

# DBMS Mini Project Review

Name: Sumithra Suresh (PES2UG23CS625)

Tanish Hegde (PES2UG23CS641)

Semester: 5<sup>th</sup> Semester

Section: J

1. Title:

Cake Database Management System

2. Description:

This project aims to design and implement a database management system for a Cake Store chain.

The system maintains data related to customers, orders, cakes, outlets, payments, and employees.

Our objective is to create a fully functional CRUD-based DBMS system supported by:

- A well-structured ER diagram
- A normalized relational schema
- SQL tables with appropriate constraints
- Triggers, procedures, and functions
- A working front-end interface for interacting with the database

The application enables outlet staff/admin to manage cake inventory, handle customer orders, track payments, monitor sales, and generate analytical insights using MySQL and a Streamlit-based UI.

3. User Requirement Specification:

Purpose of The Project:

The purpose of this project is to create an easy-to-use and well-organized database system that supports the everyday operations of a cake store. It helps the store keep track of customers, manage cake inventory, record orders and payments, and handle information from different outlets—all in one place. With a proper database

structure, the store can avoid manual errors, save time, and run its operations more efficiently.

#### Scope of the Project:

The scope of this project includes designing the database, creating all the required tables with proper relationships, and adding triggers, procedures, and functions that support the cake store's day-to-day workflow. It manages different types of data—customers, cakes, orders, payments, outlets, and even multiple phone numbers and addresses—so the system can reflect real store operations accurately.

On the front-end side, the project provides a Streamlit app that lets users easily view tables, add new records, place or cancel orders, restock cakes, and check simple analytics. The project focuses on the essential features needed for smooth store management while keeping the overall design simple and suitable for an academic mini-project.

#### Detailed Description:

To support the growing operations of a multi-outlet cake store, this project focuses on designing a structured and reliable Cake Store Database Management System. At the center of the system is the Cake Catalogue, where each cake is uniquely identified and described through its name, category, price, description, and available quantity. This ensures that every outlet can accurately view and manage the cakes it offers.

Each Outlet—identified by an Outlet ID—stores details such as its name, address, and phone numbers. Outlets receive cake catalogues and handle customer orders, making them a key part of the store's workflow. Outlets are staffed by Employees, each identified with an Employee ID and associated with attributes like their name, role, and phone number. An employee works at exactly one outlet, helping maintain clear accountability and easy staff management.

The system also includes Customers, who are uniquely identified using a Customer ID. Each customer's name, email, phone number, and address are stored, allowing the outlet to deliver the order correctly and maintain accurate customer records. Customers interact with the system by placing Orders, which record details such as order date, total amount, and status (e.g., Completed, Pending, or Cancelled). Every order connects three major entities — the customer who placed it, the cake being purchased, and the outlet fulfilling it.

Payments are captured separately through the Payment entity. Each payment includes the amount, method of payment, and payment date, and is associated with a single order. This ensures proper financial tracking and avoids mismatches between completed orders and their payments.

4. List of Software/Tools/Programming languages used:

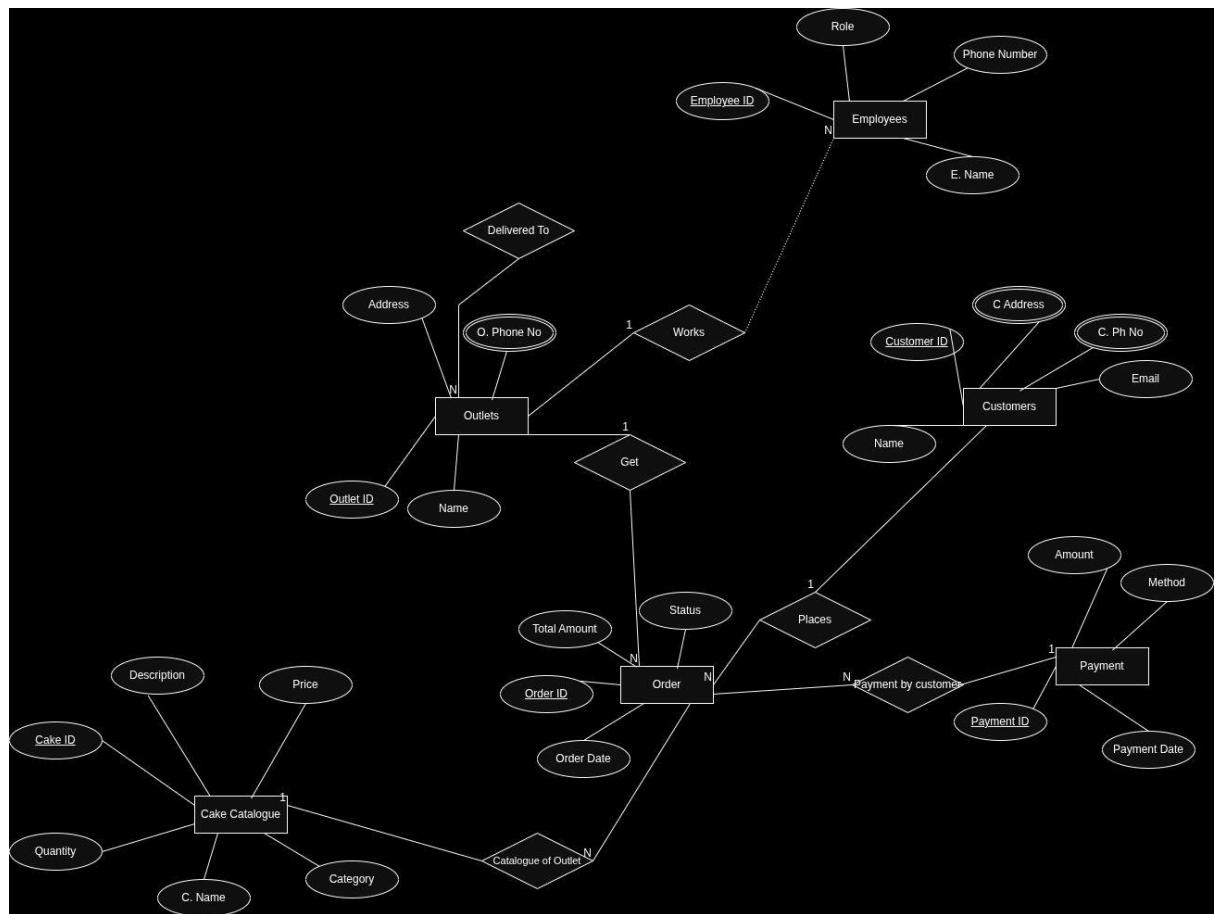
Streamlist – Python

MySQL

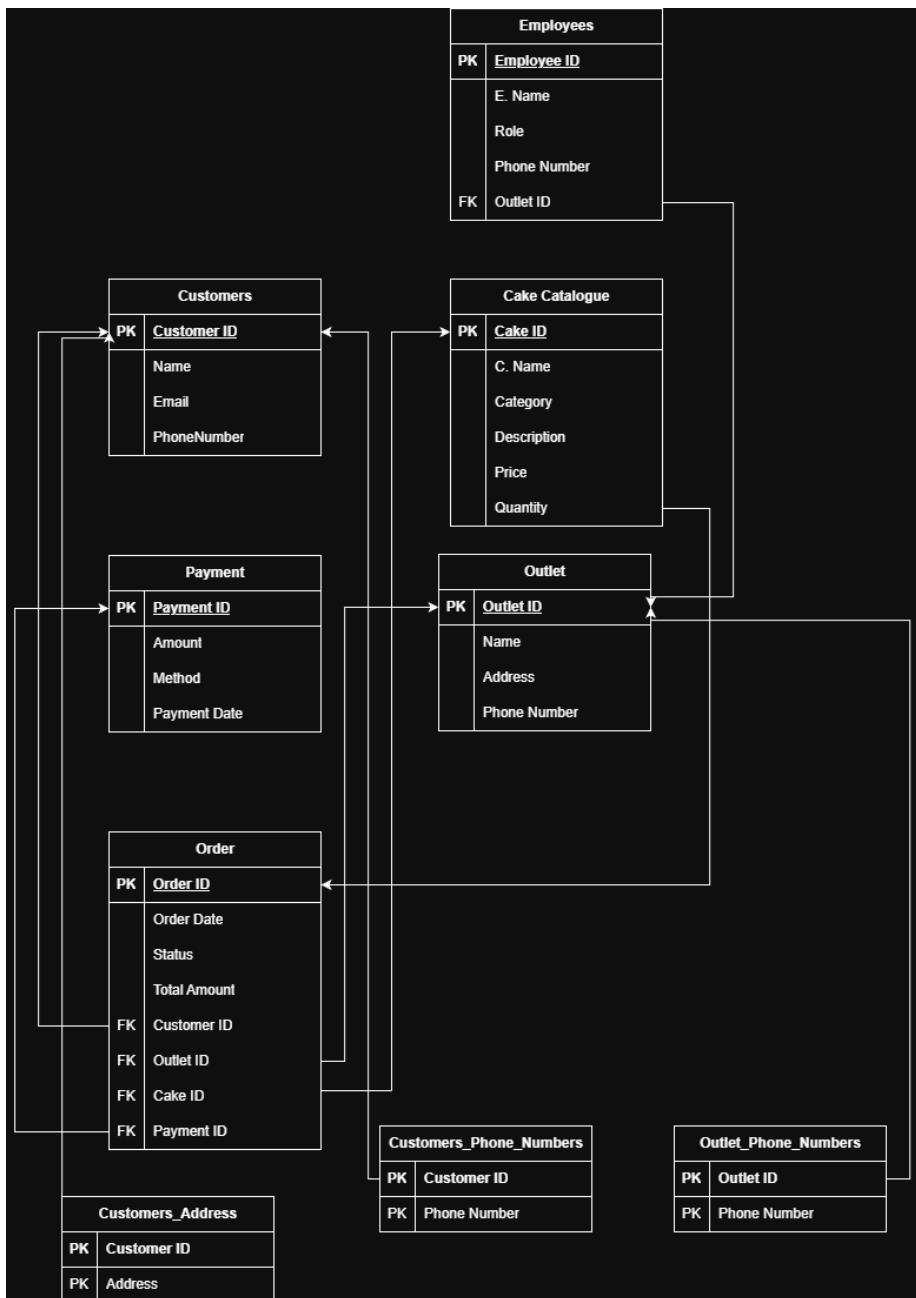
Pandas

Draw.io

5. ER Diagram



6. Relational Schema



## 7. DDL Commands

```

1 • CREATE DATABASE IF NOT EXISTS Cake_Store;
2 • USE Cake_Store;
3
4 -- =====
5 -- Tables
6 -- =====
7
8 • CREATE TABLE Payment (
9     Payment_ID INT PRIMARY KEY,
10    Amount DECIMAL(10,2),
11    Method VARCHAR(50),
12    Payment_Date DATE
13 );
14
15 • CREATE TABLE Outlet (
16     Outlet_ID INT PRIMARY KEY,
17    Name VARCHAR(100),
18    Address VARCHAR(200),
19    Phone_Number VARCHAR(15)
20 );
21
22 • CREATE TABLE Employees (
23     Employee_ID INT PRIMARY KEY,
24    E_Name VARCHAR(100),
25    RoleOfPerson VARCHAR(50),
26    Phone_Number VARCHAR(15),
27    Outlet_ID INT,
28    Production_ID INT,
29    FOREIGN KEY (Outlet_ID) REFERENCES Outlet(Outlet_ID)
30        ON DELETE CASCADE
31        ON UPDATE CASCADE
32 );
33
34 • CREATE TABLE Customers (
35     Customer_ID INT PRIMARY KEY,
36    NameOfPerson VARCHAR(100),
37    Email VARCHAR(100),
38    PhoneNumber VARCHAR(15)
39 );
40
41 • CREATE TABLE Cake_Catalogue (
42     Cake_ID INT PRIMARY KEY AUTO_INCREMENT,
43    C_Name VARCHAR(100),
44    C_Category VARCHAR(50),
45    C_Description TEXT,
46    Price DECIMAL(8,2),
47    Quantity INT
48 );
49
50 • CREATE TABLE Order_Table (
51     Order_ID INT PRIMARY KEY AUTO_INCREMENT,
52    Order_Date DATE,
53    StatusOrder VARCHAR(20),
54    Total_Amount DECIMAL(10,2),
55    Customer_ID INT,
56    Outlet_ID INT,
57    Cake_ID INT,
58    Payment_ID INT,
59    FOREIGN KEY (Customer_ID) REFERENCES Customers(Customer_ID)
60        ON DELETE CASCADE
61        ON UPDATE CASCADE,
62    FOREIGN KEY (Outlet_ID) REFERENCES Outlet(Outlet_ID)
63        ON DELETE CASCADE
64        ON UPDATE CASCADE,
65    FOREIGN KEY (Cake_ID) REFERENCES Cake_Catalogue(Cake_ID)
66        ON DELETE CASCADE
67        ON UPDATE CASCADE,
68    FOREIGN KEY (Payment_ID) REFERENCES Payment(Payment_ID)
69        ON DELETE CASCADE
70        ON UPDATE CASCADE
71 );
72
73 • CREATE TABLE Customers_Address (
74     Customer_ID INT,
75    Address VARCHAR(200),
76    PRIMARY KEY (Customer_ID, Address),
77    FOREIGN KEY (Customer_ID) REFERENCES Customers(Customer_ID)
78        ON DELETE CASCADE
79        ON UPDATE CASCADE
80 );
81
82 • CREATE TABLE Customers_Phone_Numbers (
83     Customer_ID INT,
84    Phone_Number VARCHAR(15),
85    PRIMARY KEY (Customer_ID, Phone_Number),
86    FOREIGN KEY (Customer_ID) REFERENCES Customers(Customer_ID)
87        ON DELETE CASCADE
88        ON UPDATE CASCADE
89 );
90
91 • CREATE TABLE Outlet_Phone_Numbers (
92     Outlet_ID INT,
93    Phone_Number VARCHAR(15),
94    PRIMARY KEY (Outlet_ID, Phone_Number),
95    FOREIGN KEY (Outlet_ID) REFERENCES Outlet(Outlet_ID)
96        ON DELETE CASCADE
97        ON UPDATE CASCADE
98 );
99

```

## 8. CRUD operation Screenshots

## View Tables

The screenshot shows the 'View Tables' section of the dashboard. A dropdown menu on the left says 'Choose Action' with 'View Tables' selected. The main area has a title 'Cake Store Database Dashboard' with a cake icon. Below it is a sub-section titled 'View Tables' with a bar chart icon. A dropdown menu labeled 'Select a table' shows 'Cake\_Catalogue'. The main content is a table with the following data:

Cake_ID	C_Name	C_Category	C_Description	Price	Quantity
0	Chocolate Truffle	Chocolate	Rich dark chocolate cake	450	11
1	Vanilla Delight	Classic	Soft vanilla sponge cake	350	20
2	Red Velvet	Premium	Red velvet cream cheese cake	550	9
3	Black Forest	Classic	Cherry and chocolate layered cake	500	15
4	Ian	blackforest	test	1000	17
5	Blue Beauty	Holiday Special	Blueberry Cheesecake	899	90

## Insert Data

The screenshot shows the 'Insert Data' section of the dashboard. A dropdown menu on the left says 'Choose Action' with 'Insert Data' selected. The main area has a title 'Cake Store Database Dashboard' with a cake icon. Below it is a sub-section titled 'Insert Data' with a plus sign icon. A dropdown menu labeled 'Select Table' shows 'Cake\_Catalogue'. The main content is a form for inserting data into the 'Cake\_Catalogue' table:

**Inserting into Cake\_Catalogue**

C_Name	Nutcracker
C_Category	Christmas Special
C_Description	Praline, Almonds, Caramel, Vanilla
Price	675
Quantity	50

Buttons: Insert Record, Press Enter to apply, -, +.

The screenshot shows the 'View Tables' section of the dashboard after an insertion. A dropdown menu on the left says 'Choose Action' with 'View Tables' selected. The main area has a title 'Cake Store Database Dashboard' with a cake icon. Below it is a sub-section titled 'View Tables' with a bar chart icon. A dropdown menu labeled 'Select a table' shows 'Cake\_Catalogue'. The main content is a table with the following data, including the new row:

Cake_ID	C_Name	C_Category	C_Description	Price	Quantity
0	Chocolate Truffle	Chocolate	Rich dark chocolate cake	450	11
1	Vanilla Delight	Classic	Soft vanilla sponge cake	350	20
2	Red Velvet	Premium	Red velvet cream cheese cake	550	9
3	Black Forest	Classic	Cherry and chocolate layered cake	500	15
4	Ian	blackforest	test	1000	17
5	Blue Beauty	Holiday Special	Blueberry Cheesecake	899	90
6	Nutcracker	Christmas Special	Praline, Almonds, Caramel, Vanilla	675	50

## Update Order Status

The screenshot shows the 'Update Order Status' section of the dashboard. A dropdown menu on the left says 'Choose Action' with 'Update Order Status' selected. The main area has a title 'Cake Store Database Dashboard' with a cake icon. Below it is a sub-section titled 'Update Order Status' with a gear icon. A dropdown labeled 'Select Order' contains the number '5'. Below it, 'Current Status: Cancelled' is shown. A dropdown labeled 'Select New Status' contains 'Pending'. A button labeled 'Update Status' is visible. At the bottom, a green success message box contains the text '✓ Status updated to Pending!'. The top right corner has a 'Deploy' button.

The screenshot shows the 'View Tables' section of the dashboard. A dropdown menu on the left says 'Choose Action' with 'View Tables' selected. The main area has a title 'Cake Store Database Dashboard' with a cake icon. Below it is a sub-section titled 'View Tables' with a bar chart icon. A dropdown labeled 'Select a table' has 'Order\_Table' selected. Below it is a table with the following data:

	Order_ID	Order_Date	StatusOrder	Total_Amount	Customer_ID	Outlet_ID	Cake_ID	Payment_ID
0	1	2025-02-10	Cancelled	450	1	1	1	1
1	2	2025-02-12	Cancelled	550	2	2	3	2
2	3	2025-02-15	Completed	350	3	1	2	3
3	4	2025-02-16	Cancelled	500	4	3	4	4
4	5	2025-11-14	Pending	1000	1	1	5	2
5	6	2025-11-21	Pending	675	3	2	7	3

The top right corner has a 'Deploy' button.

## Analytics Dashboard

The screenshot shows the 'Analytics Dashboard' section of the dashboard. A dropdown menu on the left says 'Choose Action' with 'Analytics Dashboard' selected. The main area has a title 'Cake Store Database Dashboard' with a cake icon. Below it is a sub-section titled 'Dashboard' with a chart icon. It displays statistics: 'Total Orders' (6), 'Completed' (1), and 'Cancelled' (3). Below this is a section titled 'Best Selling Cakes' with a chart showing a single data point for 'Vanilla Delight'. The top right corner has a 'Deploy' button.

## Custom Query

The screenshot shows a dark-themed dashboard titled "Cake Store Database Dashboard". On the left, a sidebar has a dropdown menu with "Run Custom Query" selected. The main area is titled "Custom Query" with a magnifying glass icon. Below it, a text input field contains the SQL query: "SELECT \* FROM Cake\_Catalogue WHERE Quantity > 50;". A "Run" button is located below the input field. The results are displayed in a table:

Cake_ID	C_Name	C_Category	C_Description	Price	Quantity
0	Red Velvet	Premium	Red velvet cream cheese cake	550	59
1	Blue Beauty	Holiday Special	Blueberry Cheesecake	899	90

9. List of functionalities/features of the application and its associated screenshots using front end

## Functions

The screenshot shows a dark-themed dashboard titled "Cake Store Database Dashboard". On the left, a sidebar has a dropdown menu with "Run Functions" selected. The main area is titled "Run MySQL Functions" with a brain icon. It displays two functions:  
1. **GetCakeSales(cake\_id)**:  
A dropdown menu labeled "Select Cake ID" shows the value "2". A "Get Sales" button is present.  
2. **CheckStock(cake\_id)**:  
A dropdown menu labeled "Check Stock for Cake" shows the value "5". A "Check Stock" button is present.  
The status message "Stock Status: OK" is displayed in a green bar at the bottom.

If Stock is less than 5

```
mysql> UPDATE Cake_Catalogue SET Quantity = 4 WHERE Cake_ID = 5;
Query OK, 1 row affected (0.12 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

**CheckStock(cake\_id)**

Check Stock for Cake

**Check Stock**

Stock Status: LOW STOCK

If Stock is 0 or negative

```
mysql> UPDATE Cake_Catalogue SET Quantity = 0 WHERE Cake_ID = 5;
Query OK, 1 row affected (0.14 sec)
Rows matched: 1    Changed: 1    Warnings: 0
```

**CheckStock(cake\_id)**

Check Stock for Cake

**Check Stock**

Stock Status: OUT OF STOCK

## Procedures

### Place Order

Choose Action

View Tables

Cake Store Database Dashboard

View Tables

Select a table

Order\_Table

Order_ID	Order_Date	StatusOrder	Total_Amount	Customer_ID	Outlet_ID	Cake_ID	Payment_ID
0	2025-02-10	Cancelled	450	1	1	1	1
1	2025-02-12	Cancelled	550	2	2	3	2
2	2025-02-15	Completed	350	3	1	2	3
3	2025-02-16	Cancelled	500	4	3	4	4
4	2025-11-14	Cancelled	1000	1	1	5	2

Choose Action

Place Order (Procedure)

## Cake Store Database Dashboard

### Place Order

Customer ID  
3

Outlet ID  
2

Cake ID  
7

Payment ID  
3

**Place Order**

✓ Order placed successfully!

Deploy ⚙️

Choose Action

View Tables

## Cake Store Database Dashboard

### View Tables

Select a table

Order\_Table

	Order_ID	Order_Date	StatusOrder	Total_Amount	Customer_ID	Outlet_ID	Cake_ID	Payment_ID
0	1	2025-02-10	Cancelled	450	1	1	1	1
1	2	2025-02-12	Cancelled	550	2	2	3	2
2	3	2025-02-15	Completed	350	3	1	2	3
3	4	2025-02-16	Cancelled	500	4	3	4	4
4	5	2025-11-14	Cancelled	1000	1	1	5	2
5	6	2025-11-21	Pending	675	3	2	7	3

Deploy ⚙️

## Restock Cake

The screenshot shows a dark-themed dashboard for a cake store. At the top left, there's a dropdown menu labeled "Choose Action" with "View Tables" selected. The main title is "Cake Store Database Dashboard" with a small cake icon. Below it is a section titled "View Tables" with a bar chart icon. A sub-section titled "Select a table" has "Cake\_Catalogue" chosen. The central part of the screen is a table with the following data:

Cake_ID	C_Name	C_Category	C_Description	Price	Quantity
0	Chocolate Truffle	Chocolate	Rich dark chocolate cake	450	11
1	Vanilla Delight	Classic	Soft vanilla sponge cake	350	20
2	Red Velvet	Premium	Red velvet cream cheese cake	550	9
3	Black Forest	Classic	Cherry and chocolate layered cake	500	15
4	Tan	blackforest	test	1000	17
5	Blue Beauty	Holiday Special	Blueberry Cheesecake	899	90
6	Nutcracker	Christmas Special	Praline, Almonds, Caramel, Vanilla	675	49

The screenshot shows a dark-themed dashboard for a cake store. At the top left, there's a dropdown menu labeled "Choose Action" with "Restock Cake (Proc...)" selected. The main title is "Cake Store Database Dashboard" with a small cake icon. Below it is a section titled "Restock Cake" with a cube icon. A sub-section titled "Select Cake" has "3" chosen. Below that is a "Add Quantity" input field with "50" and a plus/minus button. A "Restock" button with a clipboard icon is present. A green success message at the bottom says "✓ Stock updated!".

The screenshot shows a dark-themed dashboard for a cake store. At the top left, there's a dropdown menu labeled "Choose Action" with "View Tables" selected. The main title is "Cake Store Database Dashboard" with a small cake icon. Below it is a section titled "View Tables" with a bar chart icon. A sub-section titled "Select a table" has "Cake\_Catalogue" chosen. The central part of the screen is a table with the following data:

Cake_ID	C_Name	C_Category	C_Description	Price	Quantity
0	Chocolate Truffle	Chocolate	Rich dark chocolate cake	450	11
1	Vanilla Delight	Classic	Soft vanilla sponge cake	350	20
2	Red Velvet	Premium	Red velvet cream cheese cake	550	59
3	Black Forest	Classic	Cherry and chocolate layered cake	500	15
4	Tan	blackforest	test	1000	17
5	Blue Beauty	Holiday Special	Blueberry Cheesecake	899	90
6	Nutcracker	Christmas Special	Praline, Almonds, Caramel, Vanilla	675	49

## Triggers

Trigger when negative stock

```
mysql> UPDATE Cake_Catalogue SET Quantity = -1 WHERE Cake_ID = 5;
ERROR 1644 (45000): Cake stock cannot be negative!
```

Stock reduce on order

The screenshot shows a dark-themed dashboard titled "Cake Store Database Dashboard". On the left, a sidebar has a "Choose Action" dropdown set to "View Tables" and a "Place Order (Procedure)" option. The main area is titled "View Tables" and shows a table titled "Cake\_Catalogue". The table has columns: Cake\_ID, C\_Name, C\_Category, C\_Description, Price, and Quantity. The data is as follows:

Cake_ID	C_Name	C_Category	C_Description	Price	Quantity
0	Chocolate Truffle	Chocolate	Rich dark chocolate cake	450	11
1	Vanilla Delight	Classic	Soft vanilla sponge cake	350	20
2	Red Velvet	Premium	Red velvet cream cheese cake	550	59
3	Black Forest	Classic	Cherry and chocolate layered cake	500	15
4	tan	blackforest	test	1000	0
5	Blue Beauty	Holiday Special	Blueberry Cheesecake	899	90
6	Nutcracker	Christmas Special	Praline, Almonds, Caramel, Vanilla	675	49

The screenshot shows a dark-themed dashboard titled "Place Order" under the "Place Order (Procedure)" action. It has fields for Customer ID (1), Outlet ID (2), Cake ID (2), and Payment ID (2). A "Place Order" button is at the bottom. A green success message "✓ Order placed successfully!" is displayed below the button.

Choose Action

View Tables

## Cake Store Database Dashboard

View Tables

Select a table

Cake\_Catalogue

Cake_ID	C_Name	C_Category	C_Description	Price	Quantity
0	Chocolate Truffle	Chocolate	Rich dark chocolate cake	450	11
1	Vanilla Delight	Classic	Soft vanilla sponge cake	350	19
2	Red Velvet	Premium	Red velvet cream cheese cake	550	59
3	Black Forest	Classic	Cherry and chocolate layered cake	500	15
4	tan	blackforest	test	1000	0
5	Blue Beauty	Holiday Special	Blueberry Cheesecake	699	90
6	Nutcracker	Christmas Special	Praline, Almonds, Caramel, Vanilla	675	49

Deploy

### Stock increase on order cancellation

Choose Action

View Tables

## Cake Store Database Dashboard

View Tables

Select a table

Cake\_Catalogue

Cake_ID	C_Name	C_Category	C_Description	Price	Quantity
0	Chocolate Truffle	Chocolate	Rich dark chocolate cake	450	11
1	Vanilla Delight	Classic	Soft vanilla sponge cake	350	19
2	Red Velvet	Premium	Red velvet cream cheese cake	550	59
3	Black Forest	Classic	Cherry and chocolate layered cake	500	15
4	tan	blackforest	test	1000	0
5	Blue Beauty	Holiday Special	Blueberry Cheesecake	699	90
6	Nutcracker	Christmas Special	Praline, Almonds, Caramel, Vanilla	675	49

Choose Action

View Tables

## Cake Store Database Dashboard

### View Tables

Select a table

Order\_Table

	Order_ID	Order_Date	StatusOrder	Total_Amount	Customer_ID	Outlet_ID	Cake_ID	Payment_ID
0	1	2025-02-10	Cancelled	450	1	1	1	1
1	2	2025-02-12	Cancelled	550	2	2	3	2
2	3	2025-02-15	Completed	350	3	1	2	3
3	4	2025-02-16	Cancelled	500	4	3	4	4
4	5	2025-02-14	Pending	1000	1	1	5	2
5	6	2025-11-21	Pending	675	3	2	7	3
6	7	2025-11-21	Pending	350	1	2	2	2

Choose Action

Update Order Status

## Cake Store Database Dashboard

### Update Order Status

Select Order

7

Current Status: Pending

Select New Status

Cancelled

Update Status

✓ Status updated to Cancelled!

Choose Action

View Tables

## Cake Store Database Dashboard

### View Tables

Select a table

Cake\_Catalogue

Cake_ID	C_Name	C_Category	C_Description	Price	Quantity
0	1 Chocolate Truffle	Chocolate	Rich dark chocolate cake	450	11
1	2 Vanilla Delight	Classic	Soft vanilla sponge cake	350	20
2	3 Red Velvet	Premium	Red velvet cream cheese cake	550	59
3	4 Black Forest	Classic	Cherry and chocolate layered cake	500	15
4	5 tan	blackforest	test	1000	0
5	6 Blue Beauty	Holiday Special	Blueberry Cheesecake	899	90
6	7 Nutcracker	Christmas Special	Praline, Almonds, Caramel, Vanilla	675	49

## 10. Triggers, Procedures/Functions, Nested query, Join, Aggregate queries

```
167 -- =====
168 -- TRIGGERS
169 -- =====
170
171 -- 1. Prevent Negative Stock
172 DELIMITER //
173
174 CREATE TRIGGER trg_prevent_negative_stock
175 BEFORE UPDATE ON Cake_Catalogue
176 FOR EACH ROW
177 BEGIN
178     IF NEW.Quantity < 0 THEN
179         SIGNAL SQLSTATE '45000'
180         SET MESSAGE_TEXT = 'Cake stock cannot be negative!';
181     END IF;
182 END;
183 //
184
185 DELIMITER ;
186
187
188 -- 2. Auto Reduce Cake Stock on Order Insert
189 DELIMITER //
190
191 CREATE TRIGGER trg_reduce_cake_stock
192 AFTER INSERT ON Order_Table
193 FOR EACH ROW
194 BEGIN
195     UPDATE Cake_Catalogue
196     SET Quantity = Quantity - 1
197     WHERE Cake_ID = NEW.Cake_ID;
198 END;
199 //
200
201 DELIMITER ;
```

```
206
207 CREATE TRIGGER trg_calculate_order_total
208 BEFORE INSERT ON Order_Table
209 FOR EACH ROW
210 BEGIN
211     IF NEW.Total_Amount IS NULL OR NEW.Total_Amount = 0 THEN
212         SET NEW.Total_Amount = (
213             SELECT Price FROM Cake_Catalogue WHERE Cake_ID = NEW.Cake_ID
214         );
215     END IF;
216 END;
217 //
218
219 DELIMITER ;
220
221
222 -- 4. Validate Payment Exists
223 DELIMITER //
224
225 CREATE TRIGGER trg_validate_payment
226 BEFORE INSERT ON Order_Table
227 FOR EACH ROW
228 BEGIN
229     IF (SELECT COUNT(*) FROM Payment WHERE Payment_ID = NEW.Payment_ID) = 0 THEN
230         SIGNAL SQLSTATE '45000'
231         SET MESSAGE_TEXT = 'Invalid Payment_ID! Payment record does not exist.';
232     END IF;
233 END;
234 //
235
236 [DELIMITER ;
237
238 -- 5. Auto Increase Cake Stock on Order Cancellation
239 DELIMITER //
240
241 CREATE TRIGGER trg_increase_cake_stock_on_cancel
242 AFTER UPDATE ON Order_Table
243 FOR EACH ROW
244 BEGIN
245     -- Only trigger when status changes from NOT Cancelled → Cancelled
246     IF OLD.StatusOrder <> 'Cancelled' AND NEW.StatusOrder = 'Cancelled' THEN
```

```
246      IF OLD.StatusOrder <> 'Cancelled' AND NEW.StatusOrder = 'Cancelled' THEN
247          UPDATE Cake_Catalogue
248              SET Quantity = Quantity + 1
249              WHERE Cake_ID = NEW.Cake_ID;
250      END IF;
251 END;
252 //
253
254 DELIMITER ;
255
256
257 -- =====
258 -- PROCEDURES
259 -- =====
260
261 DELIMITER //
262
263 -- 1. Procedure: Place an Order
264 CREATE PROCEDURE Place_Order (
265     IN p_customer INT,
266     IN p_outlet INT,
267     IN p_cake INT,
268     IN p_payment INT
269 )
270 BEGIN
271     INSERT INTO Order_Table (
272         Order_Date,
273         StatusOrder,
274         Customer_ID,
275         Outlet_ID,
276         Cake_ID,
277         Payment_ID
278     )
279     VALUES (
280         CURDATE(),
281         'Pending',
282         p_customer,
283         p_outlet,
284         p_cake,
285         p_payment
286     );

```

```

290
291 -- 2. Procedure: Restock Cakes
292 CREATE PROCEDURE Restock_Cake (
293     IN p_cake_id INT,
294     IN p_add_quantity INT
295 )
296 BEGIN
297     UPDATE Cake_Catalogue
298     SET Quantity = Quantity + p_add_quantity
299     WHERE Cake_ID = p_cake_id;
300 END;
301 //
302
303 DELIMITER ;
304
305 -- =====
306 -- FUNCTIONS
307 -- =====
308
309 DELIMITER //
310
311 -- 1. Function: Get Total Sales of a Cake
312 CREATE FUNCTION GetCakeSales(cake INT)
313 RETURNS DECIMAL(10,2)
314 DETERMINISTIC
315 BEGIN
316     RETURN (
317         SELECT SUM(Total_Amount)
318         FROM Order_Table
319         WHERE Cake_ID = cake AND StatusOrder = 'Completed'
320     );
321 END;
322 //
323
324
325 -- 2. Function: Check Stock Health
326 CREATE FUNCTION CheckStock(cake INT)
327 RETURNS VARCHAR(20)
328 DETERMINISTIC
329 BEGIN
330     DECLARE qty INT;
331
332     -- 2. Function: Check Stock Health
333     CREATE FUNCTION CheckStock(cake INT)
334     RETURNS VARCHAR(20)
335     DETERMINISTIC
336     BEGIN
337         DECLARE qty INT;
338
339         SELECT Quantity INTO qty
340         FROM Cake_Catalogue
341         WHERE Cake_ID = cake;
342
343         IF qty <= 0 THEN
344             RETURN 'OUT OF STOCK';
345         ELSEIF qty < 5 THEN
346             RETURN 'LOW STOCK';
347         ELSE
348             RETURN 'OK';
349         END IF;
350     END;
351 //
352
353     DELIMITER ;

```

## 11. Code snippets for invoking the Procedures/Functions/Trigger

```
# -----
# UTILITIES
# -----
def fetch_df(query):
    conn = get_connection()
    df = pd.read_sql(query, conn)
    conn.close()
    return df

def get_list(table, col):
    df = fetch_df(f"SELECT {col} FROM {table}")
    return df[col].tolist()

def execute_sql(query, params=None):
    conn = get_connection()
    cursor = conn.cursor()
    cursor.execute(query, params if params else [])
    conn.commit()
    conn.close()

def get_value(table, select_col, where_col, where_val):
    conn = get_connection()
    cursor = conn.cursor()
    query = f"SELECT {select_col} FROM {table} WHERE {where_col} = %s"
    cursor.execute(query, (where_val,))
    result = cursor.fetchone()
    conn.close()
    return result[0] if result else None
```

```
# -----
# VIEW TABLES
# -----
if menu == "View Tables":
    st.header("📊 View Tables")

    tables = fetch_df("SHOW TABLES")
    table = st.selectbox("Select a table", tables.iloc[:, 0])

    if table:
        df = fetch_df(f"SELECT * FROM {table}")
        st.dataframe(df, use_container_width=True)
```

```
# INSERT DATA (generic)
# -----
elif menu == "Insert Data":
    st.header("✚ Insert Data")

    tables = fetch_df("SHOW TABLES")
    table = st.selectbox("Select Table", tables.iloc[:, 0])

    if table:
        conn = get_connection()
        cursor = conn.cursor()
        cursor.execute(f"DESCRIBE {table}")
        cols = cursor.fetchall()
        conn.close()

        auto_inc = ["Order_ID", "Cake_ID"]

        form = {}
        st.subheader(f"Inserting into {table}")

        for col in cols:
            name, dtype = col[0], col[1]

            if name in auto_inc:
                continue

            if "int" in dtype:
                form[name] = st.number_input(name, step=1)
            elif "date" in dtype:
                form[name] = st.date_input(name)
            else:
                form[name] = st.text_input(name)

        if st.button("Insert Record"):
            col_names = ", ".join(form.keys())
            placeholders = ", ".join(["%s"] * len(form))
            values = list(form.values())
            query = f"INSERT INTO {table} ({col_names}) VALUES ({placeholders})"

            try:
                execute_sql(query, values)
```

```
col_names = ", ".join(form.keys())
placeholders = ", ".join(["%s"] * len(form))
values = list(form.values())
query = f"INSERT INTO {table} ({col_names}) VALUES ({placeholders})"

try:
    execute_sql(query, values)
    st.success("✓ Record inserted!")
except Exception as e:
    st.error(f"✗ {e}")

# -----
# PLACE ORDER - CALL PROCEDURE
# -----
elif menu == "Place Order (Procedure)":
    st.header("💻 Place Order")

    customer = st.selectbox("Customer ID", get_list("Customers", "Customer_ID"))
    outlet = st.selectbox("Outlet ID", get_list("Outlet", "Outlet_ID"))
    cake = st.selectbox("Cake ID", get_list("Cake Catalogue", "Cake_ID"))
    payment = st.selectbox("Payment ID", get_list("Payment", "Payment_ID"))

    if st.button("Place Order"):
        try:
            execute_sql("CALL Place_Order(%s,%s,%s,%s)", (customer, outlet, cake, payment))
        except Exception as e:
            st.error(f"✗ {e}")
    else:
```

```
# RESTOCK CAKE – CALL PROCEDURE
# -----
elif menu == "Restock Cake (Procedure)":
    st.header("📦 Restock Cake")

    cake_id = st.selectbox("Select Cake", get_list("Cake_Catalogue", "Cake_ID"))
    qty = st.number_input("Add Quantity", step=1)

    if st.button("Restock"):
        execute_sql("CALL Restock_Cake(%s, %s)", (cake_id, qty))
        st.success("✓ Stock updated!")

# -----
# RUN FUNCTIONS
# -----
elif menu == "Run Functions":
    st.header("🌐 Run MySQL Functions")

    st.subheader("GetCakeSales(cake_id)")
    cake = st.selectbox("Select Cake ID", get_list("Cake_Catalogue", "Cake_ID"))
    if st.button("Get Sales"):
        df = fetch_df(f"SELECT GetCakeSales({{cake}}) AS TotalSales")
        st.info(f"Total Sales for Cake {cake}: ₹{df['TotalSales'][0]}")

    st.subheader("CheckStock(cake_id)")
    cake2 = st.selectbox("Check Stock for Cake", get_list("Cake_Catalogue", "Cake_ID"))
    if st.button("Check Stock"):
        df = fetch_df(f"SELECT CheckStock({{cake2}}) AS StockStatus")
        st.warning(f"Stock Status: {df['StockStatus'][0]}")
```

```
# -----
# UPDATE ORDER STATUS
# -----
elif menu == "Update Order Status":
    st.header("📝 Update Order Status")

    # Get existing order IDs
    orders = get_list("Order_Table", "Order_ID")
    order_id = st.selectbox("Select Order", orders)

    # Get current status to show in UI (optional)
    current_status = get_value("Order_Table", "StatusOrder", "Order_ID", order_id)
    st.write(f"**Current Status:** {current_status}")

    # Choose new status
    new_status = st.selectbox(
        "Select New Status",
        ["Pending", "Completed", "Cancelled"],
        index=["Pending", "Completed", "Cancelled"].index(current_status)
        if current_status in ["Pending", "Completed", "Cancelled"]
        else 0
    )

    # Update DB
    if st.button("Update Status"):
        execute_sql(
            "UPDATE Order_Table SET StatusOrder=%s WHERE Order_ID=%s",
            (new_status, order_id)
        )
        st.success(f"✓ Status updated to **{new_status}**!")

# -----
# ANALYTICS DASHBOARD
# -----
elif menu == "Analytics Dashboard":
    st.header("📈 Dashboard")

    col1, col2, col3 = st.columns(3)

    col1.metric("Total Orders", fetch_df("SELECT COUNT(*) AS c FROM Order_Table")["c"][0])
```

```
# -----
# ANALYTICS DASHBOARD
# -----
elif menu == "Analytics Dashboard":
    st.header("📈 Dashboard")

    col1, col2, col3 = st.columns(3)

    col1.metric("Total Orders", fetch_df("SELECT COUNT(*) AS c FROM Order_Table")["c"][0])
    col2.metric("Completed", fetch_df("SELECT COUNT(*) AS c FROM Order_Table WHERE Status Order='Completed'")["c"][0])
    col3.metric("Cancelled", fetch_df("SELECT COUNT(*) AS c FROM Order_Table WHERE Status Order='Cancelled'")["c"][0])

    st.subheader("Best Selling Cakes")
    df_best = fetch_df("""
        SELECT C.C_Name, COUNT(*) AS Sold
        FROM Order_Table O
        JOIN Cake_Catalogue C ON O.Cake_ID=C.Cake_ID
        WHERE StatusOrder='Completed'
        GROUP BY O.Cake_ID
        ORDER BY Sold DESC
    """)
    st.bar_chart(df_best.set_index("C_Name"))

# -----
# RUN SELECT QUERIES
# -----
elif menu == "Run Custom Query":
    st.header("🔍 Custom Query")

    sql = st.text_area("Enter SELECT query only")

    if st.button("Run"):
        if not sql.strip().lower().startswith("select"):
            st.error("Only SELECT queries are allowed.")
        else:
            st.dataframe(fetch_df(sql), use_container_width=True)
```

12. SQL queries(Create, Insert, Triggers, Procedures/Functions, Nested query, Join, Aggregate queries ) used in the project in the form of .sql file

.sql file code

```
1 CREATE DATABASE IF NOT EXISTS Cake_Store;
2 USE Cake_Store;
3
4 -- =====
5 -- Tables
6 -- =====
7
8 CREATE TABLE Payment (
9     Payment_ID INT PRIMARY KEY,
10    Amount DECIMAL(10,2),
11    Method VARCHAR(50),
12    Payment_Date DATE
13 );
14
15 CREATE TABLE Outlet (
16     Outlet_ID INT PRIMARY KEY,
17     Name VARCHAR(100),
18     Address VARCHAR(200),
19     Phone_Number VARCHAR(15)
20 );
21
22 CREATE TABLE Employees (
23     Employee_ID INT PRIMARY KEY,
24     E_Name VARCHAR(100),
25     RoleOfPerson VARCHAR(50),
26     Phone_Number VARCHAR(15),
27     Outlet_ID INT,
28     Production_ID INT,
29     FOREIGN KEY (Outlet_ID) REFERENCES Outlet(Outlet_ID)
30         ON DELETE CASCADE
31         ON UPDATE CASCADE
32 );
33
34 CREATE TABLE Customers (
35     Customer_ID INT PRIMARY KEY,
36     NameOfPerson VARCHAR(100),
37     Email VARCHAR(100),
38     PhoneNumber VARCHAR(15)
39 );
40
41 CREATE TABLE Cake_Catalogue (
```

```
40
41 CREATE TABLE Cake_Catalogue (
42     Cake_ID INT PRIMARY KEY AUTO_INCREMENT,
43     C_Name VARCHAR(100),
44     C_Category VARCHAR(50),
45     C_Description TEXT,
46     Price DECIMAL(8,2),
47     Quantity INT
48 );
49
50 CREATE TABLE Order_Table (
51     Order_ID INT PRIMARY KEY AUTO_INCREMENT,
52     Order_Date DATE,
53     StatusOrder VARCHAR(20),
54     Total_Amount DECIMAL(10,2),
55     Customer_ID INT,
56     Outlet_ID INT,
57     Cake_ID INT,
58     Payment_ID INT,
59     FOREIGN KEY (Customer_ID) REFERENCES Customers(Customer_ID)
60         ON DELETE CASCADE
61         ON UPDATE CASCADE,
62     FOREIGN KEY (Outlet_ID) REFERENCES Outlet(Outlet_ID)
63         ON DELETE CASCADE
64         ON UPDATE CASCADE,
65     FOREIGN KEY (Cake_ID) REFERENCES Cake_Catalogue(Cake_ID)
66         ON DELETE CASCADE
67         ON UPDATE CASCADE,
68     FOREIGN KEY (Payment_ID) REFERENCES Payment(Payment_ID)
69         ON DELETE CASCADE
70         ON UPDATE CASCADE
71 );
72
73 CREATE TABLE Customers_Address (
74     Customer_ID INT,
75     Address VARCHAR(200),
76     PRIMARY KEY (Customer_ID, Address),
77     FOREIGN KEY (Customer_ID) REFERENCES Customers(Customer_ID)
78         ON DELETE CASCADE
79         ON UPDATE CASCADE
80 );
```

```
81
82 CREATE TABLE Customers_Phone_Numbers (
83     Customer_ID INT,
84     Phone_Number VARCHAR(15),
85     PRIMARY KEY (Customer_ID, Phone_Number),
86     FOREIGN KEY (Customer_ID) REFERENCES Customers(Customer_ID)
87         ON DELETE CASCADE
88         ON UPDATE CASCADE
89 );
90
91 CREATE TABLE Outlet_Phone_Numbers (
92     Outlet_ID INT,
93     Phone_Number VARCHAR(15),
94     PRIMARY KEY (Outlet_ID, Phone_Number),
95     FOREIGN KEY (Outlet_ID) REFERENCES Outlet(Outlet_ID)
96         ON DELETE CASCADE
97         ON UPDATE CASCADE
98 );
99
100
101 -- =====
102 -- Inserts
103 -- =====
104
105 -- Payment
106 INSERT INTO Payment (Payment_ID, Amount, Method, Payment_Date) VALUES
107 (1, 450.00, 'Credit Card', '2025-01-12'),
108 (2, 720.50, 'UPI', '2025-01-15'),
109 (3, 299.99, 'Cash', '2025-02-01'),
110 (4, 1500.00, 'Debit Card', '2025-02-05');
111 □
112 -- Outlet
113 INSERT INTO Outlet (Outlet_ID, Name, Address, Phone_Number) VALUES
114 (1, 'Sweet Haven', 'MG Road, Bengaluru', '9876543210'),
115 (2, 'Cake World', 'Indiranagar, Bengaluru', '9988776655'),
116 (3, 'Bakers Hub', 'Koramangala, Bengaluru', '9123456780');
117
118 -- Employees
119 INSERT INTO Employees (Employee_ID, E_Name, RoleOfPerson, Phone_Number, Outlet_ID, Production_ID) VALUES
120 (1, 'Ramesh Kumar', 'Manager', '9123012301', 1, NULL),
```

```

119 INSERT INTO Employees (Employee_ID, E_Name, RoleOfPerson, Phone_Number, Outlet_ID, Production_ID) VALUES
120 (1, 'Ramesh Kumar', 'Manager', '9123012301', 1, NULL),
121 (2, 'Anita Sharma', 'Chef', '9898989898', 1, NULL),
122 (3, 'John Mathew', 'Cashier', '9000090000', 2, NULL),
123 (4, 'Sara Ali', 'Baker', '8080808080', 3, NULL);
124
125 -- Customers
126 INSERT INTO Customers (Customer_ID, NameOfPerson, Email, PhoneNumber) VALUES
127 (1, 'Aarav Gupta', 'aarav@example.com', '8881112222'),
128 (2, 'Riya Sharma', 'riya@example.com', '9991112222'),
129 (3, 'Karan Verma', 'karan@example.com', '7771112222'),
130 (4, 'Meera Rao', 'meera@example.com', '9662223333');
131
132 -- Cake Catalogue
133 INSERT INTO Cake_Catalogue (C_Name, C_Category, C_Description, Price, Quantity) VALUES
134 ('Chocolate Truffle', 'Chocolate', 'Rich dark chocolate cake', 450.00, 10),
135 ('Vanilla Delight', 'Classic', 'Soft vanilla sponge cake', 350.00, 20),
136 ('Red Velvet', 'Premium', 'Red velvet cream cheese cake', 550.00, 8),
137 ('Black Forest', 'Classic', 'Cherry and chocolate layered cake', 500.00, 15),
138
139 -- Orders
140 INSERT INTO Order_Table (Order_Date, StatusOrder, Total_Amount, Customer_ID, Outlet_ID, Cake_ID, Payment_ID) VALUES
141 ('2025-02-10', 'Completed', 450.00, 1, 1, 1, 1),
142 ('2025-02-12', 'Pending', 550.00, 2, 2, 3, 2),
143 ('2025-02-15', 'Completed', 350.00, 3, 1, 2, 3),
144 ('2025-02-16', 'Cancelled', 500.00, 4, 3, 4, 4);
145
146 -- Customer Addresses
147 INSERT INTO Customers_Address (Customer_ID, Address) VALUES
148 (1, 'HSR Layout, Bengaluru'),
149 (2, 'Whitefield, Bengaluru'),
150 (3, 'JP Nagar, Bengaluru'),
151 (4, 'BTM Layout, Bengaluru');
152
153 -- Customer Phone Numbers
154 INSERT INTO Customers_Phone_Numbers (Customer_ID, Phone_Number) VALUES
155 (1, '8881112222'),
156 (1, '8881113333'),
157 (2, '9991112222'),

```

```
156 (1, '8881113333'),
157 (2, '9991112222'),
158 (3, '7771112222');
159
160 -- Outlet Phone Numbers
161 INSERT INTO Outlet_Phone_Numbers (Outlet_ID, Phone_Number) VALUES
162 (1, '9876543210'),
163 (1, '9765432109'),
164 (2, '9988776655'),
165 (3, '9123456780');
166
167 -- =====
168 -- TRIGGERS
169 -- =====
170
171 -- 1. Prevent Negative Stock
172 DELIMITER //
173
174 CREATE TRIGGER trg_prevent_negative_stock
175 BEFORE UPDATE ON Cake_Catalogue
176 FOR EACH ROW
177 BEGIN
178     IF NEW.Quantity < 0 THEN
179         SIGNAL SQLSTATE '45000'
180         SET MESSAGE_TEXT = 'Cake stock cannot be negative!';
181     END IF;
182 END;
183 //
184
185 DELIMITER ;
186 □
187
188 -- 2. Auto Reduce Cake Stock on Order Insert
189 DELIMITER //
190
191 CREATE TRIGGER trg_reduce_cake_stock
192 AFTER INSERT ON Order_Table
193 FOR EACH ROW
194 BEGIN
195     UPDATE Cake_Catalogue
196     SET Quantity = Quantity - 1
```

```
195      UPDATE Cake_Catalogue
196          SET Quantity = Quantity - 1
197          WHERE Cake_ID = NEW.Cake_ID;
198    END;
199  //
200
201 DELIMITER ;
202
203
204 -- 3. Auto Calculate Order Total
205 DELIMITER //
206
207 CREATE TRIGGER trg_calculate_order_total
208 BEFORE INSERT ON Order_Table
209 FOR EACH ROW
210 BEGIN
211     IF NEW.Total_Amount IS NULL OR NEW.Total_Amount = 0 THEN
212         SET NEW.Total_Amount =
213             (SELECT Price FROM Cake_Catalogue WHERE Cake_ID = NEW.Cake_ID
214 );
215     END IF;
216 END;
217 //
218
219 DELIMITER ;
220
221
222 -- 4. Validate Payment Exists
223 DELIMITER //
224
225 CREATE TRIGGER trg_validate_payment
226 BEFORE INSERT ON Order_Table
227 FOR EACH ROW
228 BEGIN
229     IF (SELECT COUNT(*) FROM Payment WHERE Payment_ID = NEW.Payment_ID) = 0 THEN
230         SIGNAL SQLSTATE '45000'
231         SET MESSAGE_TEXT = 'Invalid Payment_ID! Payment record does not exist.';
232     END IF;
233 END;
234 //
235
```

```
236 DELIMITER ;
237
238 -- 5. Auto Increase Cake Stock on Order Cancellation
239 DELIMITER //
240
241 CREATE TRIGGER trg_increase_cake_stock_on_cancel
242 AFTER UPDATE ON Order_Table
243 FOR EACH ROW
244 BEGIN
245     -- Only trigger when status changes from NOT Cancelled → Cancelled
246     IF OLD.StatusOrder <> 'Cancelled' AND NEW.StatusOrder = 'Cancelled' THEN
247         UPDATE Cake_Catalogue
248             SET Quantity = Quantity + 1
249             WHERE Cake_ID = NEW.Cake_ID;
250     END IF;
251 END;
252 //
253
254 DELIMITER ;
255
256
257 -- =====
258 -- PROCEDURES
259 -- =====
260
261 DELIMITER //
262
263 -- 1. Procedure: Place an Order
264 CREATE PROCEDURE Place_Order (
265     IN p_customer INT,
266     IN p_outlet INT,
267     IN p_cake INT,
268     IN p_payment INT
269 )
270 BEGIN
271     INSERT INTO Order_Table (
272         Order_Date,
273         StatusOrder,
274         Customer_ID,
275         Outlet_ID,
276         Cake_ID,
```

```
277      Payment_ID
278    )
279    VALUES (
280      CURDATE(),
281      'Pending',
282      p_customer,
283      p_outlet,
284      p_cake,
285      p_payment
286    );
287 END;
288 //
289
290
291 -- 2. Procedure: Restock Cakes
292 CREATE PROCEDURE Restock_Cake (
293   IN p_cake_id INT,
294   IN p_add_quantity INT
295 )
296 BEGIN
297   UPDATE Cake_Catalogue
298   SET Quantity = Quantity + p_add_quantity
299   WHERE Cake_ID = p_cake_id;
300 END;
301 //
302
303 DELIMITER ;
304
305 -- =====
306 -- FUNCTIONS
307 [+] =====
308
309 DELIMITER //
310
311 -- 1. Function: Get Total Sales of a Cake
312 CREATE FUNCTION GetCakeSales(cake INT)
313 RETURNS DECIMAL(10,2)
314 DETERMINISTIC
315 BEGIN
316   RETURN (
317     SELECT SUM(Total_Amount)
```

```
306 -- FUNCTIONS
307 -- =====
308
309 DELIMITER //
310
311 -- 1. Function: Get Total Sales of a Cake
312 CREATE FUNCTION GetCakeSales(cake INT)
313 RETURNS DECIMAL(10,2)
314 DETERMINISTIC
315 BEGIN
316     RETURN (
317         SELECT SUM(Total_Amount)
318         FROM Order_Table
319         WHERE Cake_ID = cake AND StatusOrder = 'Completed'
320     );
321 END;
322 //
323
324
325 -- 2. Function: Check Stock Health
326 CREATE FUNCTION CheckStock(cake INT)
327 RETURNS VARCHAR(20)
328 DETERMINISTIC
329 BEGIN
330     DECLARE qty INT;
331
332     SELECT Quantity INTO qty
333     FROM Cake_Catalogue
334     WHERE Cake_ID = cake;
335
336     IF qty <= 0 THEN
337         RETURN 'OUT OF STOCK';
338     ELSEIF qty < 5 THEN
339         RETURN 'LOW STOCK';
340     ELSE
341         RETURN 'OK';
342     END IF;
343 END;
344 //
345
346 DELIMITER ;
```

### 13. Github repo link

<https://github.com/tanishhhegde/CakeStore-Database-Management-System>