Implementing OOPs Concept in Electronic Gadgets Database

1: Customer Registration

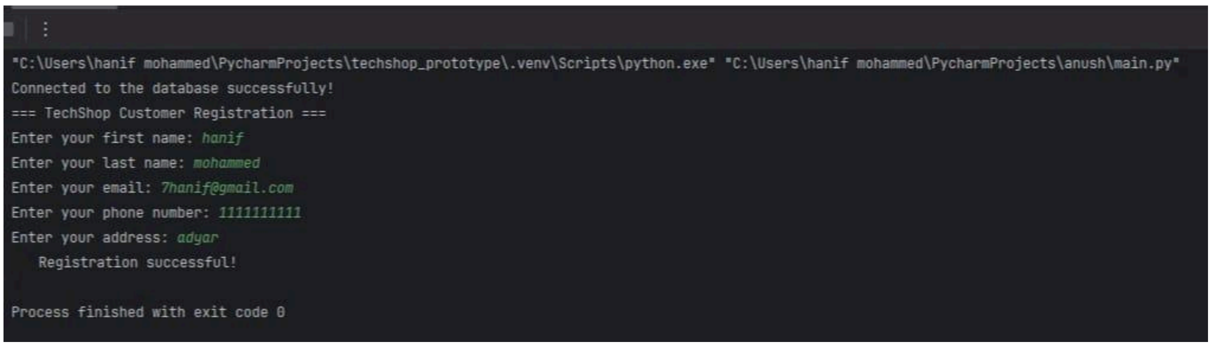
Description: When a new customer registers on the TechShop website, their information (e.g., name, email, phone) needs to be stored in the database.

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

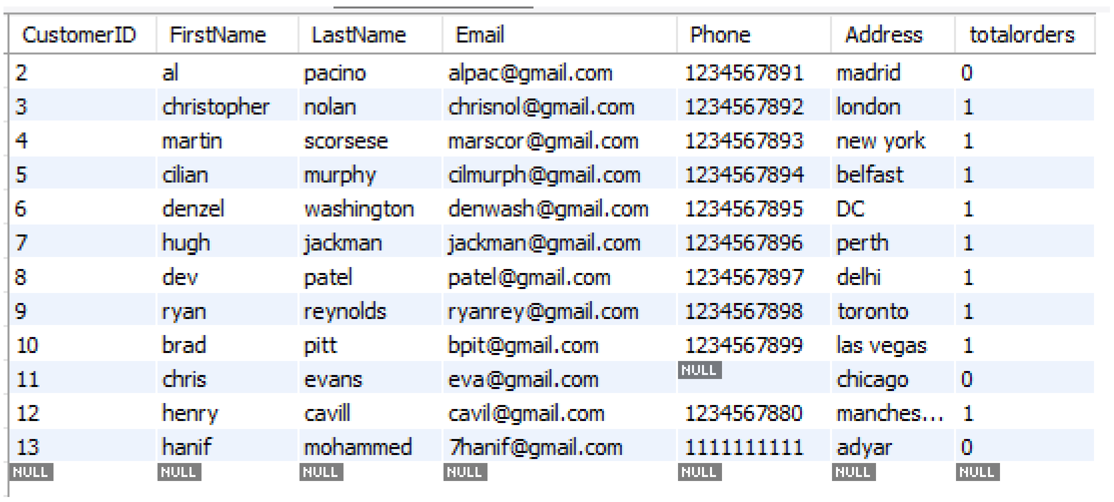
Code:

```
import pymysql
import re
from db_connection import connect_db

def validate_input(first_name, last_name, email, phone):
    """Validates customer details."""
    if not first_name or not last_name or not email or not phone:
        return "All fields are required."
    if not re.match(r'^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$', email):
        return "Invalid email format."
    if not phone.isdigit() or len(phone) < 10:
        return "Phone number must be numeric and at least 10 digits."
    return None

def register_customer():
    """Registers a new customer in the database."""

    print("=== TechShop Customer Registration ===")

    first_name = input("Enter your first name: ").strip()
    last_name = input("Enter your last name: ").strip()
    email = input("Enter your email: ").strip()
    phone = input("Enter your phone number: ").strip()
    address = input("Enter your address: ").strip()

    error = validate_input(first_name, last_name, email, phone)
    if error:
        print(f"Registration Failed: {error}")
        return

    try:
        connection = connect_db()
        if connection is None:
            print("Database connection failed.")
            return

        with connection.cursor() as cursor:
            cursor.execute("SELECT COUNT(*) FROM customers WHERE Email = %s",
(email,))
            if cursor.fetchone()[0] > 0:
                print("Error: Email already registered.")
                return

            insert_query = """
                INSERT INTO customers (FirstName, LastName, Email, Phone, Address,
totalorders)
                VALUES (%s, %s, %s, %s, %s, 0)
            """
            cursor.execute(insert_query, (first_name, last_name, email, phone, address))
            connection.commit()

            print("   Registration successful!")

    except pymysql.MySQLError as e:
        print(f"Registration Failed: {e}")
    finally:
        if connection:
            connection.close()

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

Output to enter new registration:



```
"C:\Users\hanif mohammed\PycharmProjects\techshop_prototype\.venv\Scripts\python.exe" "C:\Users\hanif mohammed\PycharmProjects\anush\main.py"
Connected to the database successfully!
=== TechShop Customer Registration ===
Enter your first name: hanif
Enter your last name: mohammed
Enter your email: 7hanif@gmail.com
Enter your phone number: 1111111111
Enter your address: adyar
   Registration successful!

Process finished with exit code 0
```



CustomerID	FirstName	LastName	Email	Phone	Address	totalorders
2	al	pacino	alpac@gmail.com	1234567891	madrid	0
3	christopher	nolan	chrisnol@gmail.com	1234567892	london	1
4	martin	scorsese	marscor@gmail.com	1234567893	new york	1
5	cilian	murphy	cilmurph@gmail.com	1234567894	belfast	1
6	denzel	washington	denwash@gmail.com	1234567895	DC	1
7	hugh	jackman	jackman@gmail.com	1234567896	perth	1
8	dev	patel	patel@gmail.com	1234567897	delhi	1
9	ryan	reynolds	ryanrey@gmail.com	1234567898	toronto	1
10	brad	pitt	bpit@gmail.com	1234567899	las vegas	1
11	chris	evans	eva@gmail.com	NULL	chicago	0
12	henry	cavill	cavil@gmail.com	1234567880	manches...	1
13	hanif	mohammed	7hanif@gmail.com	1111111111	adyar	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL

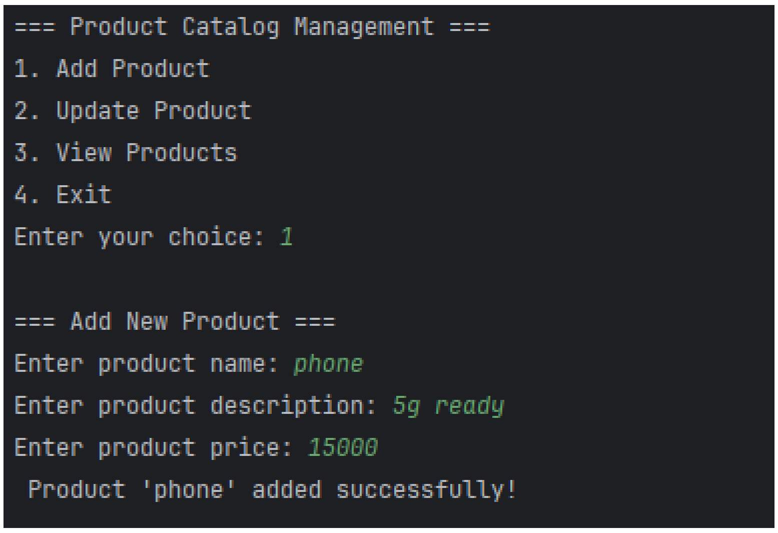
2: Product Catalog Management

Description: TechShop regularly updates its product catalog with new items and changes in product details (e.g., price, description). These changes need to be reflected in the database.
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.

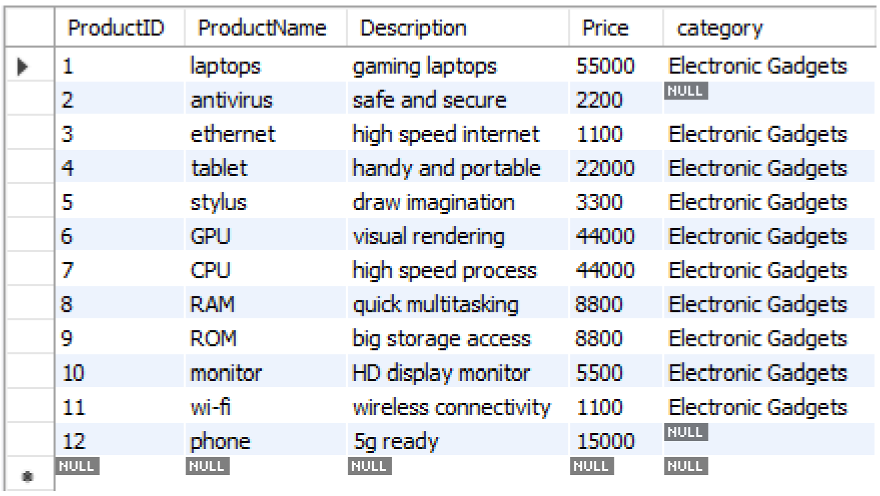
Code:

```
import pymysql
from db_connection import connect_db

def create_product_table():
    """Creates the products table if it doesn't exist."""
    conn = connect_db()
    if conn:
        try:
            with conn.cursor() as cursor:
                cursor.execute('''CREATE TABLE IF NOT EXISTS products (
                                    ProductID INT AUTO_INCREMENT PRIMARY KEY,
                                    Name VARCHAR(255) NOT NULL,
                                    Description TEXT,
                                    Price DECIMAL(10,2) NOT NULL)''')
                conn.commit()
        finally:
            conn.close()

def add_product():
    """Adds a new product to the database."""
    print("\n=== Add New Product ===")
    productname = input("Enter product name: ").strip()
    description = input("Enter product description: ").strip()
    price = input("Enter product price: ").strip()

    if not productname or not price:
        print("Error: Product name and price are required.")
        return

    try:
        price = float(price)
    except ValueError:
        print("Error: Invalid price format.")
        return

    conn = connect_db()
    if conn:
        try:
            with conn.cursor() as cursor:
                cursor.execute("INSERT INTO products (productname, description, Price)
VALUES (%s, %s, %s)",
                               (productname, description, price))
                conn.commit()
                print(f" Product '{productname}' added successfully!")
        except pymysql.MySQLError as e:
            print(f"Error: {e}")
        finally:
            conn.close()


def update_product():
    """Updates product details in the database."""
    print("\n=== Update Product Details ===")
    product_id = input("Enter product ID to update: ").strip()

    if not product_id.isdigit():
        print("Error: Invalid product ID.")
        return

    conn = connect_db()
    if conn:
        try:
            with conn.cursor() as cursor:
                cursor.execute("SELECT * FROM products WHERE ProductID = %s",
(product_id,))
                product = cursor.fetchone()
                if not product:
                    print("Error: Product ID not found.")
                    return
                new_name = input(f"Enter new name [{product[1]}]: ").strip() or product[1]
                new_description = input(f"Enter new description [{product[2]}]: ").strip() or
product[2]
                new_price = input(f"Enter new price [{product[3]}]: ").strip() or product[3]

                cursor.execute("UPDATE products SET productname = %s, description = %s,
price = %s WHERE ProductID = %s",
                               (new_name, new_description, new_price, product_id))
                conn.commit()
                print(f" Product '{new_name}' updated successfully!")

        except pymysql.MySQLError as e:
            print(f"Error: {e}")
        finally:
            conn.close()

def view_products():
    """Displays the product catalog."""
    print("\n=== Product Catalog ===")

    conn = connect_db()
    if conn:
        try:
            with conn.cursor() as cursor:
                cursor.execute("SELECT * FROM products")
                products = cursor.fetchall()
                if not products:
                    print("No products found.")
                    return

                for product in products:
                    print(f"ID: {product[0]}, Name: {product[1]}, Price: ${product[3]:.2f}, Description:
{product[2]}")

        except pymysql.MySQLError as e:
            print(f"Error: {e}")
        finally:
            conn.close()

def main():
    """Main function to manage product catalog."""
    create_product_table()

    while True:
        print("\n=== Product Catalog Management ===")
        print("1. Add Product")
        print("2. Update Product")
        print("3. View Products")
        print("4. Exit")

        choice = input("Enter your choice: ").strip()

        if choice == "1":
            add_product()
        elif choice == "2":
            update_product()
        elif choice == "3":
            view_products()
        elif choice == "4":
            print("Exiting Product Catalog Management.")
            break
        else:
            print("Invalid choice. Please try again.")

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

Output : To add product

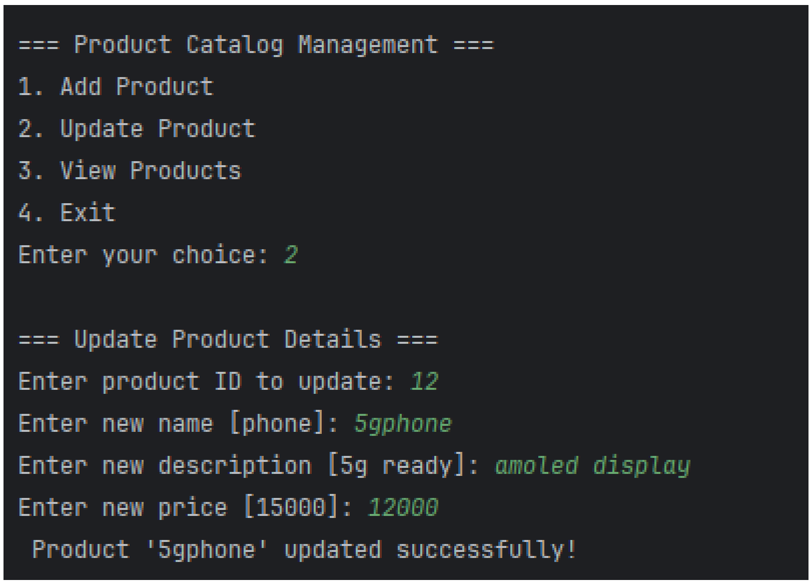


```
=== Product Catalog Management ===
1. Add Product
2. Update Product
3. View Products
4. Exit
Enter your choice: 1

=== Add New Product ===
Enter product name: phone
Enter product description: 5g ready
Enter product price: 15000
 Product 'phone' added successfully!
```

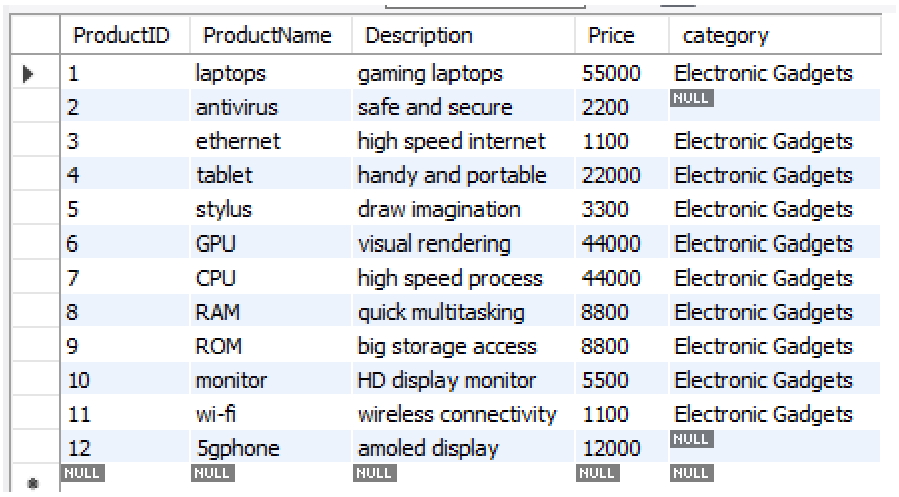


ProductID	ProductName	Description	Price	category
1	laptops	gaming laptops	55000	Electronic Gadgets
2	antivirus	safe and secure	2200	NULL
3	ethernet	high speed internet	1100	Electronic Gadgets
4	tablet	handy and portable	22000	Electronic Gadgets
5	stylus	draw imagination	3300	Electronic Gadgets
6	GPU	visual rendering	44000	Electronic Gadgets
7	CPU	high speed process	44000	Electronic Gadgets
8	RAM	quick multitasking	8800	Electronic Gadgets
9	ROM	big storage access	8800	Electronic Gadgets
10	monitor	HD display monitor	5500	Electronic Gadgets
11	wi-fi	wireless connectivity	1100	Electronic Gadgets
12	phone	5g ready	15000	NULL
NULL	NULL	NULL	NULL	NULL

Output: To update product

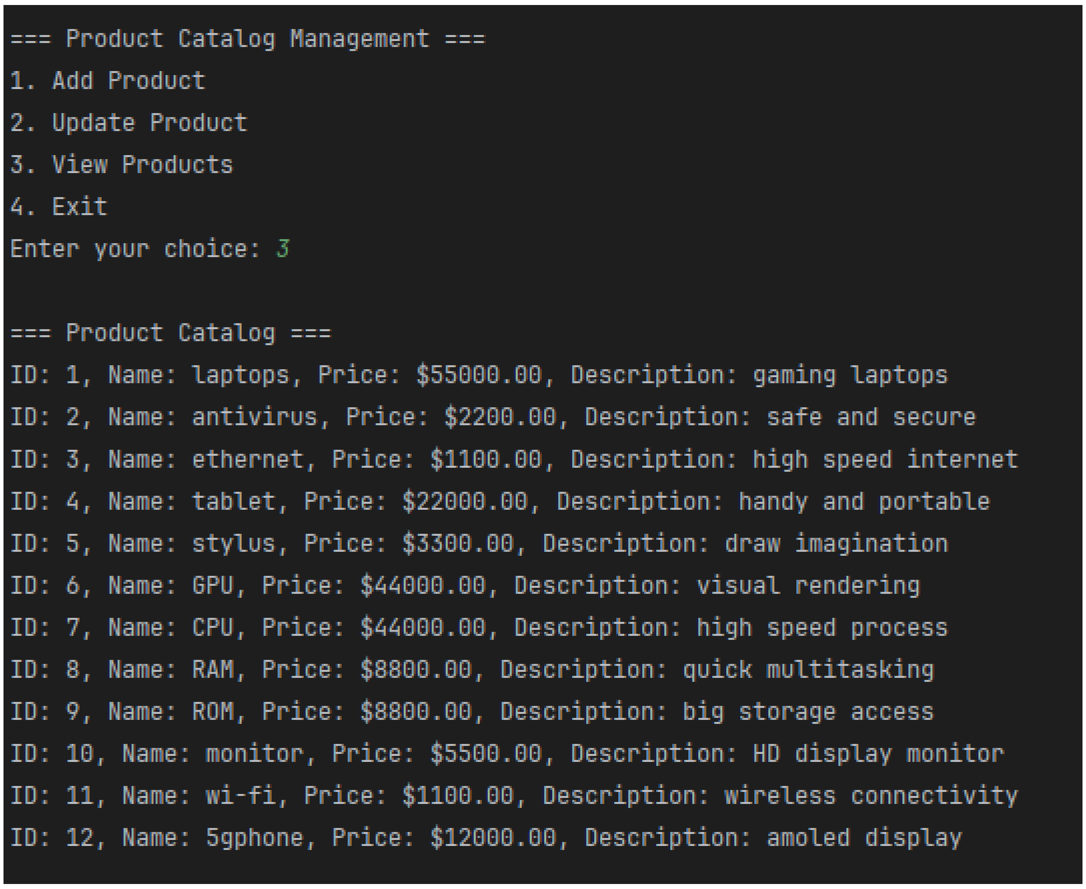


```
=== Product Catalog Management ===
1. Add Product
2. Update Product
3. View Products
4. Exit
Enter your choice: 2

=== Update Product Details ===
Enter product ID to update: 12
Enter new name [phone]: 5gphone
Enter new description [5g ready]: amoled display
Enter new price [15000]: 12000
 Product '5gphone' updated successfully!
```



ProductID	ProductName	Description	Price	category
1	laptops	gaming laptops	55000	Electronic Gadgets
2	antivirus	safe and secure	2200	NULL
3	ethernet	high speed internet	1100	Electronic Gadgets
4	tablet	handy and portable	22000	Electronic Gadgets
5	stylus	draw imagination	3300	Electronic Gadgets
6	GPU	visual rendering	44000	Electronic Gadgets
7	CPU	high speed process	44000	Electronic Gadgets
8	RAM	quick multitasking	8800	Electronic Gadgets
9	ROM	big storage access	8800	Electronic Gadgets
10	monitor	HD display monitor	5500	Electronic Gadgets
11	wi-fi	wireless connectivity	1100	Electronic Gadgets
12	5gphone	amoled display	12000	NULL
NULL	NULL	NULL	NULL	NULL

Output: To view products:



```
=== Product Catalog Management ===
1. Add Product
2. Update Product
3. View Products
4. Exit
Enter your choice: 3

=== Product Catalog ===
ID: 1, Name: laptops, Price: $55000.00, Description: gaming laptops
ID: 2, Name: antivirus, Price: $2200.00, Description: safe and secure
ID: 3, Name: ethernet, Price: $1100.00, Description: high speed internet
ID: 4, Name: tablet, Price: $22000.00, Description: handy and portable
ID: 5, Name: stylus, Price: $3300.00, Description: draw imagination
ID: 6, Name: GPU, Price: $44000.00, Description: visual rendering
ID: 7, Name: CPU, Price: $44000.00, Description: high speed process
ID: 8, Name: RAM, Price: $8800.00, Description: quick multitasking
ID: 9, Name: ROM, Price: $8800.00, Description: big storage access
ID: 10, Name: monitor, Price: $5500.00, Description: HD display monitor
ID: 11, Name: wi-fi, Price: $1100.00, Description: wireless connectivity
ID: 12, Name: 5gphone, Price: $12000.00, Description: amoled display
```

3: Placing Customer Orders

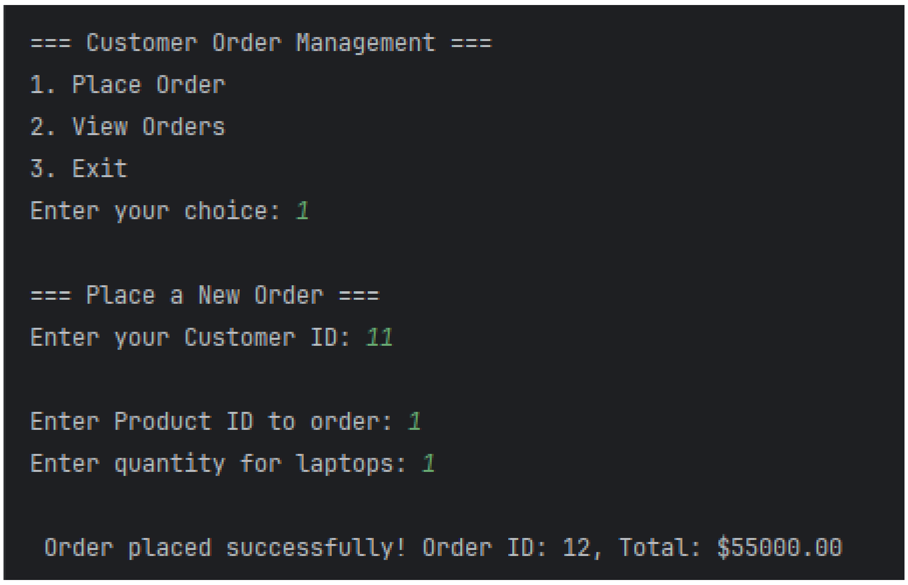
Description: Customers browse the product catalog and place orders for products they want to purchase. The orders need to be stored in the database.

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

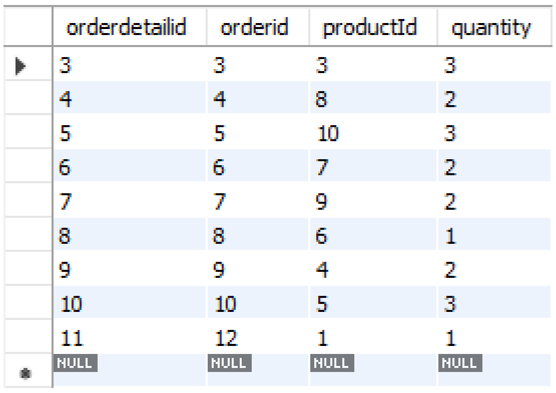
Code:

```
import pymysql
from db_connection import connect_db

def place_order():
    """Processes a new customer order."""
    print("\n=== Place a New Order ===")
    customer_id = input("Enter your Customer ID: ").strip()

    if not customer_id.isdigit():
        print(" Error: Invalid Customer ID.")
        return
    conn = connect_db()
    if conn:
        try:
            with conn.cursor() as cursor:
                cursor.execute("SELECT * FROM customers WHERE CustomerID = %s",
(customer_id,))
                customer = cursor.fetchone()
                if not customer:
                    print(" Error: Customer not found.")
                    return

                product_id = input("\nEnter Product ID to order: ").strip()
                if not product_id.isdigit():
                    print(" Error: Invalid Product ID.")
                    return

                cursor.execute("SELECT ProductName, Price FROM products WHERE ProductID
= %s", (product_id,))
                product = cursor.fetchone()

                if not product:
                    print(" Error: Product not found.")
                    return

                product_name, product_price = product
                quantity = input(f"Enter quantity for {product_name}: ").strip()

                if not quantity.isdigit() or int(quantity) <= 0:
                    print(" Error: Invalid quantity.")
                    return

                quantity = int(quantity)
                subtotal = quantity * float(product_price)
                total_cost = subtotal

                cursor.execute("INSERT INTO orders (CustomerID, OrderDate, TotalAmount)
VALUES (%s, NOW(), %s)",
                               (customer_id, total_cost))
                order_id = cursor.lastrowid

                cursor.execute("INSERT INTO orderdetails (OrderID, ProductID, Quantity)
VALUES (%s, %s, %s)",
                               (order_id, product_id, quantity))

                conn.commit()
                print(f"\n Order placed successfully! Order ID: {order_id}, Total: ${total_cost:.2f}")
        except pymysql.MySQLError as e:
            print(f" Error: {e}")
        finally:
            conn.close()


def view_orders():
    """Displays all customer orders."""
    print("\n=== View Orders ===")

    conn = connect_db()
    if conn:
        try:
            with conn.cursor() as cursor:
                cursor.execute("SELECT * FROM orders")
                orders = cursor.fetchall()

                if not orders:
                    print("No orders found.")
                    return

                for order in orders:
                    print(f"\nOrder ID: {order[0]}, Customer ID: {order[1]}, Total: ${order[3]:.2f}, Date:
{order[2]}")
                    cursor.execute("SELECT p.ProductName, od.Quantity FROM orderdetails od "
                                   "JOIN products p ON od.ProductID = p.ProductID WHERE od.OrderID
= %s", (order[0],))
                    items = cursor.fetchall()

                    for item in items:
                        print(f"  - {item[0]} | Quantity: {item[1]}")

        except pymysql.MySQLError as e:
            print(f" Error: {e}")
        finally:
            conn.close()


def main():
    """Main function for customer order processing."""
    while True:
        print("\n=== Customer Order Management ===")
        print("1. Place Order")
        print("2. View Orders")
        print("3. Exit")

        choice = input("Enter your choice: ").strip()
        if choice == "1":
            place_order()
        elif choice == "2":
            view_orders()
        elif choice == "3":
            print("Exiting Customer Order Management.")
            break
        else:
            print("Invalid choice. Please try again.")

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

Output: To place order

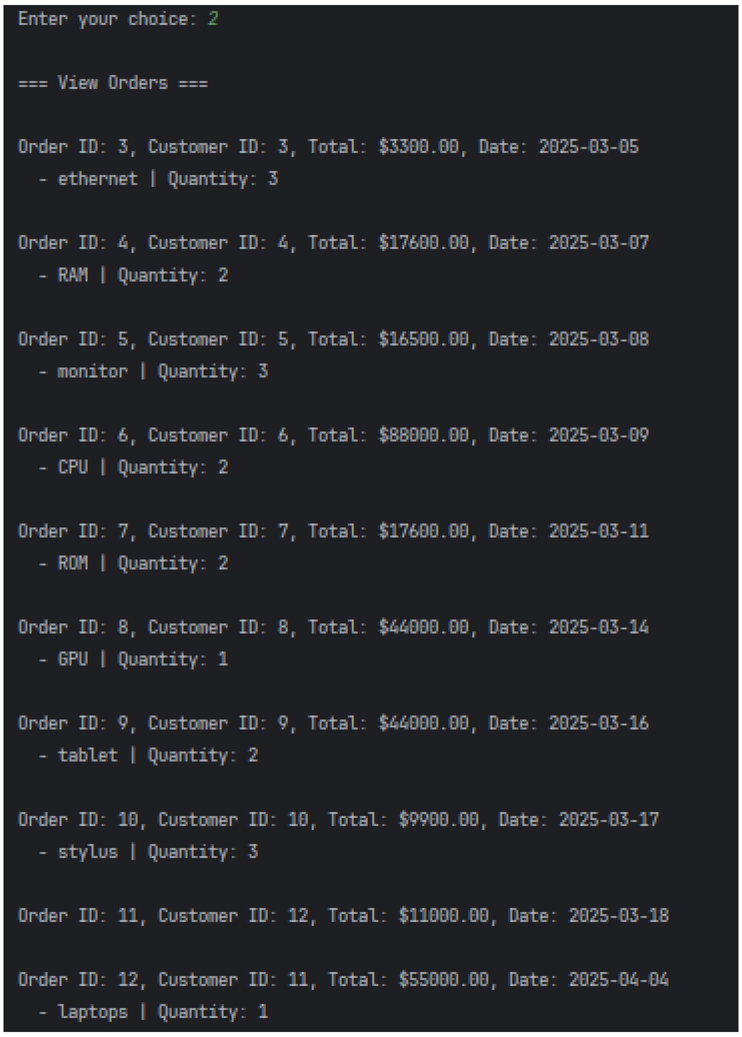


```
=== Customer Order Management ===
1. Place Order
2. View Orders
3. Exit
Enter your choice: 1

=== Place a New Order ===
Enter your Customer ID: 11

Enter Product ID to order: 1
Enter quantity for laptops: 1

 Order placed successfully! Order ID: 12, Total: $55000.00
```



orderdetailid	orderid	productId	quantity
3	3	3	3
4	4	8	2
5	5	10	3
6	6	7	2
7	7	9	2
8	8	6	1
9	9	4	2
10	10	5	3
11	12	1	1
NULL	NULL	NULL	NULL

Output: To view Order



```
Enter your choice: 2

=== View Orders ===

Order ID: 3, Customer ID: 3, Total: $3300.00, Date: 2025-03-05
  - ethernet | Quantity: 3

Order ID: 4, Customer ID: 4, Total: $17600.00, Date: 2025-03-07
  - RAM | Quantity: 2

Order ID: 5, Customer ID: 5, Total: $16500.00, Date: 2025-03-08
  - monitor | Quantity: 3

Order ID: 6, Customer ID: 6, Total: $88000.00, Date: 2025-03-09
  - CPU | Quantity: 2

Order ID: 7, Customer ID: 7, Total: $17600.00, Date: 2025-03-11
  - ROM | Quantity: 2

Order ID: 8, Customer ID: 8, Total: $44000.00, Date: 2025-03-14
  - GPU | Quantity: 1

Order ID: 9, Customer ID: 9, Total: $44000.00, Date: 2025-03-16
  - tablet | Quantity: 2

Order ID: 10, Customer ID: 10, Total: $9900.00, Date: 2025-03-17
  - stylus | Quantity: 3

Order ID: 11, Customer ID: 12, Total: $11000.00, Date: 2025-03-18

Order ID: 12, Customer ID: 11, Total: $55000.00, Date: 2025-04-04
  - laptops | Quantity: 1
```

4: Tracking Order Status

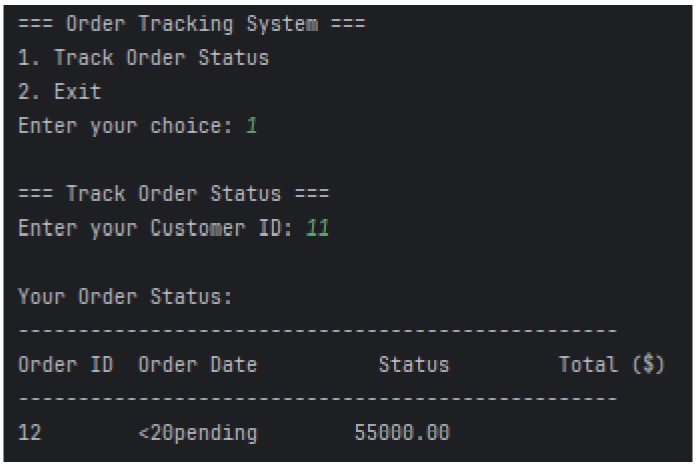
Description: Customers and employees need to track the status of their orders. The order status information is stored in the database.

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 pymysql
from db_connection import connect_db

def track_order_status():
    """Allows users to view the status of their orders."""
    print("\n=== Track Order Status ===")
    customer_id = input("Enter your Customer ID: ").strip()

    if not customer_id.isdigit():
        print(" Error: Invalid Customer ID.")
        return

    conn = connect_db()
    if conn:
        try:
            with conn.cursor() as cursor:
                cursor.execute("SELECT OrderID, OrderDate, OrderStatus, TotalAmount FROM
orders WHERE CustomerID = %s",
                               (customer_id,))
                orders = cursor.fetchall()

                if not orders:
                    print(" No orders found for this customer.")
                    return

                print("\nYour Order Status:")
                print("-" * 50)
                print(f"{'Order ID':<10}{'Order Date':<20}{'Status':<15}{'Total ($)'}")
                print("-" * 50)

                for order in orders:
                    print(f"{order[0]:<10}{order[1]:<20}{order[2]:<15}{order[3]:.2f}")

        except pymysql.MySQLError as e:
            print(f" Error: {e}")
        finally:
            conn.close()

def main():
    """Main function to track order status."""
    while True:
        print("\n=== Order Tracking System ===")
        print("1. Track Order Status")
        print("2. Exit")

        choice = input("Enter your choice: ").strip()

        if choice == "1":
            track_order_status()
        elif choice == "2":
            print("Exiting Order Tracking System.")
            break
        else:
            print("Invalid choice. Please try again.")

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

Output: To track order status



```
=== Order Tracking System ===
1. Track Order Status
2. Exit
Enter your choice: 1

=== Track Order Status ===
Enter your Customer ID: 11

Your Order Status:
--------------------------------------------------
Order ID  Order Date          Status         Total ($)
--------------------------------------------------
12        <20pending        55000.00
```

5: Inventory Management

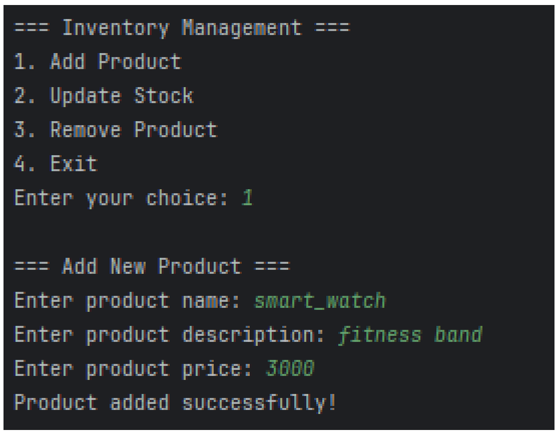
Description: TechShop needs to manage product inventory, including adding new products, updating stock levels, and removing discontinued items.

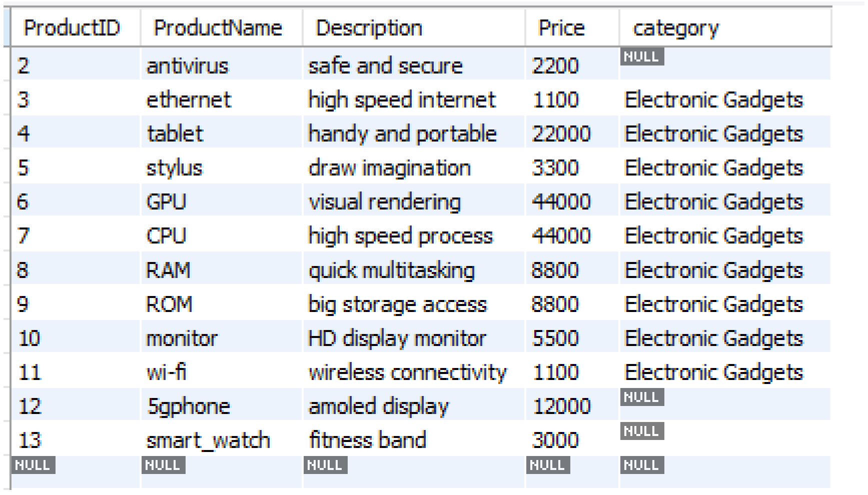
Task: Create an inventory management system with database connectivity. Implement features for adding new products, updating quantities, and handling discontinued products.

Code:

```
import pymysql
from db_connection import connect_db

def add_product():
    """Adds a new product to the inventory."""
    print("\n=== Add New Product ===")
    product_name = input("Enter product name: ").strip()
    description = input("Enter product description: ").strip()
    price = input("Enter product price: ").strip()

    if not price.isdigit() or int(price) <= 0:
        print("Error: Invalid price.")
        return

    conn = connect_db()
    if conn:
        try:
            with conn.cursor() as cursor:
                cursor.execute("INSERT INTO products (ProductName, Description, Price)
VALUES (%s, %s, %s)",
                               (product_name, description, price))
                conn.commit()
                print("Product added successfully!")

        except pymysql.MySQLError as e:
            print(f"Error: {e}")
        finally:
            conn.close()

def update_stock():
    """Updates stock quantity for a product."""
    print("\n=== Update Stock ===")
    product_id = input("Enter Product ID: ").strip()
    quantity = input("Enter new stock quantity: ").strip()

    if not product_id.isdigit() or not quantity.isdigit() or int(quantity) < 0:
        print("Error: Invalid input.")
        return

    conn = connect_db()
    if conn:
        try:
            with conn.cursor() as cursor:
                cursor.execute("UPDATE orderdetails SET Quantity = %s WHERE ProductID =
%s",
                               (quantity, product_id))
                conn.commit()
                print("Stock updated successfully!")

        except pymysql.MySQLError as e:
            print(f"Error: {e}")
        finally:
            conn.close()

def remove_product():
    """Removes a discontinued product."""
    print("\n=== Remove Product ===")
    product_id = input("Enter Product ID to remove: ").strip()

    if not product_id.isdigit():
        print("Error: Invalid Product ID.")
        return

    conn = connect_db()
    if conn:
        try:
            with conn.cursor() as cursor:
                cursor.execute("DELETE FROM products WHERE ProductID = %s", (product_id,))
                conn.commit()
                print("Product removed successfully!")

        except pymysql.MySQLError as e:
            print(f"Error: {e}")
        finally:
            conn.close()

def main():
    """Inventory Management System."""
    while True:
        print("\n=== Inventory Management ===")
        print("1. Add Product")
        print("2. Update Stock")
        print("3. Remove Product")
        print("4. Exit")

        choice = input("Enter your choice: ").strip()

        if choice == "1":
            add_product()
        elif choice == "2":
            update_stock()
        elif choice == "3":
            remove_product()
        elif choice == "4":
            print("Exiting Inventory Management.")
            break
        else:
            print("Invalid choice. Please try again.")

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

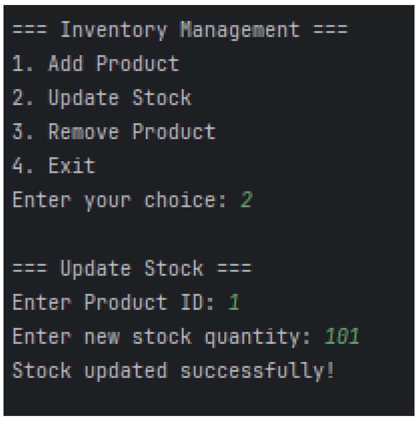
Output to add product:

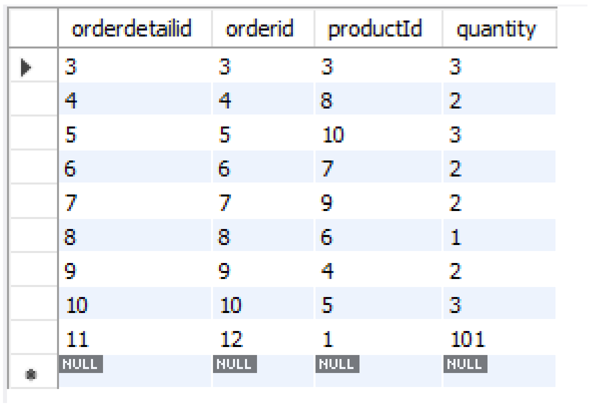

```
=== Inventory Management ===
1. Add Product
2. Update Stock
3. Remove Product
4. Exit
Enter your choice: 1

=== Add New Product ===
Enter product name: smart_watch
Enter product description: fitness band
Enter product price: 3000
Product added successfully!
```



ProductID	ProductName	Description	Price	category
2	antivirus	safe and secure	2200	NULL
3	ethernet	high speed internet	1100	Electronic Gadgets
4	tablet	handy and portable	22000	Electronic Gadgets
5	stylus	draw imagination	3300	Electronic Gadgets
6	GPU	visual rendering	44000	Electronic Gadgets
7	CPU	high speed process	44000	Electronic Gadgets
8	RAM	quick multitasking	8800	Electronic Gadgets
9	ROM	big storage access	8800	Electronic Gadgets
10	monitor	HD display monitor	5500	Electronic Gadgets
11	wi-fi	wireless connectivity	1100	Electronic Gadgets
12	5gphone	amoled display	12000	NULL
13	smart_watch	fitness band	3000	NULL
NULL	NULL	NULL	NULL	NULL

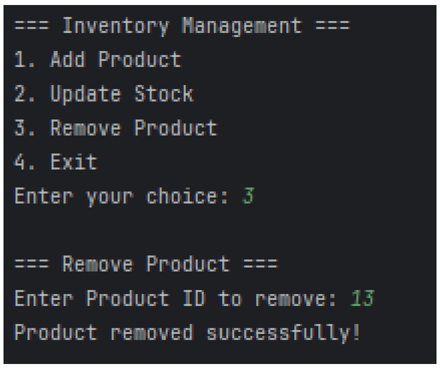
Output to update stock:


```
=== Inventory Management ===
1. Add Product
2. Update Stock
3. Remove Product
4. Exit
Enter your choice: 2

=== Update Stock ===
Enter Product ID: 1
Enter new stock quantity: 101
Stock updated successfully!
```

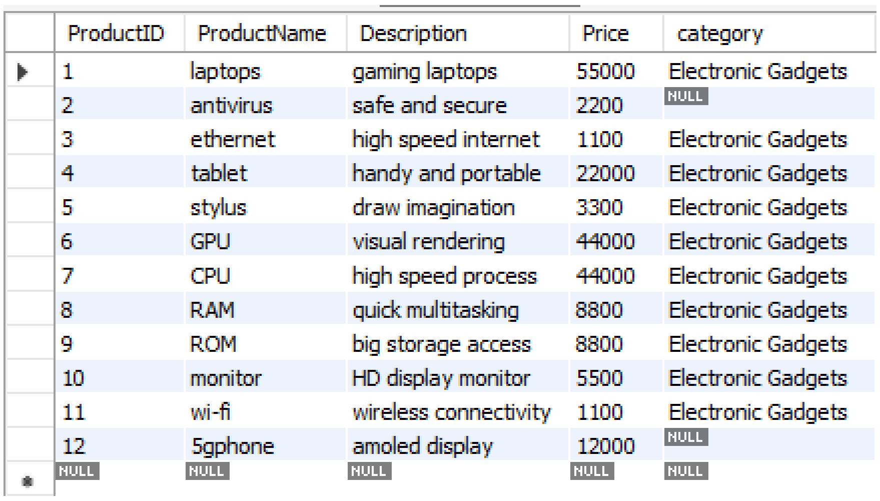


orderdetailid	orderid	productId	quantity
3	3	3	3
4	4	8	2
5	5	10	3
6	6	7	2
7	7	9	2
8	8	6	1
9	9	4	2
10	10	5	3
11	12	1	101
NULL	NULL	NULL	NULL

Output to remove product:



```
=== Inventory Management ===
1. Add Product
2. Update Stock
3. Remove Product
4. Exit
Enter your choice: 3

=== Remove Product ===
Enter Product ID to remove: 13
Product removed successfully!
```



ProductID	ProductName	Description	Price	category
1	laptops	gaming laptops	55000	Electronic Gadgets
2	antivirus	safe and secure	2200	NULL
3	ethernet	high speed internet	1100	Electronic Gadgets
4	tablet	handy and portable	22000	Electronic Gadgets
5	stylus	draw imagination	3300	Electronic Gadgets
6	GPU	visual rendering	44000	Electronic Gadgets
7	CPU	high speed process	44000	Electronic Gadgets
8	RAM	quick multitasking	8800	Electronic Gadgets
9	ROM	big storage access	8800	Electronic Gadgets
10	monitor	HD display monitor	5500	Electronic Gadgets
11	wi-fi	wireless connectivity	1100	Electronic Gadgets
12	5gphone	amoled display	12000	NULL
NULL	NULL	NULL	NULL	NULL

6: Sales Reporting

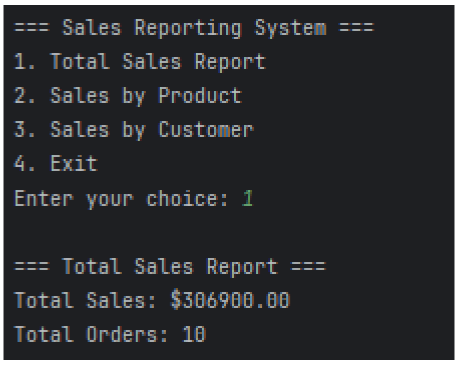
Description: TechShop management requires sales reports for business analysis. The sales data is stored in the database.

Task: Design and implement a reporting system that retrieves sales data from the database and generates reports based on specified criteria.

Code:

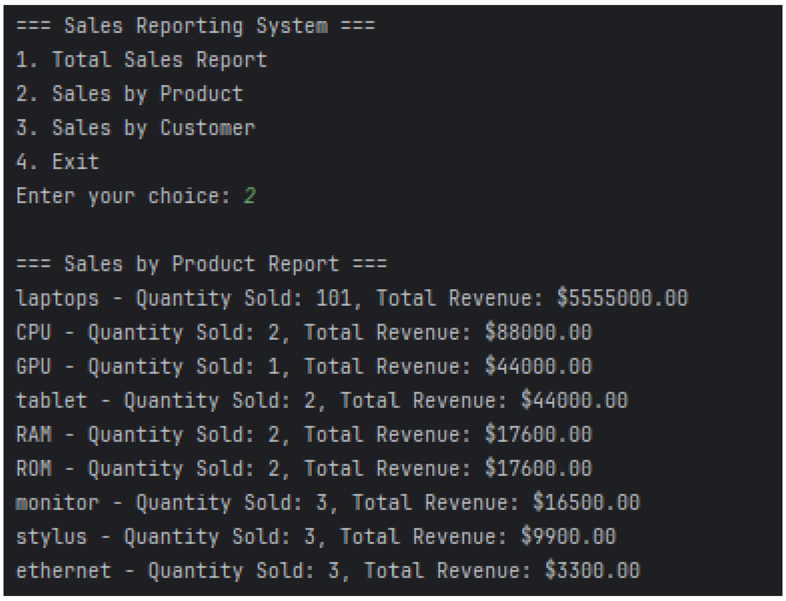
```
import pymysql
from db_connection import connect_db


def total_sales_report():
    """Generates a report of total sales."""
    print("\n=== Total Sales Report ===")
    conn = connect_db()

    if conn:
        try:
            with conn.cursor() as cursor:
                cursor.execute("""
                    SELECT SUM(o.TotalAmount) AS TotalSales, COUNT(o.OrderID) AS
TotalOrders
                    FROM orders o
                """)
                result = cursor.fetchone()

                if result:
                    total_sales, total_orders = result
                    print(f"Total Sales: ${total_sales:.2f}")
                    print(f"Total Orders: {total_orders}")
                else:
                    print("No sales data available.")

        except pymysql.MySQLError as e:
            print(f"Error: {e}")
        finally:
            conn.close()


def sales_by_product():
    """Generates a report of sales per product."""
    print("\n=== Sales by Product Report ===")
    conn = connect_db()

    if conn:
        try:
            with conn.cursor() as cursor:
                cursor.execute("""
                    SELECT p.ProductName, SUM(od.Quantity) AS TotalQuantity,
SUM(od.Quantity * p.Price) AS TotalRevenue
                    FROM orderdetails od
                    JOIN products p ON od.ProductID = p.ProductID
                    GROUP BY p.ProductName
                    ORDER BY TotalRevenue DESC
            """)
            results = cursor.fetchall()

            if results:
                for row in results:
                    print(f"{row[0]} - Quantity Sold: {row[1]}, Total Revenue: ${row[2]:.2f}")
            else:
                print("No sales data available.")

        except pymysql.MySQLError as e:
            print(f"Error: {e}")
        finally:
            conn.close()


def sales_by_customer():
    """Generates a report of sales per customer."""
    print("\n=== Sales by Customer Report ===")
    conn = connect_db()

    if conn:
        try:
            with conn.cursor() as cursor:
                cursor.execute("""
                    SELECT c.CustomerID, COUNT(o.OrderID) AS TotalOrders,
SUM(o.TotalAmount) AS TotalSpent
                    FROM orders o
                    JOIN customers c ON o.CustomerID = c.CustomerID
                    GROUP BY c.CustomerID
                    ORDER BY TotalSpent DESC
                """)
                results = cursor.fetchall()

                if results:
                    for row in results:
                        print(f"Customer ID: {row[0]} - Orders: {row[1]}, Total Spent: ${row[2]:.2f}")
                else:
                    print("No sales data available.")

        except pymysql.MySQLError as e:
            print(f"Error: {e}")
        finally:
            conn.close()


def main():
    """Sales Reporting System."""
    while True:
        print("\n=== Sales Reporting System ===")
        print("1. Total Sales Report")
        print("2. Sales by Product")
        print("3. Sales by Customer")
        print("4. Exit")

        choice = input("Enter your choice: ").strip()

        if choice == "1":
            total_sales_report()
        elif choice == "2":
            sales_by_product()
        elif choice == "3":
            sales_by_customer()
        elif choice == "4":
            print("Exiting Sales Reporting System.")
            break
        else:
            print("Invalid choice. Please try again.")


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

Output to generate total sales report



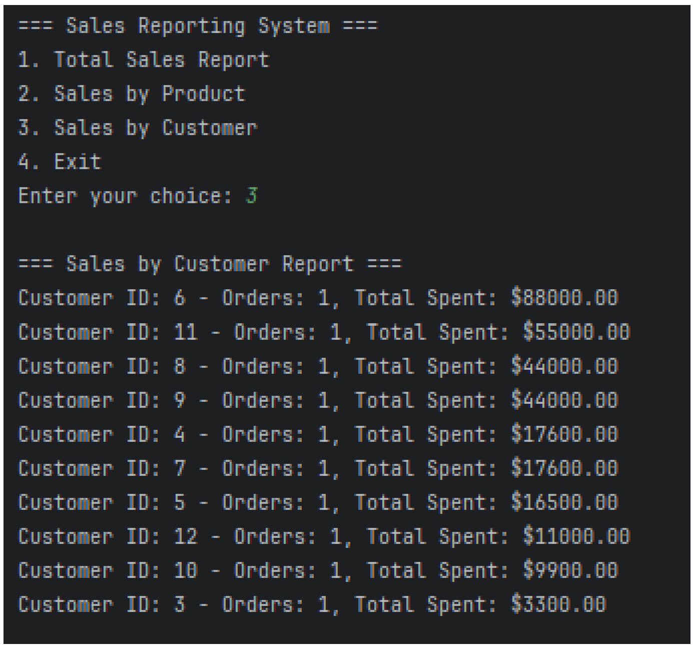
```
=== Sales Reporting System ===
1. Total Sales Report
2. Sales by Product
3. Sales by Customer
4. Exit
Enter your choice: 1

=== Total Sales Report ===
Total Sales: $306900.00
Total Orders: 10
```

Output to generate sales by product:



```
=== Sales Reporting System ===
1. Total Sales Report
2. Sales by Product
3. Sales by Customer
4. Exit
Enter your choice: 2

=== Sales by Product Report ===
laptops - Quantity Sold: 101, Total Revenue: $5555000.00
CPU - Quantity Sold: 2, Total Revenue: $88000.00
GPU - Quantity Sold: 1, Total Revenue: $44000.00
tablet - Quantity Sold: 2, Total Revenue: $44000.00
RAM - Quantity Sold: 2, Total Revenue: $17600.00
ROM - Quantity Sold: 2, Total Revenue: $17600.00
monitor - Quantity Sold: 3, Total Revenue: $16500.00
stylus - Quantity Sold: 3, Total Revenue: $9900.00
ethernet - Quantity Sold: 3, Total Revenue: $3300.00
```

Output to generate sales by customer:



```
=== Sales Reporting System ===
1. Total Sales Report
2. Sales by Product
3. Sales by Customer
4. Exit
Enter your choice: 3

=== Sales by Customer Report ===
Customer ID: 6 - Orders: 1, Total Spent: $88000.00
Customer ID: 11 - Orders: 1, Total Spent: $55000.00
Customer ID: 8 - Orders: 1, Total Spent: $44000.00
Customer ID: 9 - Orders: 1, Total Spent: $44000.00
Customer ID: 4 - Orders: 1, Total Spent: $17600.00
Customer ID: 7 - Orders: 1, Total Spent: $17600.00
Customer ID: 5 - Orders: 1, Total Spent: $16500.00
Customer ID: 12 - Orders: 1, Total Spent: $11000.00
Customer ID: 10 - Orders: 1, Total Spent: $9900.00
Customer ID: 3 - Orders: 1, Total Spent: $3300.00
```

7: Customer Account Updates

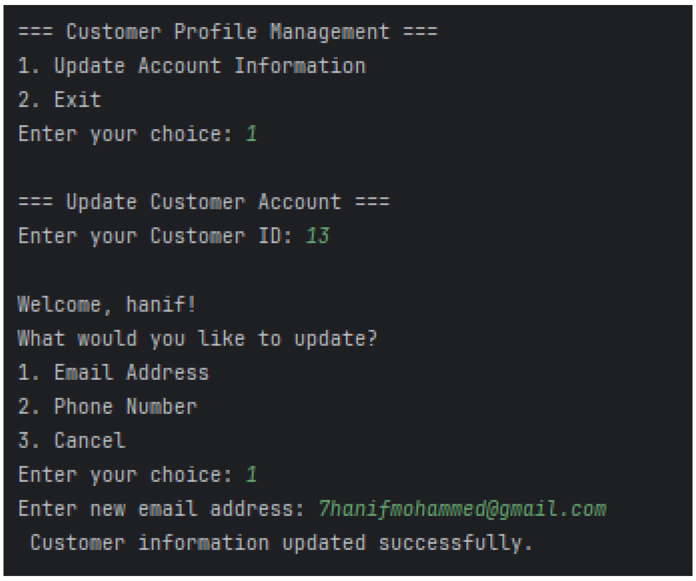
Description: Customers may need to update their account information, such as changing their email address or phone number.

Task: Implement a user profile management feature with database connectivity to allow customers to update their account details. Ensure data validation and integrity.

Code:
```
import pymysql
import re
from db_connection import connect_db


def is_valid_email(email):
   return re.match(r"[^@]+@[^@]+\.[^@]+", email)


def is_valid_phone(phone):
   return phone.isdigit() and len(phone) == 10


def update_customer_account():
   print("\n=== Update Customer Account ===")
   customer_id = input("Enter your Customer ID: ").strip()

   if not customer_id.isdigit():
      print("Error: Invalid Customer ID.")
      return

   conn = connect_db()
   if conn:
      try:
         with conn.cursor() as cursor:
            cursor.execute("SELECT * FROM customers WHERE CustomerID = %s",
(customer_id,))
            customer = cursor.fetchone()

            if not customer:
               print(" Error: Customer not found.")
               return

            print(f"\nWelcome, {customer[1]}!")
            print("What would you like to update?")
            print("1. Email Address")
            print("2. Phone Number")
            print("3. Cancel")

            choice = input("Enter your choice: ").strip()

            if choice == "1":
               new_email = input("Enter new email address: ").strip()
               if not is_valid_email(new_email):
                  print(" Error: Invalid email format.")
                  return
```

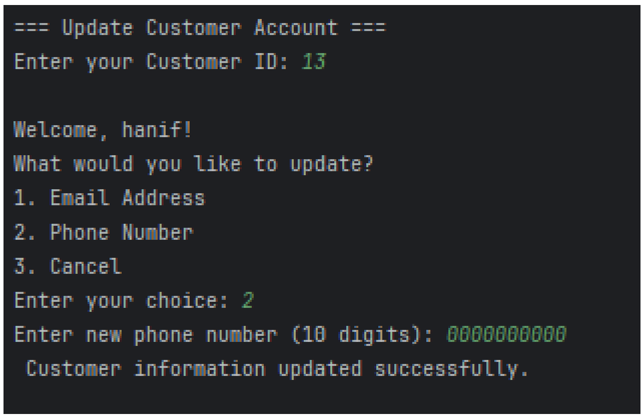
```
                cursor.execute("UPDATE customers SET email = %s WHERE CustomerID = 
%s", (new_email, customer_id))

            elif choice == "2":
                new_phone = input("Enter new phone number (10 digits): ").strip()
                if not is_valid_phone(new_phone):
                    print(" Error: Invalid phone number.")
                    return
                cursor.execute("UPDATE customers SET phone = %s WHERE CustomerID = 
%s",
                               (new_phone, customer_id))

            elif choice == "3":
                print("Cancelled.")
                return

            else:
                print(" Invalid choice. Please try again.")
                return

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

        except pymysql.MySQLError as e:
            print(f" Error: {e}")
        finally:
            conn.close()


def main():
    while True:
        print("\n=== Customer Profile Management ===")
        print("1. Update Account Information")
        print("2. Exit")

        choice = input("Enter your choice: ").strip()

        if choice == "1":
            update_customer_account()
        elif choice == "2":
            print("Exiting...")
            break
        else:
            print(" Invalid choice. Please try again.")


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

Output to update account information(email address):

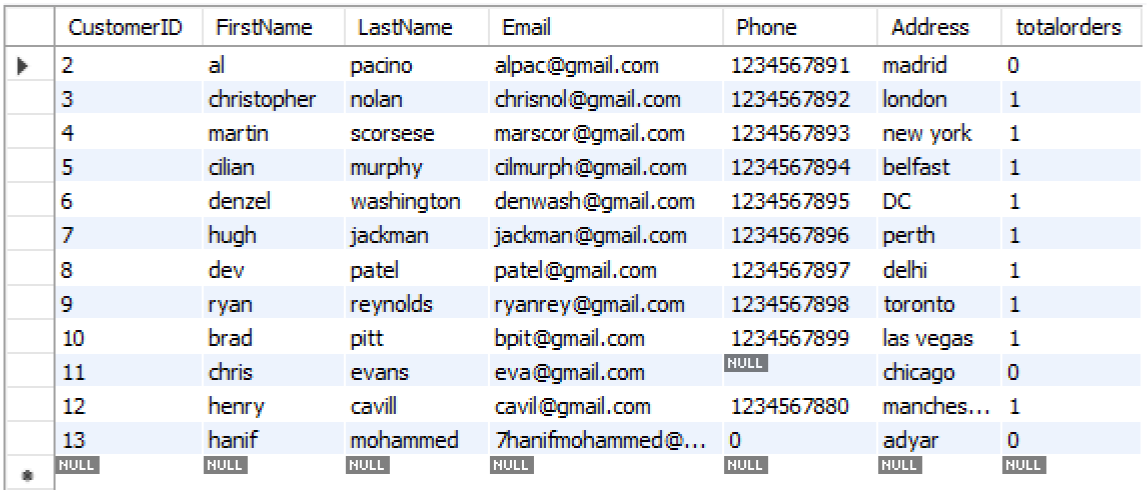


```
=== Customer Profile Management ===
1. Update Account Information
2. Exit
Enter your choice: 1

=== Update Customer Account ===
Enter your Customer ID: 13

Welcome, hanif!
What would you like to update?
1. Email Address
2. Phone Number
3. Cancel
Enter your choice: 1
Enter new email address: 7hanifmohammed@gmail.com
 Customer information updated successfully.
```

Output to update account information(phone number):



```
=== Update Customer Account ===
Enter your Customer ID: 13

Welcome, hanif!
What would you like to update?
1. Email Address
2. Phone Number
3. Cancel
Enter your choice: 2
Enter new phone number (10 digits): 0000000000
 Customer information updated successfully.
```



CustomerID	FirstName	LastName	Email	Phone	Address	totalorders
2	al	pacino	alpac@gmail.com	1234567891	madrid	0
3	christopher	nolan	chrisnol@gmail.com	1234567892	london	1
4	martin	scorsese	marscor@gmail.com	1234567893	new york	1
5	cilian	murphy	cilmurph@gmail.com	1234567894	belfast	1
6	denzel	washington	denwash@gmail.com	1234567895	DC	1
7	hugh	jackman	jackman@gmail.com	1234567896	perth	1
8	dev	patel	patel@gmail.com	1234567897	delhi	1
9	ryan	reynolds	ryanrey@gmail.com	1234567898	toronto	1
10	brad	pitt	bpit@gmail.com	1234567899	las vegas	1
11	chris	evans	eva@gmail.com	NULL	chicago	0
12	henry	cavill	cavil@gmail.com	1234567880	manches...	1
13	hanif	mohammed	7hanifmohammed@...	0	adyar	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL

8: Payment Processing

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

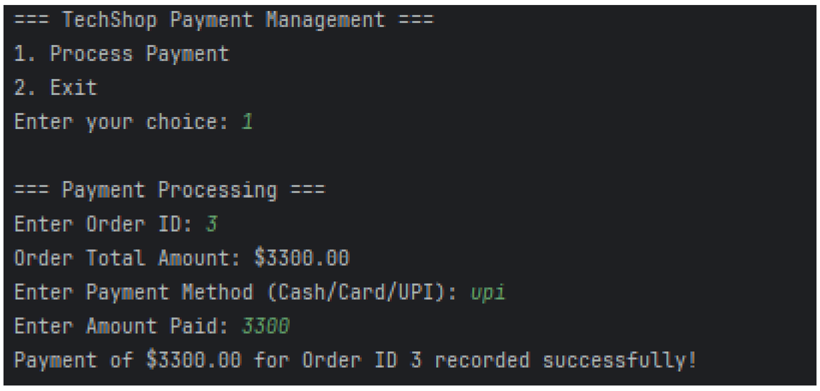
Task: Develop a payment processing system that interacts with the database to record payment transactions, validate payment information, and handle errors.

Code:

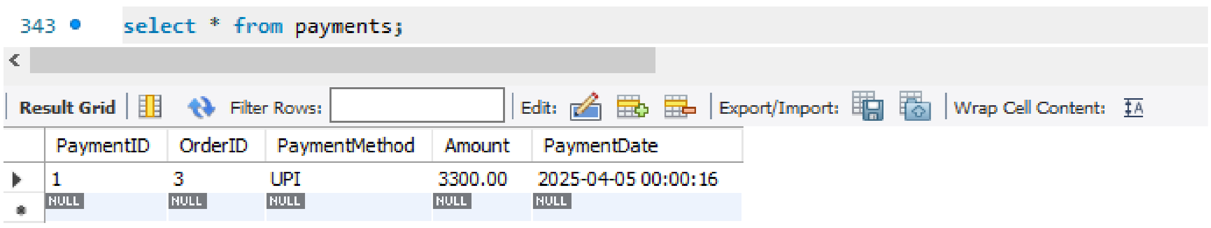
```
import pymysql
from db_connection import connect_db

def create_payments_table():
    """Creates the payments table if it does not exist."""
    conn = connect_db()
    if conn:
        try:
            with conn.cursor() as cursor:
                cursor.execute("""
                    CREATE TABLE IF NOT EXISTS payments (
                        PaymentID INT AUTO_INCREMENT PRIMARY KEY,
                        OrderID INT NOT NULL,
                        PaymentMethod ENUM('Cash', 'Card', 'UPI') NOT NULL,
                        Amount DECIMAL(10,2) NOT NULL,
                        PaymentDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
                        FOREIGN KEY (OrderID) REFERENCES orders(OrderID) ON DELETE
CASCADE
                    )
                """)
                conn.commit()
                print("Payments table is ready.")
        except pymysql.MySQLError as e:
            print(f"Database Error: {e}")
        finally:
            conn.close()

def process_payment():
    """Records a payment for a specific order."""
    print("\n=== Payment Processing ===")
    order_id = input("Enter Order ID: ").strip()

    if not order_id.isdigit():
        print("Error: Invalid Order ID.")
        return

    conn = connect_db()
    if conn:
```

```
    try:
        with conn.cursor() as cursor:
            cursor.execute("SELECT OrderID, TotalAmount FROM orders WHERE OrderID =
%s", (order_id,))
            order = cursor.fetchone()
            if not order:
                print("Error: Order not found.")
                return

            order_total = order[1]
            print(f"Order Total Amount: ${order_total:.2f}")

            # Get payment method and amount
            payment_method = input("Enter Payment Method (Cash/Card/UPI):
").strip().lower()
            if payment_method not in ['cash', 'card', 'upi']:
                print("Error: Invalid payment method.")
                return

            amount_paid = input("Enter Amount Paid: ").strip()
            if not amount_paid.replace('.', '', 1).isdigit():
                print("Error: Invalid amount.")
                return

            amount_paid = float(amount_paid)
            if amount_paid < order_total:
                print("Error: Paid amount is less than order total.")
                return

            # Insert payment record
            cursor.execute(
                "INSERT INTO payments (OrderID, PaymentMethod, Amount) VALUES (%s,
%s, %s)",
                (order_id, payment_method, amount_paid)
            )
            conn.commit()
            print(f"Payment of ${amount_paid:.2f} for Order ID {order_id} recorded
successfully!")

    except pymysql.MySQLError as e:
        print(f"Database Error: {e}")
    finally:
        conn.close()

def main():
    """Main function to trigger payment feature."""
    create_payments_table()
    while True:
    print("\n=== TechShop Payment Management ===")
    print("1. Process Payment")
    print("2. Exit")

    choice = input("Enter your choice: ").strip()

    if choice == "1":
        process_payment()
    elif choice == "2":
        print("Exiting Payment Management.")
        break
    else:
        print("Invalid choice. Please try again.")

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

Output to process payment:



```
=== TechShop Payment Management ===
1. Process Payment
2. Exit
Enter your choice: 1

=== Payment Processing ===
Enter Order ID: 3
Order Total Amount: $3300.00
Enter Payment Method (Cash/Card/UPI): upi
Enter Amount Paid: 3300
Payment of $3300.00 for Order ID 3 recorded successfully!
```



```
select * from payments;
```

PaymentID	OrderID	PaymentMethod	Amount	PaymentDate
1	3	UPI	3300.00	2025-04-05 00:00:16
NULL	NULL	NULL	NULL	NULL

9: Product Search and Recommendations

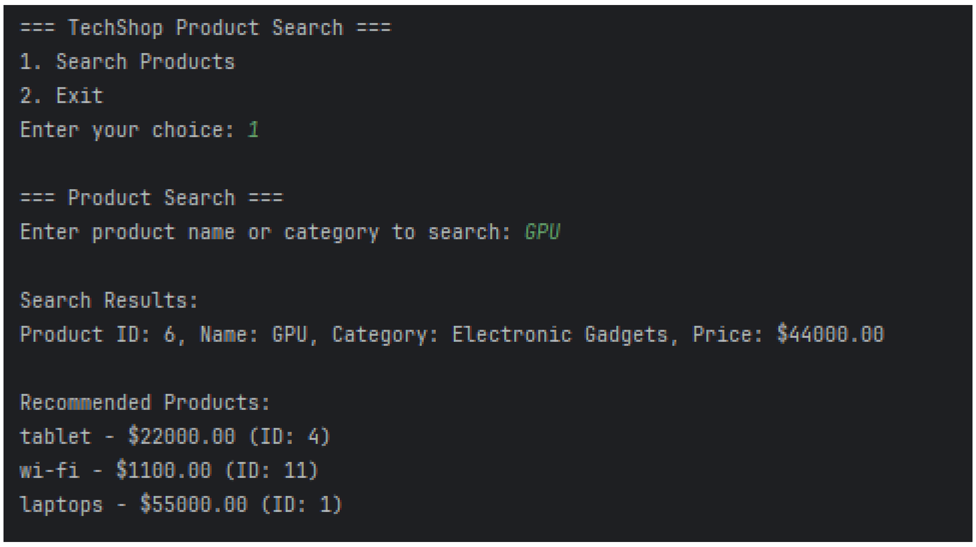
Description: Customers should be able to search for products based on various criteria (e.g., name, category) and receive product recommendations.

Task: Implement a product search and recommendation engine that uses database connectivity to retrieve relevant product information.

Code:

```
import pymysql
from db_connection import connect_db

def track_order_status():
    """Allows users to view the status of their orders."""
    print("\n=== Track Order Status ===")
    customer_id = input("Enter your Customer ID: ").strip()

    if not customer_id.isdigit():
        print(" Error: Invalid Customer ID.")
        return

    conn = connect_db()
    if conn:
        try:
            with conn.cursor() as cursor:
                cursor.execute("SELECT OrderID, OrderDate, OrderStatus, TotalAmount FROM
orders WHERE CustomerID = %s",
                               (customer_id,))
                orders = cursor.fetchall()

                if not orders:
                    print(" No orders found for this customer.")
                    return

                print("\nYour Order Status:")
                print("-" * 50)
                print(f"{'Order ID':<10}{'Order Date':<20}{'Status':<15}{'Total ($)'}")
                print("-" * 50)

                for order in orders:
                    print(f"{order[0]:<10}{order[1]:<20}{order[2]:<15}{order[3]:.2f}")

        except pymysql.MySQLError as e:
            print(f" Error: {e}")
        finally:
            conn.close()


def main():
    """Main function to track order status."""
    while True:
        print("\n=== Order Tracking System ===")
        print("1. Track Order Status")
        print("2. Exit")
        choice = input("Enter your choice: ").strip()

        if choice == "1":
            track_order_status()
        elif choice == "2":
            print("Exiting Order Tracking System.")
            break
        else:
            print("Invalid choice. Please try again.")

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

Output to search products with their recommendations:



```
=== TechShop Product Search ===
1. Search Products
2. Exit
Enter your choice: 1

=== Product Search ===
Enter product name or category to search: GPU

Search Results:
Product ID: 6, Name: GPU, Category: Electronic Gadgets, Price: $44000.00

Recommended Products:
tablet - $22000.00 (ID: 4)
wi-fi - $1100.00 (ID: 11)
laptops - $55000.00 (ID: 1)
```