Trigger to block admin when login fails for three times:

```
DELIMITER $$
CREATE TRIGGER check_failed_logins
AFTER INSERT ON login_attempts
FOR EACH ROW
BEGIN
   DECLARE failed_attempts INT;
   DECLARE last_attempt_time TIMESTAMP;

   -- Get the number of failed attempts for the admin in the last 24 hours
   SELECT COUNT(*), MAX(attempt_time) INTO failed_attempts, last_attempt_time
   FROM login_attempts
   WHERE admin_id = NEW.admin_id
   AND attempt_time >= DATE_SUB(NOW(), INTERVAL 1 DAY)
   AND success = FALSE;

   -- Check if the admin has exceeded the limit
   IF failed_attempts >= 3 THEN
      -- Update the admin_lockouts table to indicate blocked
      UPDATE admin_lockouts
      SET locked_out = TRUE,
         lockout_start = last_attempt_time,
         lockout_end = last_attempt_time + INTERVAL 24 HOUR
      WHERE admin_id = NEW.admin_id;
   END IF;
END$$
DELIMITER ;
```

Trigger to send notification to vendor when particular item goes low in stock

```
DELIMITER $$

CREATE TRIGGER low_quantity_notification
AFTER UPDATE ON product_inventory
FOR EACH ROW
BEGIN
   DECLARE product_count INT;
   DECLARE vendorr_id INT;

   -- Retrieve the current quantity of the updated product
   SET product_count = NEW.quantity;

   -- Check if the updated quantity is below 5
   IF product_count <= 5 THEN
      -- Retrieve the vendor associated with the product
      SELECT vendor_id INTO vendorr_id FROM product_inventory WHERE product_id
= NEW.product_id;

      -- Print notification for the vendor when they log in
      INSERT INTO notifications (vendor_id, message) VALUES (vendorr_id,
CONCAT('Product with ID ', NEW.product_id, ' is low in quantity. Please restock.'));
   END IF;
        IF product_count > 5 THEN
                -- Retrieve the vendor associated with the product
                SELECT vendor_id INTO vendorr_id FROM product_inventory WHERE
product_id = NEW.product_id;
                -- Delete notification when product is restocked
                DELETE FROM notifications WHERE vendor_id = vendorr_id;
   END IF;
END$$

DELIMITER ;
```

This trigger sends notification to vendors about their product whenever there's an update in product inventory and their product quantity is below 5.
This trigger also removes the notification for the vendor when they restock the item more than 5.

–Trigger to add admin data from block after failed attempts record when new admin is added

```
DELIMITER $$

CREATE TRIGGER remove_admin_from_lockout_when_deleted
AFTER DELETE ON Admins
FOR EACH ROW
BEGIN
   DELETE FROM admin_lockouts WHERE admin_id = OLD.admin_id;
END$$

DELIMITER ;
```

–Trigger to remove admin data from block after failed attempts record when a admin is deleted

```
DELIMITER $$

CREATE TRIGGER adding_new_admin_to_lockouts
AFTER INSERT ON Admins
FOR EACH ROW
BEGIN
   INSERT INTO admin_lockouts (admin_id, locked_out, lockout_start,
lockout_end)
   VALUES (NEW.admin_id, 0, NULL, NULL);
END$$

DELIMITER ;
```

New created tables:

```
CREATE TABLE notifications (
    notification_id INT AUTO_INCREMENT PRIMARY KEY,
    vendor_id INT,
    message TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE login_attempts (
    attempt_id INT AUTO_INCREMENT PRIMARY KEY,
    admin_id INT NOT NULL,
    attempt_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    success BOOLEAN NOT NULL
);

CREATE TABLE admin_lockouts (
    admin_id INT PRIMARY KEY,
    locked_out BOOLEAN DEFAULT 0,
    lockout_start TIMESTAMP,
    lockout_end TIMESTAMP
);
```

There should be data of existing admin in admin_lockouts, there is a trigger created above to look into this

This is the embedded sql queries in python

```
import mysql. connector
from datetime import datetime




def place_order(customer_id, product_id, quantity):
    conn = mysql.connector.connect(host='localhost',
username='root', password='mysql2023', database='drop_24')
    my_cursor = conn.cursor()

    my_cursor.execute("SELECT COUNT(*) FROM Customers WHERE
customer_id = %s", (customer_id,))
    customer_exists = my_cursor.fetchone()[0]

    if not customer_exists:
        print(f"Customer with ID {customer_id} does not exist.")
        conn.close()
        return

    my_cursor.execute("SELECT COUNT(*) FROM Products WHERE
product_id = %s", (product_id,))
    product_exists = my_cursor.fetchone()[0]

    if not product_exists:
        print(f"Product with ID {product_id} does not exist.")
        conn.close()
        return
```

```
    my_cursor.execute("SELECT quantity FROM Products WHERE
product_id = %s", (product_id,))
    available_quantity = my_cursor.fetchone()[0]

    if available_quantity >= quantity:
        my_cursor.execute("INSERT INTO Orders (customer_id,
order_date) VALUES (%s, %s)", (customer_id, datetime.now()))
        order_id = my_cursor.lastrowid

        my_cursor.execute("UPDATE Products SET quantity =
quantity - %s WHERE product_id = %s", (quantity, product_id))

        my_cursor.execute("INSERT INTO Order_details (order_id,
product_id, quantity) VALUES (%s, %s, %s)", (order_id,
product_id, quantity))

        my_cursor.execute("UPDATE product_inventory SET quantity
= quantity - %s WHERE product_id = %s", (quantity, product_id))

        conn.commit()
        print("\nOrder placed successfully!")
    else:
        print(f"Insufficient stock for product {product_id}.
Available quantity: {available_quantity}")

    conn.close()



def low_stock_products():
    conn=mysql. connector .connect(host='localhost',
username='root', password = 'mysql2023',database = 'drop_24')
```

```
    my_cursor = conn.cursor()

    my_cursor.execute("""
        SELECT product_name, quantity
        FROM products
        WHERE quantity <= 5
    """)

    low_stock_items = my_cursor.fetchall()

    if low_stock_items:
        print("\nFollowing products have low stock:")
        for item in low_stock_items:
            print(f"- {item[0]} (Quantity: {item[1]})")
    else:
        print("\nAll products have sufficient stock.")

    conn.close()




def show_tables():
    conn=mysql. connector .connect(host='localhost',
username='root', password = 'mysql2023',database = 'drop_24')
    my_cursor = conn.cursor()
    my_cursor.execute("""Show tables""")
    conn.close()
```

```
def get_admin_id(admin_name):
    try:
        conn = mysql.connector.connect(host='localhost',
username='root', password='mysql2023', database='drop_24')
        my_cursor = conn.cursor()

        my_cursor.execute("SELECT admin_id FROM Admins WHERE
admin_name = %s", (admin_name,))
        result = my_cursor.fetchone()

        if result:
            admin_id = result[0]
            return admin_id
        else:
            print("Admin not found.")
            return None

    except mysql.connector.Error as err:
        print("Error:", err)

    finally:
        if 'conn' in locals():
            my_cursor.close()
            conn.close()
```

```
def admin_loginn(username, password):
   try:
       conn = mysql.connector.connect(host='localhost',
username='root', password='mysql2023', database='drop_24')
       my_cursor = conn.cursor()

       # verify if admin is blocked
       admin_id = get_admin_id(username)
       if admin_id:
           my_cursor.execute("SELECT locked_out, lockout_end
FROM admin_lockouts WHERE admin_id = %s", (admin_id,))
           lockout_status = my_cursor.fetchone()

           if lockout_status and lockout_status[0] == 1:
               if lockout_status[1] > datetime.now():
                   print("\nAdmin is currently blocked. Please
try again later.")
                   return False
               elif lockout_status[1] <= datetime.now():
                       my_cursor.execute("UPDATE admin_lockouts
SET locked_out = 0 WHERE admin_id = %s", (admin_id,))
                       conn.commit()
                       print("\Block period is expired. Admin is
now unblocked.")
```

```
            my_cursor.execute("SELECT * FROM Admins WHERE
admin_name = %s", (username,))
            user = my_cursor.fetchone()
            if user:
                if user[2] == password:  # Check if password
matches
                    print("\nLogin successful!")
                    return True
                else:
                    my_cursor.execute("INSERT INTO login_attempts
(admin_id, attempt_time, success) VALUES (%s, %s, %s)",
                                      (admin_id, datetime.now(),
False))
                    conn.commit()
                    print("\nInvalid password.")
                    return False
            else:
                print("\nInvalid username.")
                return False
        else:
            print("\nAdmin not found.")
            return False

    except mysql.connector.Error as err:
        print("Error:", err)

    finally:
        if 'conn' in locals():
            my_cursor.close()
            conn.close()
```

```
def customer_login(email, password):
   try:
       conn=mysql. connector .connect(host='localhost',
username='root', password = 'mysql2023',database = 'drop_24')
       my_cursor = conn.cursor()

       my_cursor.execute("SELECT * FROM customers WHERE email =
%s AND password = %s", (email, password))
       user = my_cursor.fetchone()

       if user:
           print("Login successful!")
           return True
       else:
           print("Invalid username or password.")
           return False

   except mysql.connector.Error as err:
       print("Error:", err)

   finally:
       if 'conn' in locals():
           my_cursor.close()
           conn.close()
```

```
def vendor_login(email, password):
    try:
        conn=mysql. connector .connect(host='localhost',
username='root', password = 'mysql2023',database = 'drop_24')
        my_cursor = conn.cursor()

        my_cursor.execute("SELECT * FROM vendors WHERE
vendor_email = %s AND password = %s", (email, password))
        user = my_cursor.fetchone()

        if user:
            print("Login successful!")
            print()
            return True
        else:
            print("Invalid username or password.")
            return False

    except mysql.connector.Error as err:
        print("Error:", err)

    finally:
        if 'conn' in locals():
            my_cursor.close()
            conn.close()
```

```
def get_customer_name(email):

   conn=mysql. connector .connect(host='localhost',
username='root', password = 'mysql2023',database = 'drop_24')
   my_cursor = conn.cursor()

   query = "SELECT customer_name FROM customers WHERE email =
%s"

   my_cursor.execute(query, (email,))

   result = my_cursor.fetchone()

   my_cursor.close()
   conn.close()

   if result:
       return result[0]
   else:
       return None
```

```python
def get_customer_id(email):
    conn=mysql. connector .connect(host='localhost',
username='root', password = 'mysql2023',database = 'drop_24')
    my_cursor = conn.cursor()

    query = "SELECT customer_id FROM customers WHERE email = %s"


    my_cursor.execute(query, (email,))

    result = my_cursor.fetchone()

    my_cursor.close()
    conn.close()

    if result:
        return result[0]
    else:
        return None









def get_vendor_id(email):
```

```
    conn=mysql. connector .connect(host='localhost',
username='root', password = 'mysql2023',database = 'drop_24')
    my_cursor = conn.cursor()

    query = "SELECT vendor_id FROM vendors WHERE vendor_email =
%s"



    my_cursor.execute(query, (email,))

    result = my_cursor.fetchone()

    my_cursor.close()
    conn.close()

    if result:
        return result[0]
    else:
        return None









def my_orders(customer_id):
    conn=mysql. connector .connect(host='localhost',
username='root', password = 'mysql2023',database = 'drop_24')
    my_cursor = conn.cursor()
```

```
    my_cursor.execute("""
        SELECT order_id, order_date
        FROM orders
        WHERE customer_id = %s
        ORDER BY order_date
    """, (customer_id,))
    orders = my_cursor.fetchall()


    for order in orders:
        order_id, order_date = order
        print()
        print("..............................")
        print("Order Date:", order_date)

        my_cursor.execute("""
            SELECT od.product_id, p.product_name, od.quantity,
p.price
            FROM order_details od
            JOIN products p ON od.product_id = p.product_id
            WHERE od.order_id = %s
        """, (order_id,))
        order_details = my_cursor.fetchall()

        total_order_value = 0


        for order_detail in order_details:
            product_id, product_name, quantity, price =
order_detail
            total_order_value += quantity * price
```

```
            print(f"{product_name}  x{quantity}          ->
{price}")

        print("Total order value        ->", total_order_value)
        print("..............................")
        print()
        print()


    my_cursor.close()
    conn.close()









def list_products():
    try:
        conn = mysql.connector.connect(host='localhost',
username='root', password='mysql2023', database='drop_24')
        my_cursor = conn.cursor(dictionary=True)

        my_cursor.execute("SELECT category_id, category_name FROM
categories")
        categories = my_cursor.fetchall()

        for category in categories:
            category_id = category['category_id']
```

```
            category_name = category['category_name']
            print("\n" + category_name + ":")

            my_cursor.execute("""
                SELECT DISTINCT p.product_id, p.product_name,
p.price
                FROM products p
                JOIN product_inventory pi ON p.product_id =
pi.product_id
                WHERE p.category_id = %s
            """, (category_id,))
            products = my_cursor.fetchall()

            for product in products:
                product_id = product['product_id']
                product_name = product['product_name']
                price = product['price']
                print(f"Product ID: {product_id}, Product Name:
{product_name}, Price: {price}")


        my_cursor.close()
        conn.close()

    except mysql.connector.Error as err:
        print("Error:", err)
```

```
def list_categories():
   try:
       conn = mysql.connector.connect(host='localhost',
username='root', password='mysql2023', database='drop_24')
       my_cursor = conn.cursor(dictionary=True)

       my_cursor.execute("SELECT category_id, category_name FROM
categories")
       categories = my_cursor.fetchall()

       for category in categories:
           category_id = category['category_id']
           category_name = category['category_name']
           print("\n" + "ID: " + str(category_id) + "    " +
category_name )

       print()



       my_cursor.close()
       conn.close()

   except mysql.connector.Error as err:
       print("Error:", err)
```

```
def get_total_sales():
   conn=mysql. connector .connect(host='localhost',
username='root', password = 'mysql2023',database = 'drop_24')
   my_cursor = conn.cursor()

   my_cursor.execute("""
       SELECT SUM(p.price * od.quantity)
       FROM orders o
       JOIN order_details od ON o.order_id = od.order_id
       JOIN products p ON od.product_id = p.product_id
   """)
   total_sales = my_cursor.fetchone()[0] or 0

   my_cursor.close()
   conn.close()

   return total_sales




def get_sales_by_vendors():
   conn=mysql. connector .connect(host='localhost',
username='root', password = 'mysql2023',database = 'drop_24')
   my_cursor = conn.cursor(dictionary=True)
```

```
    my_cursor.execute("""
        SELECT v.vendor_name, SUM(p.price * od.quantity) AS
vendor_sales
        FROM orders o
        JOIN order_details od ON o.order_id = od.order_id
        JOIN products p ON od.product_id = p.product_id
        JOIN product_inventory pi ON p.product_id = pi.product_id
        JOIN vendors v ON pi.vendor_id = v.vendor_id
        GROUP BY v.vendor_name
    """)
    sales_by_vendors = my_cursor.fetchall()

    my_cursor.close()
    conn.close()

    return sales_by_vendors







def get_sales_by_categories():
    conn=mysql. connector .connect(host='localhost',
username='root', password = 'mysql2023',database = 'drop_24')
    my_cursor = conn.cursor(dictionary=True)

    my_cursor.execute("""
        SELECT c.category_name, SUM(p.price * od.quantity) AS
category_sales
```

```
        FROM orders o
        JOIN order_details od ON o.order_id = od.order_id
        JOIN products p ON od.product_id = p.product_id
        JOIN categories c ON p.category_id = c.category_id
        GROUP BY c.category_name
    """)
    sales_by_categories = my_cursor.fetchall()

    my_cursor.close()
    conn.close()

    return sales_by_categories




def remove_customer(email):
    try:
        conn=mysql. connector .connect(host='localhost',
username='root', password = 'mysql2023',database = 'drop_24')
        my_cursor = conn.cursor()

        my_cursor.execute("DELETE FROM customers WHERE email =
%s", (email,))
        conn.commit()

    except mysql.connector.Error as err:
        print("Error:", err)

    finally:
        my_cursor.close()
        conn.close()
```

```
def remove_vendor(email):
    try:
        conn=mysql. connector .connect(host='localhost',
username='root', password = 'mysql2023',database = 'drop_24')
        my_cursor = conn.cursor()

        my_cursor.execute("DELETE FROM vendors WHERE vendor_email
= %s", (email,))
        conn.commit()

    except mysql.connector.Error as err:
        print("Error:", err)

    finally:
        my_cursor.close()
        conn.close()




def add_vendor():
    while True:
        name = input("Enter vendor name: ")
        email = input("Enter vendor email: ")
        if not validate_email(email):
            print("Invalid email format. Please try again.")
            continue
```

```
        phone_number = input("Enter vendor phone number: ")
        password = input("Enter vendor password: ")

        try:
            conn=mysql. connector .connect(host='localhost',
username='root', password = 'mysql2023',database = 'drop_24')
            my_cursor = conn.cursor()

            my_cursor.execute("""
                INSERT INTO vendors (vendor_name, vendor_email,
vendor_phone, password)
                VALUES (%s, %s, %s, %s)
            """, (name, email, phone_number, password))
            conn.commit()

            print("Vendor added successfully.")

        except mysql.connector.Error as err:
            print("Error:", err)

        finally:
            my_cursor.close()
            conn.close()

        break




def add_customer():
    while True:
        name = input("Enter customer name: ")
```

```
        email = input("Enter customer email: ")
        if not validate_email(email):
            print("Invalid email format. Please try again.")
            continue
        phone_number = input("Enter customer phone number: ")
        password = input("Enter customer password: ")
        address = input("Enter Your Address")
        dob = input("Enter customer date of birth (YYYY-MM-DD):
")
        if not validate_dob(dob):
            print("Invalid date of birth format. Please try
again.")
            continue
        age = calculate_age(dob)

        try:
            conn=mysql. connector .connect(host='localhost',
username='root', password = 'mysql2023',database = 'drop_24')
            my_cursor = conn.cursor()

            my_cursor.execute("""
                INSERT INTO Customers (customer_name,
customer_address, email, password, DOB, age, phone_number)
                VALUES (%s, %s, %s, %s, %s, %s, %s)
            """, (name, address, email, password, dob, age,
phone_number))
            conn.commit()

            print("Customer added successfully.")

        except mysql.connector.Error as err:
            print("Error:", err)
```

```
        finally:
            my_cursor.close()
            conn.close()

        break









def add_product(product_name, price, category_id, quantity,
vendor_id):
    try:
        conn = mysql.connector.connect(host='localhost',
username='root', password='mysql2023', database='drop_24')
        my_cursor = conn.cursor()

        my_cursor.execute("INSERT INTO products (product_name,
price, category_id, quantity) VALUES (%s, %s, %s, %s)",
                          (product_name, price, category_id,
quantity))
        product_id = my_cursor.lastrowid

        my_cursor.execute("INSERT INTO product_inventory
(product_id, category_id, quantity, vendor_id) VALUES (%s, %s,
%s, %s)",
                          (product_id, category_id, quantity,
vendor_id))
```

```
        conn.commit()
        print("Product added successfully!")

    except mysql.connector.Error as err:
        print("Error:", err)

    finally:
        if 'conn' in locals():
            my_cursor.close()
            conn.close()









def list_all_product_of_vendor(vendor_id):
    try:
        conn = mysql.connector.connect(host='localhost',
username='root', password='mysql2023', database='drop_24')
        my_cursor = conn.cursor()

        my_cursor.execute("""
            SELECT p.product_id, p.product_name, p.price,
pi.quantity
            FROM products p
            JOIN product_inventory pi ON p.product_id =
pi.product_id
            WHERE pi.vendor_id = %s
        """, (vendor_id,))
```

```
        products = my_cursor.fetchall()

        # Print the products
        if products:
            print("Vendor Products:")
            for product in products:
                print(f"Product ID: {product[0]},    Product
Name: {product[1]},  Quantity: {product[3]}")
        else:
            print("No products found for the vendor.")

        print()
        print("Enter X to exit ")

    except mysql.connector.Error as err:
        print("Error:", err)

    finally:
        if 'conn' in locals():
            my_cursor.close()
            conn.close()



def remove_product_by_vendor(vendor_id, product_id, amount):
    try:
        conn = mysql.connector.connect(host='localhost',
username='root', password='mysql2023', database='drop_24')
        my_cursor = conn.cursor()
        my_cursor.execute("SELECT 1 FROM product_inventory WHERE
vendor_id = %s AND product_id = %s", (vendor_id, product_id))
        exists = my_cursor.fetchone()
```

```
        if exists:
            my_cursor.execute("UPDATE products SET quantity =
quantity - %s WHERE product_id = %s", (amount, product_id))

            my_cursor.execute("UPDATE product_inventory SET
quantity = quantity - %s WHERE vendor_id = %s AND product_id =
%s", (amount, vendor_id, product_id))

            conn.commit()
            print("Product removed successfully!")
        else:
            print("Product not found for the given vendor.")

    except mysql.connector.Error as err:
        print("Error:", err)

    finally:
        if 'conn' in locals():
            my_cursor.close()
            conn.close()




def restock_product(vendor_id, product_id, amount):
    try:
        conn = mysql.connector.connect(host='localhost',
username='root', password='mysql2023', database='drop_24')
        my_cursor = conn.cursor()
```

```
        my_cursor.execute("SELECT 1 FROM product_inventory WHERE
vendor_id = %s AND product_id = %s", (vendor_id, product_id))
        exists = my_cursor.fetchone()

        if exists:
            my_cursor.execute("UPDATE products SET quantity =
quantity + %s WHERE product_id = %s", (amount, product_id))

            my_cursor.execute("UPDATE product_inventory SET
quantity = quantity + %s WHERE vendor_id = %s AND product_id =
%s", (amount, vendor_id, product_id))

            conn.commit()
            print("Product restocked successfully!")
        else:
            print("Product not found for the given vendor.")

        print()
        print("Enter X to exit ")

    except mysql.connector.Error as err:
        print("Error:", err)

    finally:
        if 'conn' in locals():
            my_cursor.close()
            conn.close()



def vendor_notification(vendor_id):
    try:
```

```
        conn = mysql.connector.connect(host='localhost',
username='root', password='mysql2023', database='drop_24')
        my_cursor = conn.cursor()

        my_cursor.execute("SELECT message, created_at FROM
notifications WHERE vendor_id = %s", (vendor_id,))
        messages = my_cursor.fetchall()
        conn.commit()

        if messages:
            print("Notifications: ")
            for product in messages:
                print(f"Message: {product[0]}     Time:
{product[1]}")
        else:
            print("No New Messages for you")
        print()

    except mysql.connector.Error as err:
        print("Error:", err)

    finally:
        if 'conn' in locals():
            my_cursor.close()
            conn.close()
```

```
def validate_email(email):
    return "@" in email and "." in email

def validate_dob(dob):
    try:
        datetime.strptime(dob, "%Y-%m-%d")
        return True
    except ValueError:
        return False

def calculate_age(dob):
    dob_date = datetime.strptime(dob, "%Y-%m-%d")
    today = datetime.today()
    age = today.year - dob_date.year - ((today.month, today.day)
< (dob_date.month, dob_date.day))
    return age









def main():
    while True:
        print("\nRetail Store Management System")
        print("1. Admin Login")
        print("2. Customer Login")
        print("3. Vendor Login")
```

```
        print("4. Exit")


        login_choice = input("Enter your choice: ")

        if login_choice == '1':
            username = input("Enter your username: ")
            password = input("Enter your password: ")
            if admin_loginn(username, password):
                while True:
                    print("\nRetail Store Management System")
                    print("1. Add / Remove customer")
                    print("2. Add / Remove vendor")
                    print("3. Check Low Stock Products")
                    print("4. Total Sales")
                    print("5. Total Sales by vendor")
                    print("6. Total Sales by category")
                    print("7. Logout")

                    admin_choice = input("Enter your choice: ")

                    if admin_choice == '1':
                        print("1. Add customer")
                        print("2. Remove customer")

                        adre_choice = input("Enter Your choice:
")

                        if adre_choice == '1':
                            add_customer()
                        elif adre_choice == '2':
```

```
                            cus_email = input("Enter customer
email to remove: ")
                            print()
                            cus_name =
get_customer_name(cus_email)
                            print("are you sure you want to
remove ", cus_name, " from database?")
                            print("y/n")
                            if input()=='y':
                                print(cus_name, " removed
succesfully from database")
                                remove_customer(cus_email)
                            print()

                    elif admin_choice == '2':
                        print("1. Add Vendor")
                        print("2. Remove Vendor")

                        adre_choice = input("Enter Your choice:
")


                        if adre_choice == '1':
                            add_vendor()
                        elif adre_choice == '2':
                            vend_email = input("Enter customer
email to remove: ")
                            print()
                            vend_name =
get_vendor_name(vend_email)
                            print("are you sure you want to
remove ", vend_name, " from database?")
                            print("y/n")
```

```
                            if input()=='y':
                                print(vend_name, " removed
succesfully from database")
                                remove_vendor(vend_email)
                            print()

                    elif admin_choice == '3':
                        low_stock_products()

                    elif admin_choice == '4':
                        total_sales = get_total_sales()
                        print()
                        print("Total Sales ->", total_sales)
                        print()

                    elif admin_choice == '5':
                        sales_by_vendors = get_sales_by_vendors()
                        print()
                        print("\nSales by Vendors:")
                        for record in sales_by_vendors:
                            print(record['vendor_name'], "->",
record['vendor_sales'])
                        print()

                    elif admin_choice == '6':
                        sales_by_categories =
get_sales_by_categories()
                        print()
                        print("\nSales by Categories:")
                        for record in sales_by_categories:
                            print(record['category_name'], "->",
record['category_sales'])
```

```
                    print()

                elif admin_choice == '7':
                    print("Logging out...")
                    print(".")
                    print(".")
                    print(".")
                    print(".")
                    print(".")
                    print(".")
                    print(".")
                    print("Logged out")
                    break


    elif login_choice == '2':
        email = input("Enter your Email: ")
        password = input("Enter your password: ")
        if customer_login(email, password):
            username = get_customer_name(email)
            customer_id = get_customer_id(email)
            while True:
                print("\nWelcome ", username, "!")
                print("1. Place Order")
                print("2. My Orders")
                print("3. Logout")

                customer_choice = input("Enter Your Choice:
")

                if customer_choice == '1':
                    list_products()
```

```
                        product_id = int(input("Enter product ID:
"))
                        quantity = int(input("Enter quantity: "))
                        place_order(customer_id, product_id,
quantity)
                    elif customer_choice == '2':
                        my_orders(customer_id)
                    elif customer_choice == '3':
                        print("Logging out...")
                        print(".")
                        print(".")
                        print(".")
                        print(".")
                        print(".")
                        print(".")
                        print(".")
                        print("Logged out")
                        break
                    else:
                        print("Invalid choice. Please try
again.\n")

        elif login_choice == '3':
            email = input("Enter Your Email: ")
            password = input("Enter Your Password: ")
            if vendor_login(email, password):
                ven_id = get_vendor_id(email)
                vendor_notification(ven_id)
                while True:
                    print("1. Add Product")
                    print("2. Remove Product")
                    print("3. restock")
```

```
                    print("4. Log Out")

                    vend_choice = input("Enter Your Choice: ")

                    if vend_choice == '1':
                        name = input("Enter Product's name: ")
                        price = input("Enter Price of product: ")
                        list_categories()
                        category = input("Sele which category it
falls into: ")
                        quantityy = input("Enter the quantity: ")
                        add_product(name, price, category,
quantityy, ven_id)
                    elif vend_choice == '2':
                        list_all_product_of_vendor(ven_id)
                        print()
                        remove_pro = input("Enter ID of product
to remove: ")
                        if (remove_pro == "X") or (remove_pro ==
"x"):
                            print()
                        else:
                            hello = input("Enter the quantity you
want to delete: ")
                            remove_product_by_vendor(ven_id,
remove_pro, hello)
                    elif vend_choice == '3':
                        list_all_product_of_vendor(ven_id)
                        print()
                        restock = input("Enter ID of product to
restock: ")
                        if (restock == "X") or (restock == "x"):
```

```
                            print()
                        else:
                            amount = input("Enter the number of
items you want to add: ")
                            restock_product(ven_id, restock,
amount)
                    elif vend_choice == '4':
                        print("Logging out...")
                        print(".")
                        print(".")
                        print(".")
                        print(".")
                        print(".")
                        print(".")
                        print(".")
                        print("Logged out")
                        break


        elif login_choice == '4':
            print("Exiting...")
            return
        else:
            print("Invalid choice. Please try again.\n")

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