Implementing OOPs Concept in Electronic Gadgets Database

1.CUSTOMER REGISTRATION

Description: When a new customer registers on the Tech Shop website, their information (e.g., name, email, phone) needs to be stored in the database.

Task 1: Implement a registration form and database connectivity to insert new customer records. Ensure proper data validation and error handling for duplicate email addresses.

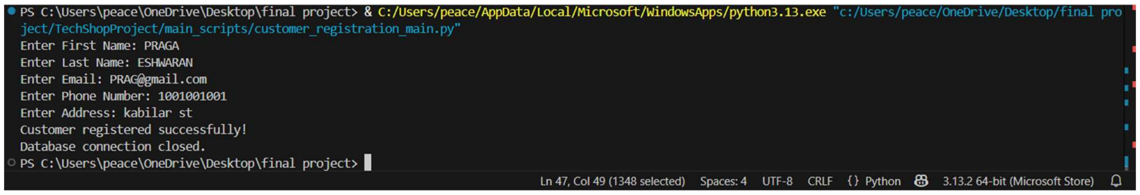
CODE:

```
import mysql.connector

import re


def get_db_connection():

  """Establish database connection"""

  try:

    conn = mysql.connector.connect(

      host="localhost",

      user="root",

      password="root",

      database="TechShop"

    )

    return conn

  except mysql.connector.Error as e:

    print(f"Error connecting to database: {e}")

    return None
```

```
if __name__ == "__main__":

  try:
    # Simulating user input
    first_name = input("Enter First Name: ")
    last_name = input("Enter Last Name: ")
    email = input("Enter Email: ")
    phone = input("Enter Phone Number: ")
    address = input("Enter Address: ")

    conn = get_db_connection()
    if conn:
      cursor = conn.cursor()
      query = "INSERT INTO CUSTOMERS (FirstName, LastName, Email, 
Phone, Address) VALUES (%s, %s, %s, %s, %s)"
      values = (first_name, last_name, email, phone, address)
      cursor.execute(query, values)
      conn.commit()
      print("Customer registered successfully!")



  except Exception as e:
    print(f"Error: {e}")
```

```
finally:
    if conn:
        cursor.close()
        conn.close()
        print("Database connection closed.")
```

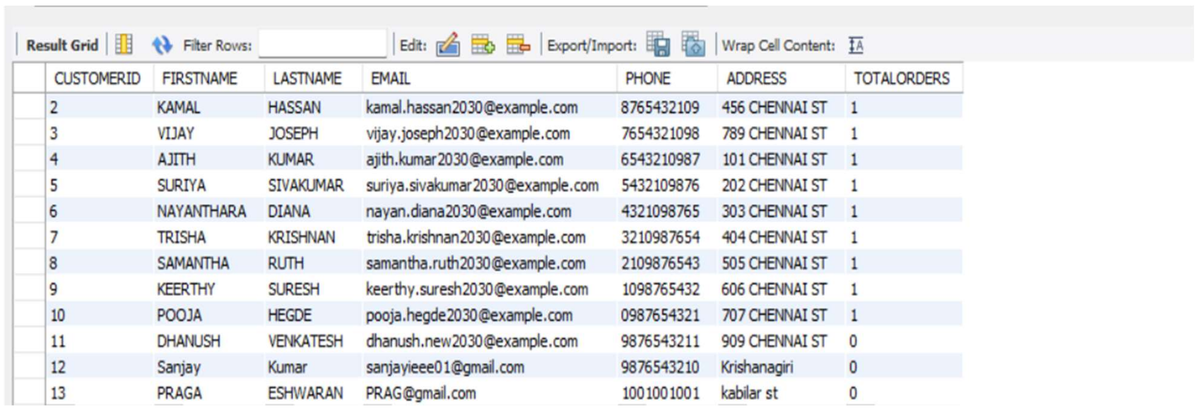
OUTPUT



```
PS C:\Users\peace\OneDrive\Desktop\final project> & C:/Users/peace/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/peace/OneDrive/Desktop/final pro
ject/TechShopProject/main_scripts/customer_registration_main.py"
Enter First Name: PRAGA
Enter Last Name: ESHWARAN
Enter Email: PRAG@gmail.com
Enter Phone Number: 1001001001
Enter Address: kabilar st
Customer registered successfully!
Database connection closed.
PS C:\Users\peace\OneDrive\Desktop\final project>
```

Ln 47, Col 49 (1348 selected)
Spaces: 4
UTF-8
CRLF
{} Python
3.13.2 64-bit (Microsoft Store)

MYSQL OUTPUT



CUSTOMERID	FIRSTNAME	LASTNAME	EMAIL	PHONE	ADDRESS	TOTALORDERS
2	KAMAL	HASSAN	kamal.hassan2030@example.com	8765432109	456 CHENNAI ST	1
3	VIJAY	JOSEPH	vijay.joseph2030@example.com	7654321098	789 CHENNAI ST	1
4	AJITH	KUMAR	ajith.kumar2030@example.com	6543210987	101 CHENNAI ST	1
5	SURIYA	SIVAKUMAR	suriya.sivakumar2030@example.com	5432109876	202 CHENNAI ST	1
6	NAYANTHARA	DIANA	nayan.diana2030@example.com	4321098765	303 CHENNAI ST	1
7	TRISHA	KRISHNAN	trisha.krishnan2030@example.com	3210987654	404 CHENNAI ST	1
8	SAMANTHA	RUTH	samantha.ruth2030@example.com	2109876543	505 CHENNAI ST	1
9	KEERTHY	SURESH	keerthy.suresh2030@example.com	1098765432	606 CHENNAI ST	1
10	POOJA	HEGDE	pooja.hegde2030@example.com	0987654321	707 CHENNAI ST	1
11	DHANUSH	VENKATESH	dhanush.new2030@example.com	9876543211	909 CHENNAI ST	0
12	Sanjay	Kumar	sanjayieee01@gmail.com	9876543210	Krishanagiri	0
13	PRAGA	ESHWARAN	PRAG@gmail.com	1001001001	kabilar st	0

2: CUSTOMER ACCOUNT MANAGEMENT

Description: The Customer Account Management module enables updating customer details like name, email, phone, and address. It ensures accurate and up-to-date customer information in the system.
This helps maintain data integrity and improves customer service.
Admins can easily modify records through a user-friendly interface or system

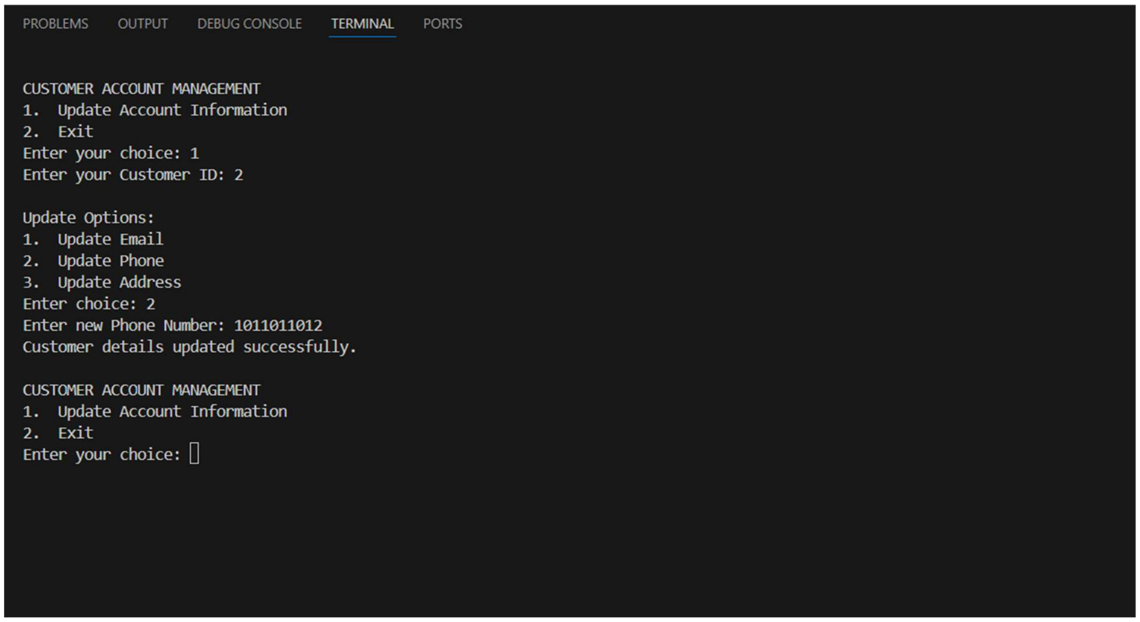
Task: Create an interface to manage the product catalog. Implement database connectivity to update product information. Handle changes in product details and ensure data consistency.

CODE

```
import mysql.connector

import re


def get_db_connection():

  """Establish database connection"""

  try:

    conn = mysql.connector.connect(

      host="localhost",

      user="root",

      password="root",

      database="TechShop"

    )

    return conn

  except mysql.connector.Error as e:

    print(f"Error connecting to database: {e}")

    return None
```

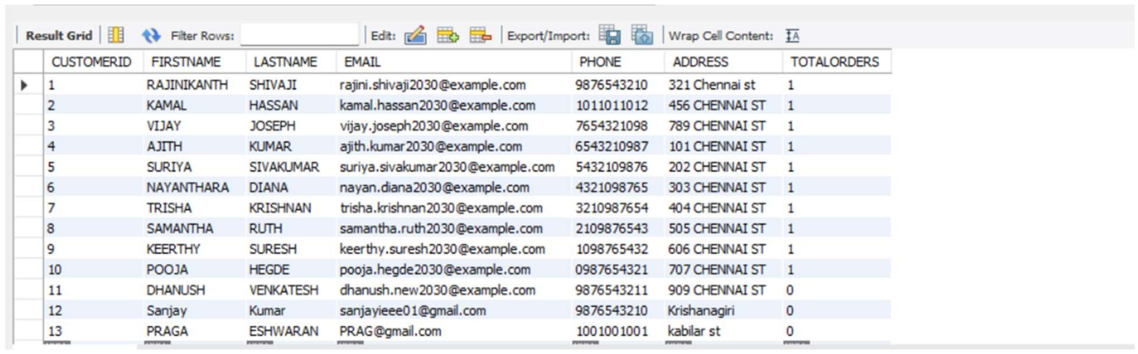
```
def validate_email(email):
    """Validate email format"""
    pattern = r'^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$'
    return re.match(pattern, email)

def validate_phone(phone):
    """Validate phone number (only digits, length 10-15)"""
    return phone.isdigit() and 10 <= len(phone) <= 15

def update_customer_profile(cursor, conn):
    """Update customer details"""
    customer_id = input("Enter your Customer ID: ")
    # Check if customer exists
    cursor.execute("SELECT * FROM customers WHERE CustomerID = %s",
(customer_id,))
    customer = cursor.fetchone()
    if not customer:
        print("Customer not found.")
        return

    print("\nUpdate Options:")
    print("1.  Update Email")
    print("2.  Update Phone")
    print("3.  Update Address")
    choice = input("Enter choice: ")

    if choice == "1":
        new_email = input("Enter new Email: ")
        if not validate_email(new_email):
            print("Invalid email format.")
            return
        cursor.execute("UPDATE customers SET Email = %s WHERE CustomerID = %s",
                       (new_email, customer_id))
    elif choice == "2":
        new_phone = input("Enter new Phone Number: ")
        if not validate_phone(new_phone):
            print("Invalid phone number. Must be 10-15 digits.")
            return
        cursor.execute("UPDATE customers SET Phone = %s WHERE CustomerID = %s",
                       (new_phone, customer_id))
    elif choice == "3":
        new_address = input("Enter new Address: ")
        cursor.execute("UPDATE customers SET Address = %s WHERE CustomerID = %s",
                       (new_address, customer_id))
    else:
        print("Invalid choice.")
        return

    conn.commit()
    print("Customer details updated successfully.")

def main():
    """Main function to manage customer updates"""
    conn = get_db_connection()
    if not conn:
        return
    cursor = conn.cursor()
    while True:
        print("\nCUSTOMER ACCOUNT MANAGEMENT")
        print("1.  Update Account Information")
        print("2.  Exit")
        choice = input("Enter your choice: ")
        if choice == "1":
            update_customer_profile(cursor, conn)
        elif choice == "2":
            print("Exiting Customer Account Management...")
            break
        else:
            print("Invalid choice. Please try again.")
    cursor.close()
    conn.close()

if __name__ == "__main__":
    main()
```

OUTPUT



```
CUSTOMER ACCOUNT MANAGEMENT
1.  Update Account Information
2.  Exit
Enter your choice: 1
Enter your Customer ID: 2

Update Options:
1.  Update Email
2.  Update Phone
3.  Update Address
Enter choice: 2
Enter new Phone Number: 1011011012
Customer details updated successfully.

CUSTOMER ACCOUNT MANAGEMENT
1.  Update Account Information
2.  Exit
Enter your choice:
```

MYSQL OUTPUT



CUSTOMERID	FIRSTNAME	LASTNAME	EMAIL	PHONE	ADDRESS	TOTALORDERS
1	RAJINIKANTH	SHIVAJI	rajini.shivaji2030@example.com	9876543210	321 Chennai st	1
2	KAMAL	HASSAN	kamal.hassan2030@example.com	1011011012	456 CHENNAI ST	1
3	VIJAY	JOSEPH	vijay.joseph2030@example.com	7654321098	789 CHENNAI ST	1
4	AJITH	KUMAR	ajith.kumar2030@example.com	6543210987	101 CHENNAI ST	1
5	SURIYA	SIVAKUMAR	suriya.sivakumar2030@example.com	5432109876	202 CHENNAI ST	1
6	NAYANTHARA	DIANA	nayan.diana2030@example.com	4321098765	303 CHENNAI ST	1
7	TRISHA	KRISHNAN	trisha.krishnan2030@example.com	3210987654	404 CHENNAI ST	1
8	SAMANTHA	RUTH	samantha.ruth2030@example.com	2109876543	505 CHENNAI ST	1
9	KEERTHY	SURESH	keerthy.suresh2030@example.com	1098765432	606 CHENNAI ST	1
10	POOJA	HEGDE	pooja.hegde2030@example.com	0987654321	707 CHENNAI ST	1
11	DHANUSH	VENKATESH	dhanush.new2030@example.com	9876543211	909 CHENNAI ST	0
12	Sanjay	Kumar	sanjayieee01@gmail.com	9876543210	Krishanagiri	0
13	PRAGA	ESHWARAN	PRAG@gmail.com	1001001001	kabilar st	0

3.INVENTORY MANAGEMENT

Description: Tech Shop needs to manage product inventory, including adding new products, updating stock levels, and removing discontinued items.

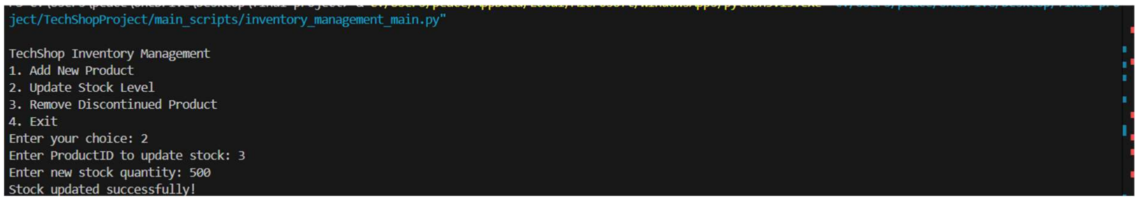
Task: Create an inventory management system with database connectivity. Implement features for adding new products, updating quantities, and handling discontinued products

CODE

import mysql.connector

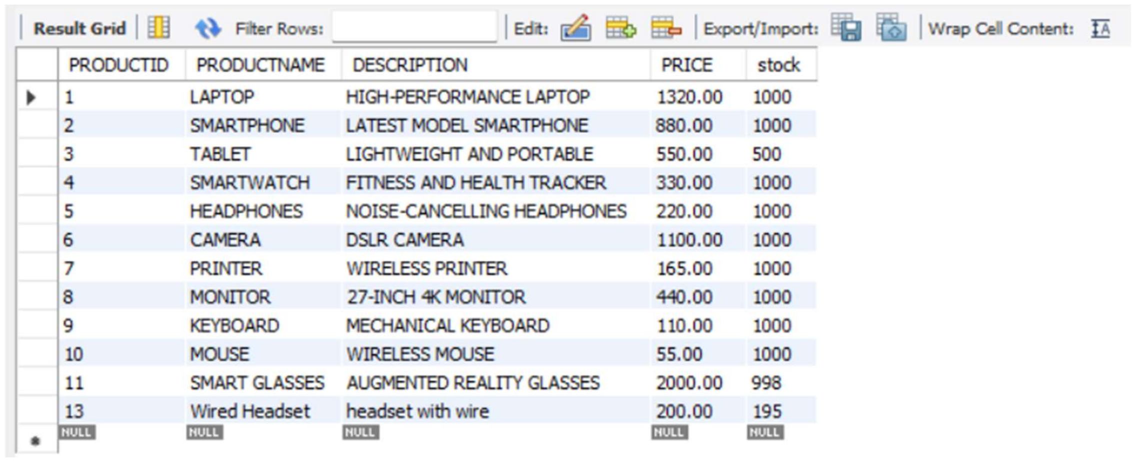
```
def add_product():
    """Add a new product to the inventory."""
    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="root",
        database="TechShop"
    )
    cursor = conn.cursor()
    name = input("Enter product name: ")
    description = input("Enter product description: ")
    price = float(input("Enter product price: "))
    stock = int(input("Enter product stock quantity: "))
    query = "INSERT INTO products (ProductName, Description, Price, stock) VALUES
(%s, %s, %s, %s)"
    values = (name, description, price, stock)
    cursor.execute(query, values)
    conn.commit()
    print("Product added successfully!")
    cursor.close()
    conn.close()


def update_stock():
    """Update stock level of an existing product."""
    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="root",
        database="TechShop"
    )
    cursor = conn.cursor()
    product_id = int(input("Enter ProductID to update stock: "))
    new_stock = int(input("Enter new stock quantity: "))
    query = "UPDATE products SET Stock = %s WHERE ProductID = %s"
    cursor.execute(query, (new_stock, product_id))
    conn.commit()
    print("Stock updated successfully!")
    cursor.close()
    conn.close()

def remove_product():
    """Remove a discontinued product from inventory."""
    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="root",
        database="TechShop"
    )
    cursor = conn.cursor()
    product_id = int(input("Enter ProductID to remove: "))
    query = "DELETE FROM products WHERE ProductID = %s"
    cursor.execute(query, (product_id,))
    conn.commit()
    print("Product removed successfully!")
    cursor.close()
    conn.close()


def main():
    while True:
        print("\nTechShop Inventory Management")
        print("1. Add New Product")
        print("2. Update Stock Level")
        print("3. Remove Discontinued Product")
        print("4. Exit")
        choice = input("Enter your choice: ")
        if choice == "1":
            add_product()
        elif choice == "2":
            update_stock()
        elif choice == "3":
            remove_product()
        elif choice == "4":
            print("Exiting... Thank you!")
            break
        else:
            print("Invalid choice. Please enter a valid option.")


if __name__ == "__main__":
    main()
```

OUTPUT



```
ject/TechShopProject/main_scripts/inventory_management_main.py"

TechShop Inventory Management
1. Add New Product
2. Update Stock Level
3. Remove Discontinued Product
4. Exit
Enter your choice: 2
Enter ProductID to update stock: 3
Enter new stock quantity: 500
Stock updated successfully!
```

MYSQL OUTPUT



	PRODUCTID	PRODUCTNAME	DESCRIPTION	PRICE	stock
▶	1	LAPTOP	HIGH-PERFORMANCE LAPTOP	1320.00	1000
	2	SMARTPHONE	LATEST MODEL SMARTPHONE	880.00	1000
	3	TABLET	LIGHTWEIGHT AND PORTABLE	550.00	500
	4	SMARTWATCH	FITNESS AND HEALTH TRACKER	330.00	1000
	5	HEADPHONES	NOISE-CANCELLING HEADPHONES	220.00	1000
	6	CAMERA	DSLR CAMERA	1100.00	1000
	7	PRINTER	WIRELESS PRINTER	165.00	1000
	8	MONITOR	27-INCH 4K MONITOR	440.00	1000
	9	KEYBOARD	MECHANICAL KEYBOARD	110.00	1000
	10	MOUSE	WIRELESS MOUSE	55.00	1000
	11	SMART GLASSES	AUGMENTED REALITY GLASSES	2000.00	998
	13	Wired Headset	headset with wire	200.00	195
*	NULL	NULL	NULL	NULL	NULL

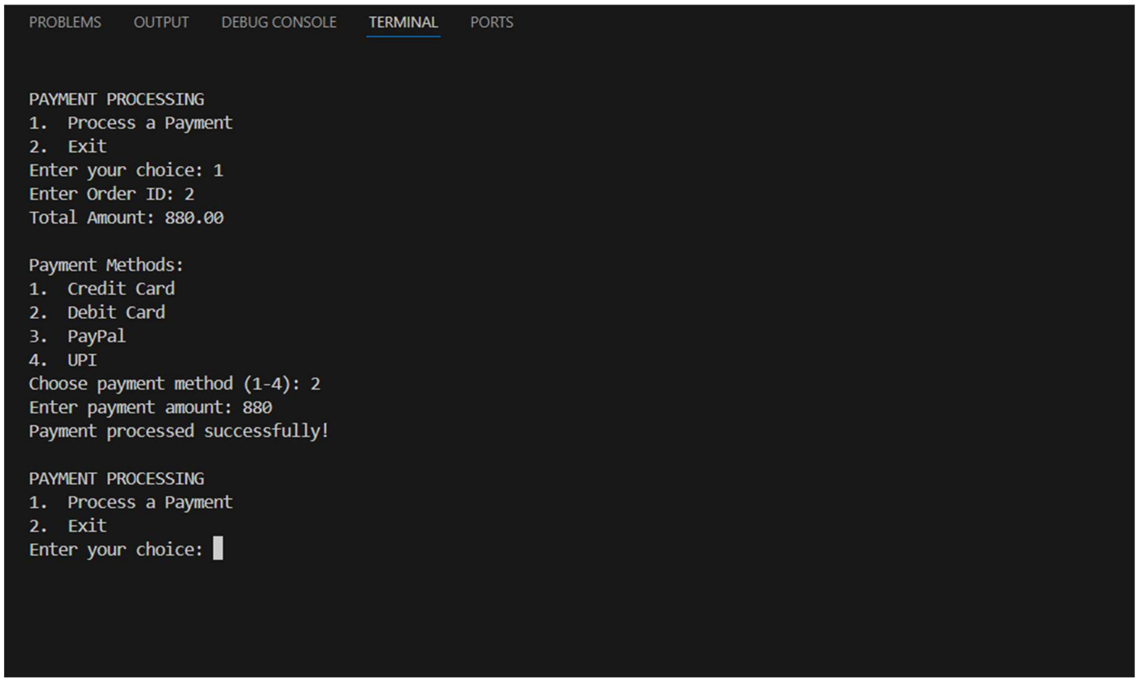
4. Payment Processing Description: When customers make payments for their orders, the payment details (e.g., payment method, amount) must be recorded in the database.

Task: Develop a payment processing system that interacts with the database to record payment transactions, validate payment information, and handle errors.

Code:

```
import mysql.connector


def get_db_connection():
    """Establish database connection"""
    try:
        conn = mysql.connector.connect(
            host="localhost",
            user="root",
            password="root",
            database="TechShop"
        )
        return conn
    except mysql.connector.Error as e:
        print(f"Error connecting to database: {e}")
        return None


def process_payment(cursor, conn):
    """Process a payment for an order"""
    order_id = input("Enter Order ID: ")
    # Check if order exists
    cursor.execute("SELECT TotalAmount, Status FROM orders WHERE OrderID =
%s",(order_id,))
    order = cursor.fetchone()
    if not order:
        print("Order not found.")
        return


    total_amount, status = order
    if status == "Paid":
        print("This order is already paid.")
        return


    print(f"Total Amount: {total_amount}")
    print("\nPayment Methods:")
    print("1.  Credit Card")
    print("2.  Debit Card")
    print("3.  PayPal")
    print("4.  UPI")
    method_choice = input("Choose payment method (1-4): ")
    payment_methods = {"1": "Credit Card", "2": "Debit Card", "3": "PayPal", "4": "UPI"}
    if method_choice not in payment_methods:
        print("Invalid payment method.")

        return


    payment_method = payment_methods[method_choice]

    amount_paid = float(input("Enter payment amount: "))

    if amount_paid != float(total_amount):

        print(f"Payment amount must match TotalAmount: {total_amount}")

        return


    # Update orders table with payment details

    cursor.execute("UPDATE orders SET PaymentMethod = %s, AmountPaid = %s, Status
= \

'Paid' WHERE OrderID = %s",

                (payment_method, amount_paid, order_id))

    conn.commit()

    print("Payment processed successfully!")


def main():

    """Main function for payment processing"""

    conn = get_db_connection()

    if not conn:

        return

    cursor = conn.cursor()

    while True:

        print("\nPAYMENT PROCESSING")

        print("1.  Process a Payment")

        print("2.  Exit")

        choice = input("Enter your choice: ")
        if choice == "1":
            process_payment(cursor, conn)
        elif choice == "2":
            print("Exiting Payment Processing...")
            break
        else:
            print("Invalid choice. Try again.")
    cursor.close()
    conn.close()

if __name__ == "__main__":
    main()
```

OUTPUT



```
PAYMENT PROCESSING
1.  Process a Payment
2.  Exit
Enter your choice: 1
Enter Order ID: 2
Total Amount: 880.00

Payment Methods:
1.  Credit Card
2.  Debit Card
3.  PayPal
4.  UPI
Choose payment method (1-4): 2
Enter payment amount: 880
Payment processed successfully!

PAYMENT PROCESSING
1.  Process a Payment
2.  Exit
Enter your choice:
```

5. Placing Customer Orders Description:

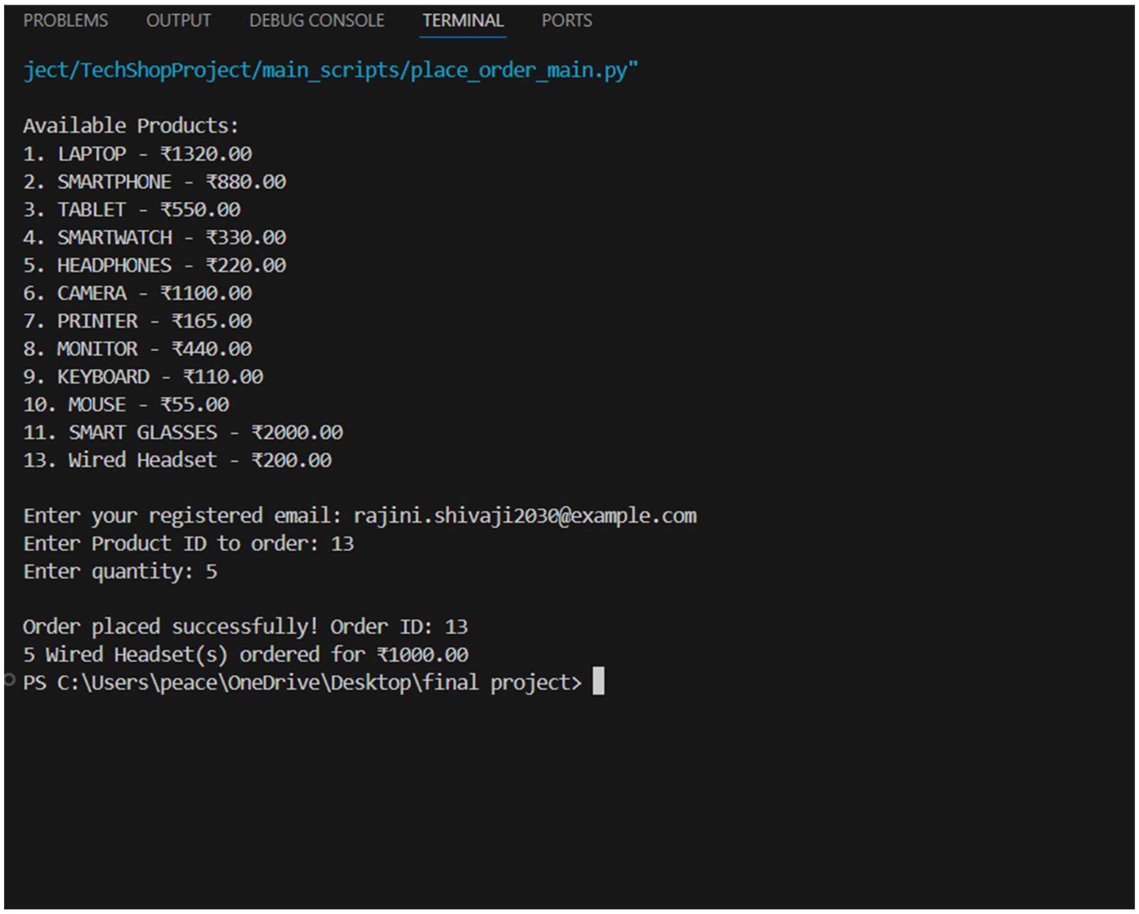
Customers browse the product catalog and place orders for products they want to purchase. The orders need to be stored in the database.

Task: Implement an order processing system. Use database connectivity to record customer orders, update product quantities in inventory, and calculate order totals

CODE

```
import mysql.connector

from datetime import datetime


# Database connection

def connect_db():

  try:

    conn = mysql.connector.connect(

      host="localhost",

      user="root",  # Change if needed

      password="root",  # Change if needed

      database="TechShop"  # Change if needed

    )

    return conn

  except mysql.connector.Error as err:

    print("Database connection error:", err)

    return None


# Display available products

def show_products(cursor):

  cursor.execute("SELECT ProductID, ProductName, Price FROM products")

  products = cursor.fetchall()

  print("\nAvailable Products:")

  for product in products:
        print(f"{product[0]}. {product[1]} - ₹{product[2]}")
    return products


# Get customer ID from email
def get_customer_id(cursor, email):
    cursor.execute("SELECT CustomerID FROM customers WHERE Email = %s", (email,))
    result = cursor.fetchone()
    return result[0] if result else None


# Place an order
def place_order(cursor, conn, customer_id, product_id, quantity):
    # Check stock availability
    cursor.execute("SELECT ProductName, Price, stock FROM products WHERE 
ProductID = %s", (product_id,))
    product = cursor.fetchone()
    if not product:
        print("Invalid product selection.")
        return
    product_name, price, stock = product
    if stock < quantity:
        print("Insufficient stock available!")
        return
    # Calculate total amount
    total_amount = price * quantity
    # Insert order
    order_date = datetime.now().date()
    cursor.execute("INSERT INTO orders (CustomerID, OrderDate, TotalAmount, Status) 
VALUES (%s, %s, %s, %s)",
                   (customer_id, order_date, total_amount, "Pending"))
    order_id = cursor.lastrowid

    # Update stock

    new_stock = stock - quantity

    cursor.execute("UPDATE products SET stock = %s WHERE ProductID = %s",
(new_stock, product_id))

    conn.commit()

    print(f"\nOrder placed successfully! Order ID: {order_id}")

    print(f"{quantity} {product_name}(s) ordered for ₹{total_amount}")


# Main function

def main():

    conn = connect_db()

    if conn is None:

        return

    cursor = conn.cursor()

    # Display products

    products = show_products(cursor)

    # Get customer email

    email = input("\nEnter your registered email: ")

    customer_id = get_customer_id(cursor, email)

    if not customer_id:

        print("Customer not found! Please register first.")

        return

    # Select product

    product_id = int(input("Enter Product ID to order: "))

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

    # Place order

    place_order(cursor, conn, customer_id, product_id, quantity)
    # Close connection
    cursor.close()
    conn.close()

if __name__ == "__main__":
    main()
```

OUTPUT



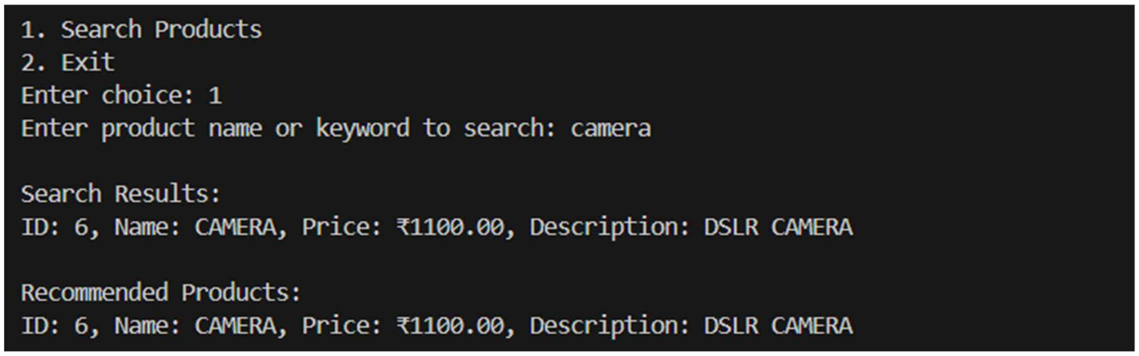
```
ject/TechShopProject/main_scripts/place_order_main.py"

Available Products:
1. LAPTOP - ₹1320.00
2. SMARTPHONE - ₹880.00
3. TABLET - ₹550.00
4. SMARTWATCH - ₹330.00
5. HEADPHONES - ₹220.00
6. CAMERA - ₹1100.00
7. PRINTER - ₹165.00
8. MONITOR - ₹440.00
9. KEYBOARD - ₹110.00
10. MOUSE - ₹55.00
11. SMART GLASSES - ₹2000.00
13. Wired Headset - ₹200.00

Enter your registered email: rajini.shivaji2030@example.com
Enter Product ID to order: 13
Enter quantity: 5

Order placed successfully! Order ID: 13
5 Wired Headset(s) ordered for ₹1000.00
PS C:\Users\peace\OneDrive\Desktop\final project>
```

6. Product Search and Recommendations Description: Customers should be able to search for products based on various criteria (e.g., name, category) and receive product recommendations. Task: Implement a product search and recommendation engine that uses database connectivity to retrieve relevant product information.

CODE

```
import mysql.connector


# Database Connection

def connect_db():

    try:

        conn = mysql.connector.connect(

            host="localhost",

            user="root",

            password="root",

            database="TechShop"

        )

        return conn

    except mysql.connector.Error as err:

        print(f"Error: {err}")

        return None


# Function to Search Products

def search_products():

    conn = connect_db()

    if not conn:

        return

    cursor = conn.cursor()

    search_query = input("Enter product name or keyword to search: ")

    query = """SELECT ProductID, ProductName, Price, Description

 FROM products

 WHERE ProductName LIKE %s OR Description LIKE %s"""

    cursor.execute(query, (f"%{search_query}%", f"%{search_query}%"))
    results = cursor.fetchall()
    if not results:
        print("No matching products found.")
    else:
        print("\nSearch Results:")
        for row in results:
            print(f"ID: {row[0]}, Name: {row[1]}, Price: ₹{row[2]}, Description: {row[3]}")
        # Fetch recommendations
        recommend_products(search_query, cursor)
    cursor.close()
    conn.close()

# Function to Recommend Products Based on Description
def recommend_products(search_query, cursor):
    query = """SELECT ProductID, ProductName, Price, Description
 FROM products
 WHERE Description LIKE %s
 LIMIT 3"""
    cursor.execute(query, (f"%{search_query}%",))
    recommendations = cursor.fetchall()
    if recommendations:
        print("\nRecommended Products:")
        for row in recommendations:
            print(f"ID: {row[0]}, Name: {row[1]}, Price: ₹{row[2]}, Description: {row[3]}")

# Main Menu
def main():
    while True:
        print("\n1. Search Products\n2. Exit")
        choice = input("Enter choice: ")
        if choice == "1":
            search_products()
        elif choice == "2":
            print("Exiting...")
            break
        else:
            print("Invalid choice. Try again.")


if __name__ == "__main__":
    main()
```

OUTPUT



```
1. Search Products
2. Exit
Enter choice: 1
Enter product name or keyword to search: camera

Search Results:
ID: 6, Name: CAMERA, Price: ₹1100.00, Description: DSLR CAMERA

Recommended Products:
ID: 6, Name: CAMERA, Price: ₹1100.00, Description: DSLR CAMERA
```

7. Sales Report

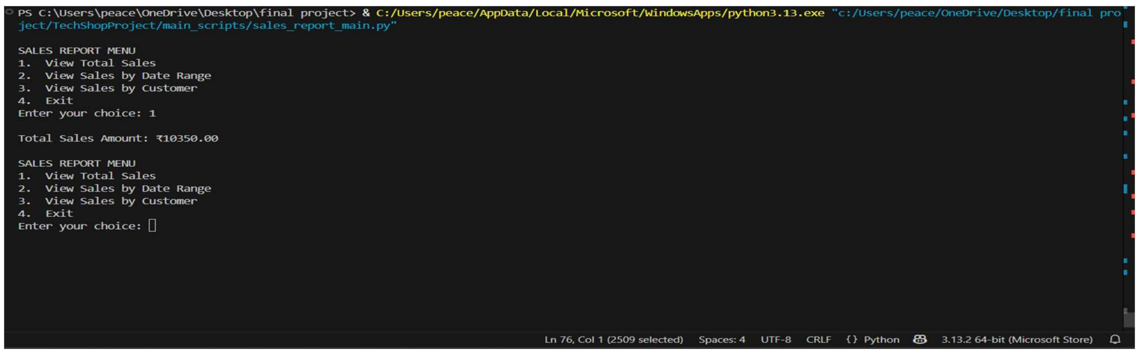
Description: Tech Shop needs a system to track and summarize sales data, including total revenue and top-selling products.
It helps analyze business performance and supports decision-making.

Task: Create a sales report module with database connectivity.
Include features to fetch sales data, calculate totals, and generate summary reports.

CODE

```
import mysql.connector

def get_db_connection():

    """Establish database connection"""

    try:

        conn = mysql.connector.connect(

            host="localhost",

            user="root",

            password="root",

            database="TechShop"

        )

        return conn

    except mysql.connector.Error as e:

        print(f"Error connecting to database: {e}")

        return None


def total_sales_report(cursor):

    """Fetch total sales amount"""

    cursor.execute("SELECT SUM(TotalAmount) FROM orders")

    total_sales = cursor.fetchone()[0]

    print(f"\nTotal Sales Amount: ₹{total_sales if total_sales else 0}")


def sales_by_date_range(cursor):

    """Fetch sales data between a specific date range"""

    start_date = input("Enter start date (YYYY-MM-DD): ")

    end_date = input("Enter end date (YYYY-MM-DD): ")

    cursor.execute(

        "SELECT OrderID, CustomerID, OrderDate, TotalAmount FROM orders WHERE \
OrderDate BETWEEN %s AND %s",
        (start_date, end_date)
    )
    orders = cursor.fetchall()
    print("\nSales Report (Date Range):")
    for order in orders:
        print(f"Order ID: {order[0]}, Customer ID: {order[1]}, Date: {order[2]}, Amount:
₹{order[3]}")

def sales_by_customer(cursor):
    """Fetch total sales made by a specific customer"""
    customer_id = input("Enter Customer ID: ")
    cursor.execute(
        "SELECT SUM(TotalAmount) FROM orders WHERE CustomerID = %s",
        (customer_id,)
    )
    total_sales = cursor.fetchone()[0]
    print(f"\nCustomer {customer_id} Total Purchases: ₹{total_sales if total_sales else 0}")

def main():
    """Main function to run the sales report system"""
    conn = get_db_connection()
    if not conn:
        return
    cursor = conn.cursor()
    while True:
        print("\nSALES REPORT MENU")
        print("1.  View Total Sales")
        print("2.  View Sales by Date Range")
        print("3.  View Sales by Customer")
        print("4.  Exit")
        choice = input("Enter your choice: ")
        if choice == "1":
            total_sales_report(cursor)
        elif choice == "2":
            sales_by_date_range(cursor)
        elif choice == "3":
            sales_by_customer(cursor)
        elif choice == "4":
            print("Exiting Sales Report...")
            break
        else:
            print("Invalid choice. Please try again.")
    cursor.close()
    conn.close()


if __name__ == "__main__":
    main()
```

OUTPUT



```
PS C:\Users\peace\OneDrive\Desktop\final project> & C:/Users/peace/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/peace/OneDrive/Desktop/final pro
ject/TechShopProject/main_scripts/sales_report_main.py"

SALES REPORT MENU
1.  View Total Sales
2.  View Sales by Date Range
3.  View Sales by Customer
4.  Exit
Enter your choice: 1

Total Sales Amount: ₹10350.00

SALES REPORT MENU
1.  View Total Sales
2.  View Sales by Date Range
3.  View Sales by Customer
4.  Exit
Enter your choice: 
```

Ln 76, Col 1 (2509 selected)   Spaces: 4   UTF-8   CRLF   {} Python   3.13.2 64-bit (Microsoft Store)

8.Tracking Order Status Description: Customers and employees need to track the status of their orders. The order status information is stored in the database.

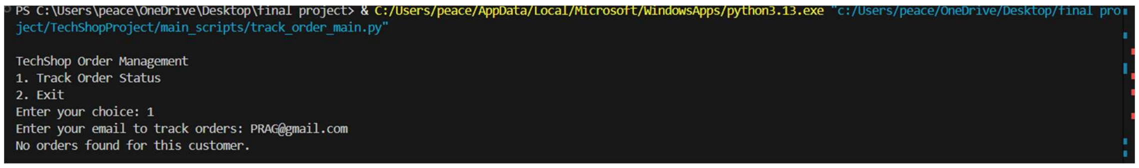
Task: Develop a feature that allows users to view the status of their orders. Implement database connectivity to retrieve and display order status information.

CODE

```
import mysql.connector


def track_order_status():
    """Retrieve and display the order status for a given customer."""
    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="root",
        database="TechShop"
    )
    cursor = conn.cursor()
    email = input("Enter your email to track orders: ")
    # Check if the customer exists
    cursor.execute("SELECT CustomerID FROM customers WHERE Email = %s", (email,))
    customer = cursor.fetchone()
    if not customer:
        print("No customer found with this email.")
        return


    customer_id = customer[0]
    # Retrieve order details
    cursor.execute("SELECT OrderID, OrderDate, TotalAmount, Status FROM orders
WHERE CustomerID = %s", (customer_id,))
```

```
    orders = cursor.fetchall()

    if not orders:

        print("No orders found for this customer.")

    else:

        print("\nYour Orders:")

        print("{:<10} {:<15} {:<10} {:<10}".format("OrderID", "OrderDate", "TotalAmount",
"Status"))

        print("-" * 50)

        for order in orders:

            print("{:<10} {:<15} {:<10} {:<10}".format(order[0], order[1], order[2], order[3]))

    cursor.close()

    conn.close()


def main():

    while True:

        print("\nTechShop Order Management")

        print("1. Track Order Status")

        print("2. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":

            track_order_status()

        elif choice == "2":

            print("Exiting... Thank you!")

            break

        else:

            print("Invalid choice. Please enter a valid option.")


if __name__ == "__main__":
    main()
```

OUTPUT



```
PS C:\Users\peace\OneDrive\Desktop\final project> & C:/Users/peace/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/peace/OneDrive/Desktop/final pro
ject/TechShopProject/main_scripts/track_order_main.py"

TechShop Order Management
1. Track Order Status
2. Exit
Enter your choice: 1
Enter your email to track orders: PRAG@gmail.com
No orders found for this customer.
```