else:

        print("Not enough stock.")

# ------------------ Profit & Loss Report ------------------

from decimal import Decimal

def get\_average\_purchase\_price(product\_id):

    cursor.execute("""

        SELECT SUM(CAST(quantity \* purchase\_price AS DECIMAL(10,2))),

               SUM(quantity)

        FROM purchases

        WHERE product\_id = %s

    """, (product\_id,))

    result = cursor.fetchone()

    if result and result[1] and result[0]:

        total\_cost = float(result[0])  # Decimal -> float

        total\_qty = float(result[1])

        return total\_cost / total\_qty

    return 0.0

def profit\_loss\_report():

    cursor.execute("SELECT product\_id, quantity, sale\_price FROM sales")

    sales = cursor.fetchall()

    profit\_total = 0

    print("\nPROFIT & LOSS REPORT")

    print("-" \* 50)

    print("Product ID | Qty | Sale Price | Avg Buy Price | Profit")

    for sale in sales:

        pid, qty, s\_price = sale

        s\_price = float(s\_price)

        avg\_buy\_price = get\_average\_purchase\_price(pid)

        profit = (s\_price - avg\_buy\_price) \* qty

        profit\_total += profit

        print(f"{pid:^11} | {qty:^3} | ${s\_price:^10.2f} | ${avg\_buy\_price:^13.2f} | ${profit:^7.2f}")

    print("-" \* 50)

    print(f"Total Profit: ${profit\_total:.2f}")

# ------------------ Menu ------------------

while True:

    print("""

========= Inventory Management System =========

1.  Add Product

2.  View All Products

3.  Update Product Quantity

4.  Delete Product

5.  Search Product

6.  Show Low Stock Items

7.  Total Inventory Value

8.  Sort Products

9.  Record Purchase

10. Record Sale

11. Profit & Loss Report

12. Exit

""")

    try:

        choice = int(input("Enter your choice: "))

        if choice == 1:

            id = int(input("Enter Product ID: "))

            name = input("Enter Product Name: ")

            quantity = int(input("Enter Quantity: "))

            price = float(input("Enter Price: "))

            add\_product(id, name, quantity, price)

        elif choice == 2:

            view\_products()

        elif choice == 3:

            id = int(input("Enter Product ID to update: "))

            quantity = int(input("Enter new Quantity: "))

            update\_product(id, quantity)

        elif choice == 4:

            id = int(input("Enter Product ID to delete: "))

            delete\_product(id)

        elif choice == 5:

            search\_product()

        elif choice == 6:

            low\_stock\_alert()

        elif choice == 7:

            total\_inventory\_value()

        elif choice == 8:

            sort\_products()

        elif choice == 9:

            product\_id = int(input("Enter Product ID: "))

            quantity = int(input("Enter Quantity Purchased: "))

            price = float(input("Enter Purchase Price per unit: "))

            record\_purchase(product\_id, quantity, price)

        elif choice == 10:

            product\_id = int(input("Enter Product ID: "))

            quantity = int(input("Enter Quantity Sold: "))

            price = float(input("Enter Sale Price per unit: "))

            record\_sale(product\_id, quantity, price)

        elif choice == 11:

            profit\_loss\_report()

        elif choice == 12:

            print("Exiting program. Goodbye!")

            break

        else:

            print("Invalid choice.")

    except Exception as e:

        print("Error:", e)