PERSONAL HABIT TRACKER

Database Management System Project Report

Team Details

Team Members:

- Kathit Joshi (PES2UG23CS264)
- Kavyansh Jain (PES2UG23CS268)

Course: Database Management Systems

Institution: PES University

Date: 24 October 2025

1. Problem Statement

Develop a database-driven Personal Habit Tracker system that enables users to create, monitor, and analyze their daily habits through structured tracking mechanisms, goal setting, and performance analytics using MySQL and Python.

2. Abstract

The Personal Habit Tracker is a comprehensive database management system designed to help individuals build and maintain positive habits systematically. The system provides complete lifecycle management from user registration to performance analytics. Users can create multiple habits with customizable frequencies (Daily/Weekly/Monthly), set specific goals with deadlines, maintain detailed daily logs with status tracking (Completed/Pending/Skipped), and analyze performance through comprehensive reports. The project demonstrates practical implementation of database concepts including normalized schema design, stored procedures, user-defined functions, triggers for data validation, and complex SQL queries involving joins, nested queries, and aggregate functions. The application features a Python-based command-line interface with color-coded outputs and formatted table displays for enhanced user experience.

3. User Requirement Specification

3.1 Purpose

The system aims to provide a structured platform for habit tracking that helps users visualize progress, identify patterns, and maintain consistency in personal development goals through data-driven insights and accountability mechanisms.

3.2 Scope

The system encompasses user account management, habit creation with frequency tracking, goal setting with deadlines, daily activity logging with notes, automated data validation, and comprehensive analytics including completion rates, performance comparisons, and overdue goal identification.

3.3 Functional Requirements

User Management

- FR1: User registration with email and phone validation
- FR2: Profile updates (name, email, phone, password)
- FR3: Account deletion with cascading cleanup
- FR4: View all registered customers

Habit Management

- FR5: Create habits with start date and frequency
- FR6: View all habits with user information
- FR7: Delete habits with automatic cleanup

Goal Management

- FR8: Create goals with deadlines and descriptions
- FR9: View all goals with achievement status
- FR10: Mark goals as achieved via stored procedure

Logging System

- FR11: Create log entries with status and notes
- FR12: View recent logs across all habits
- FR13: View logs filtered by specific habit
- FR14: Update log status

Analytics

- FR15: Calculate habit completion rates via function
- FR16: Generate user performance summaries
- FR17: Generate habit performance reports
- FR18: Identify above-average performers

- FR19: Display habits with associated goals
- FR20: Identify and report overdue goals

Data Integrity

- FR21: Enforce referential integrity via foreign keys
- FR22: Validate log dates via trigger
- FR23: Input validation for all user entries
- FR24: Transaction management for consistency

4. Software and Tools Used

Database

- MySQL Server 8.0 - Relational database management
- MySQL Workbench 8.0 - Database design and administration

Programming Languages

- SQL - Database queries, procedures, functions, triggers
- Python 3.9 - Application logic and CLI interface

Python Libraries

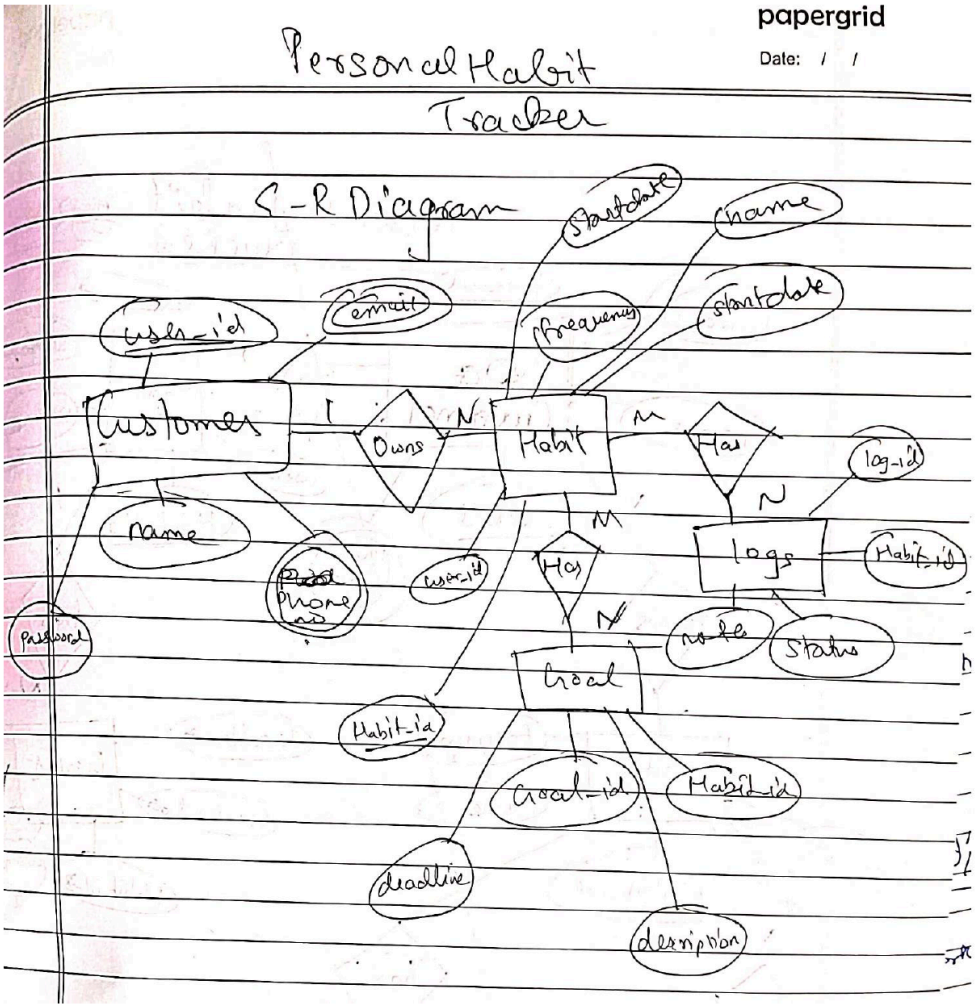
- mysql-connector-python 8.0.33 - Database connectivity
- tabulate 0.9.0 - Formatted table output
- re (built-in) - Input validation
- datetime (built-in) - Date handling

Development Environment

- Visual Studio Code - Code editor
- Git - Version control
- GitHub - Repository hosting

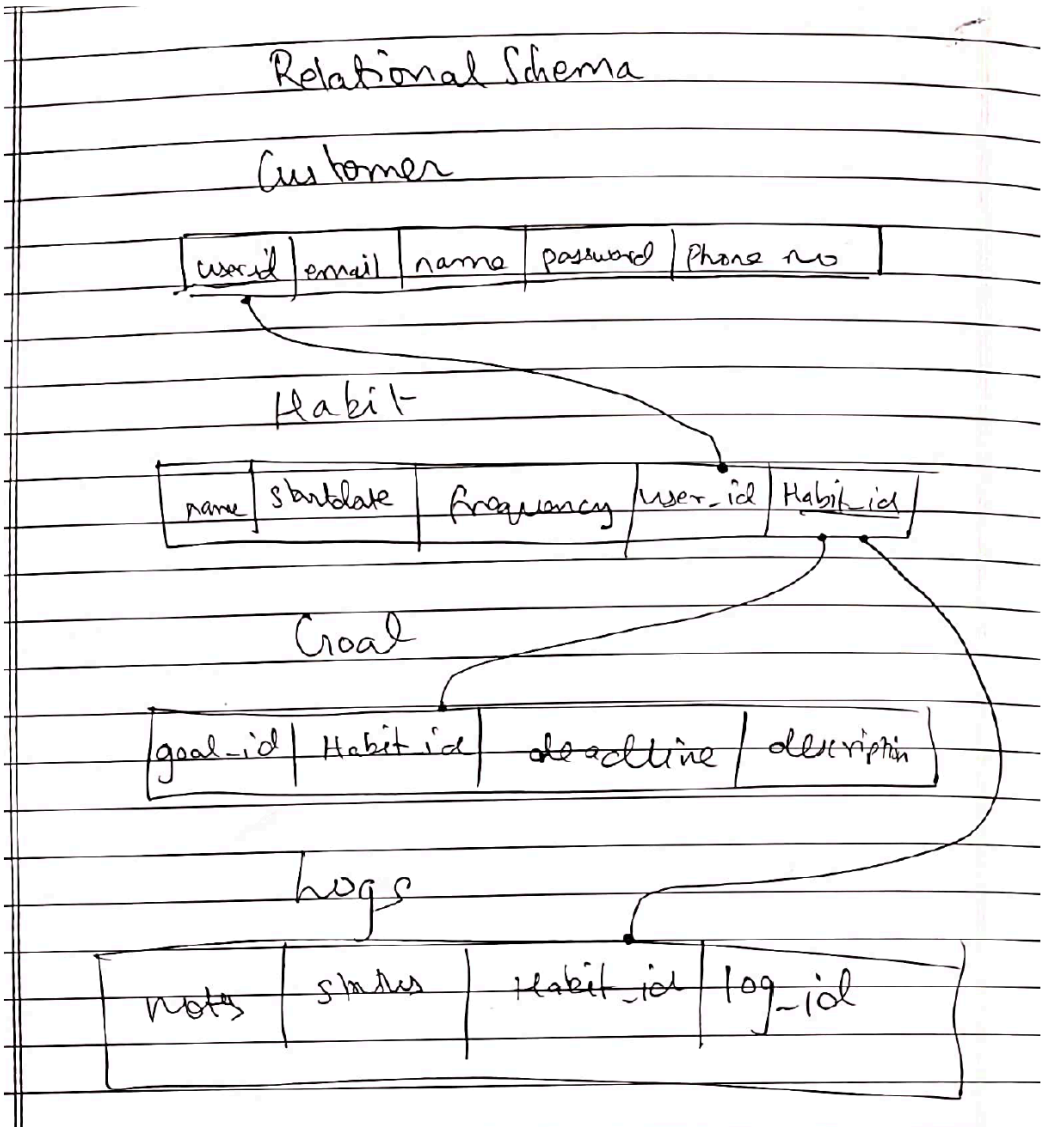
5. ER Diagram

ER DIAGRAM



6. Relational Schema

RELATIONAL SCHEMA



7. DDL Commands

Database Creation

```
CREATE DATABASE IF NOT EXISTS project;
USE project;
```

Table: Customer

```
CREATE TABLE Customer (
   user_id INT PRIMARY KEY,
   email VARCHAR(100) NOT NULL UNIQUE,
   name VARCHAR(100) NOT NULL,
   password VARCHAR(255) NOT NULL,
   phone_no VARCHAR(15) NOT NULL,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Table: Habit

```
CREATE TABLE Habit (
   habit_id INT PRIMARY KEY,
   user_id INT NOT NULL,
   name VARCHAR(100) NOT NULL,
   start_date DATE NOT NULL,
   frequency VARCHAR(20) NOT NULL,
   is_active BOOLEAN DEFAULT TRUE,
   CONSTRAINT fk_habit_customer FOREIGN KEY (user_id)
      REFERENCES Customer(user_id)
      ON DELETE CASCADE ON UPDATE CASCADE,
   CONSTRAINT chk_frequency CHECK (frequency IN ('Daily', 'Weekly', 'Monthly'))
);
```

Table: Goal

```
CREATE TABLE Goal (
   goal_id INT PRIMARY KEY,
   habit_id INT NOT NULL,
   deadline DATE NOT NULL,
   description TEXT NOT NULL,
   is_achieved BOOLEAN DEFAULT FALSE,
   CONSTRAINT fk_goal_habit FOREIGN KEY (habit_id)
      REFERENCES Habit(habit_id)
      ON DELETE CASCADE ON UPDATE CASCADE
);
```

Table: Logs

```
CREATE TABLE Logs (
    log_id INT PRIMARY KEY,
    habit_id INT NOT NULL,
    log_date DATE DEFAULT (CURRENT_DATE),
    notes TEXT,
    status VARCHAR(20) NOT NULL DEFAULT 'Pending',
    CONSTRAINT fk_logs_habit FOREIGN KEY (habit_id)
        REFERENCES Habit(habit_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT chk_status CHECK (status IN ('Completed', 'Pending', 'Skipped'))
);
```

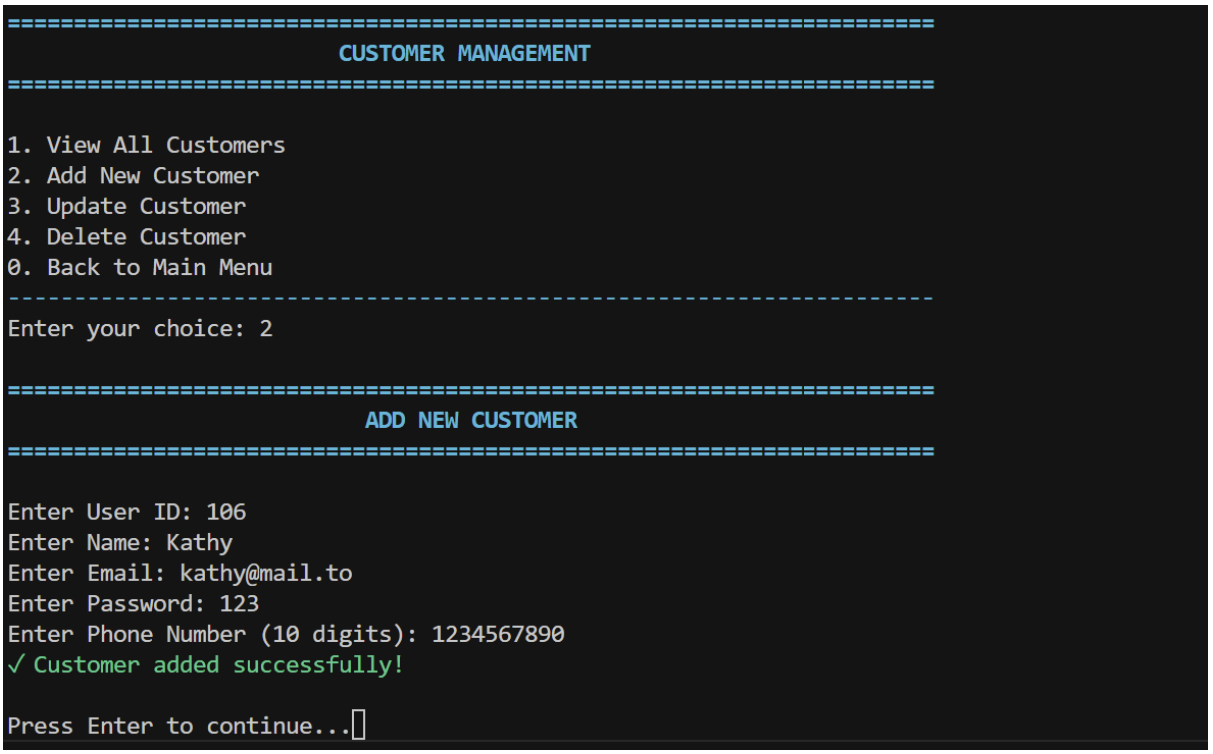
8. CRUD Operations

CREATE Operations

Add Customer

INSERT INTO Customer (user_id, email, name, password, phone_no)
VALUES (106, 'kathy@mail.to',”Kathy '', ‘123’', '1234567890’');

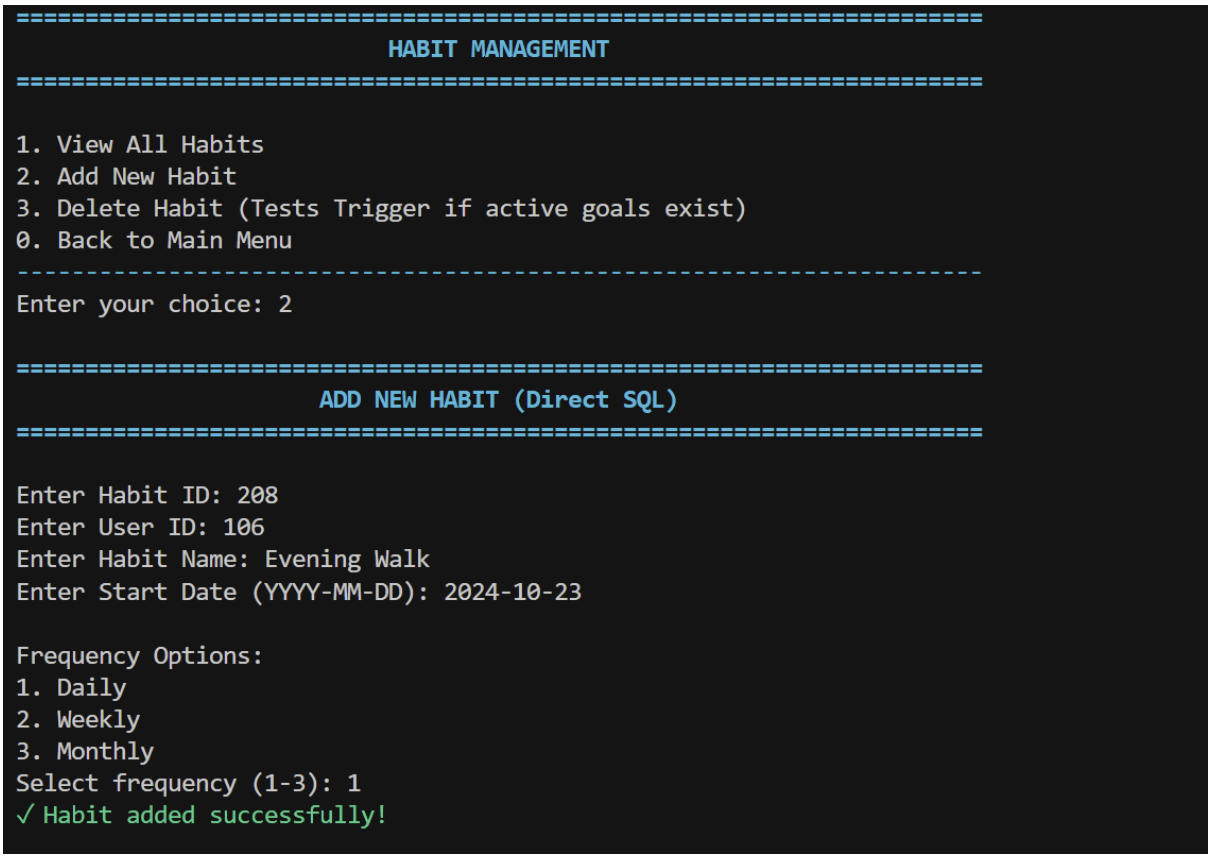
Screenshot:



Add Habit

INSERT INTO Habit (habit_id, user_id, name, start_date, frequency)
VALUES (208, 106, 'Evening Walk', '2024-10-23', 'Daily');

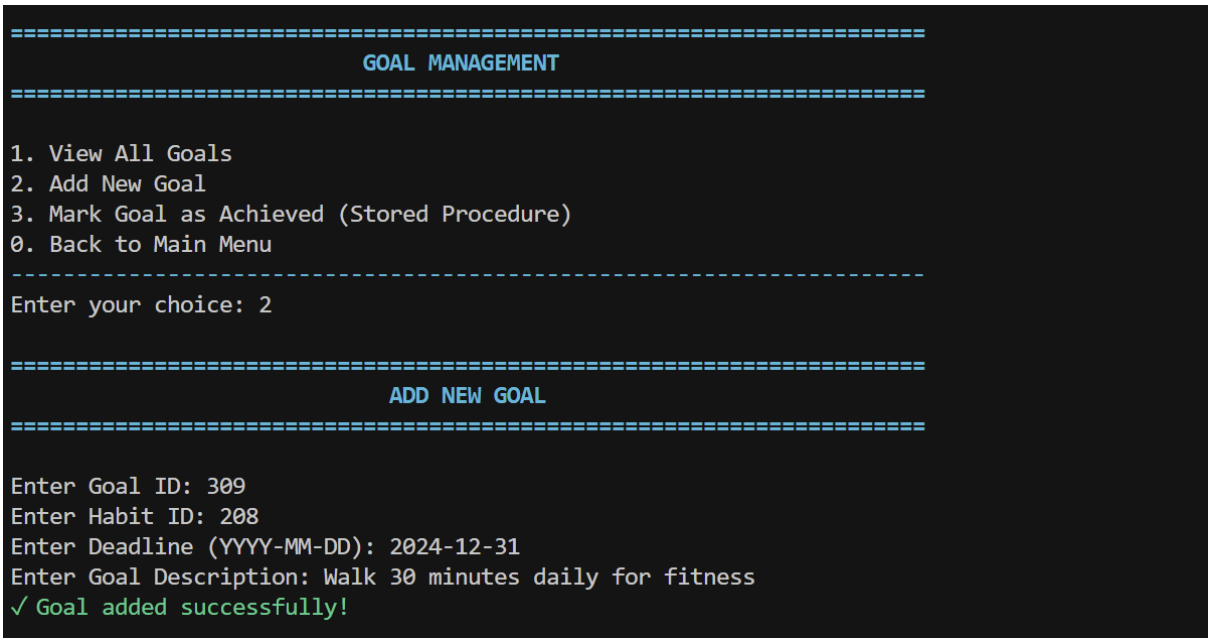
Screenshot:



Add Goal

INSERT INTO Goal (goal_id, habit_id, deadline, description)
VALUES (309, 208, '2024-12-31', 'Walk 30 minutes daily for fitness');

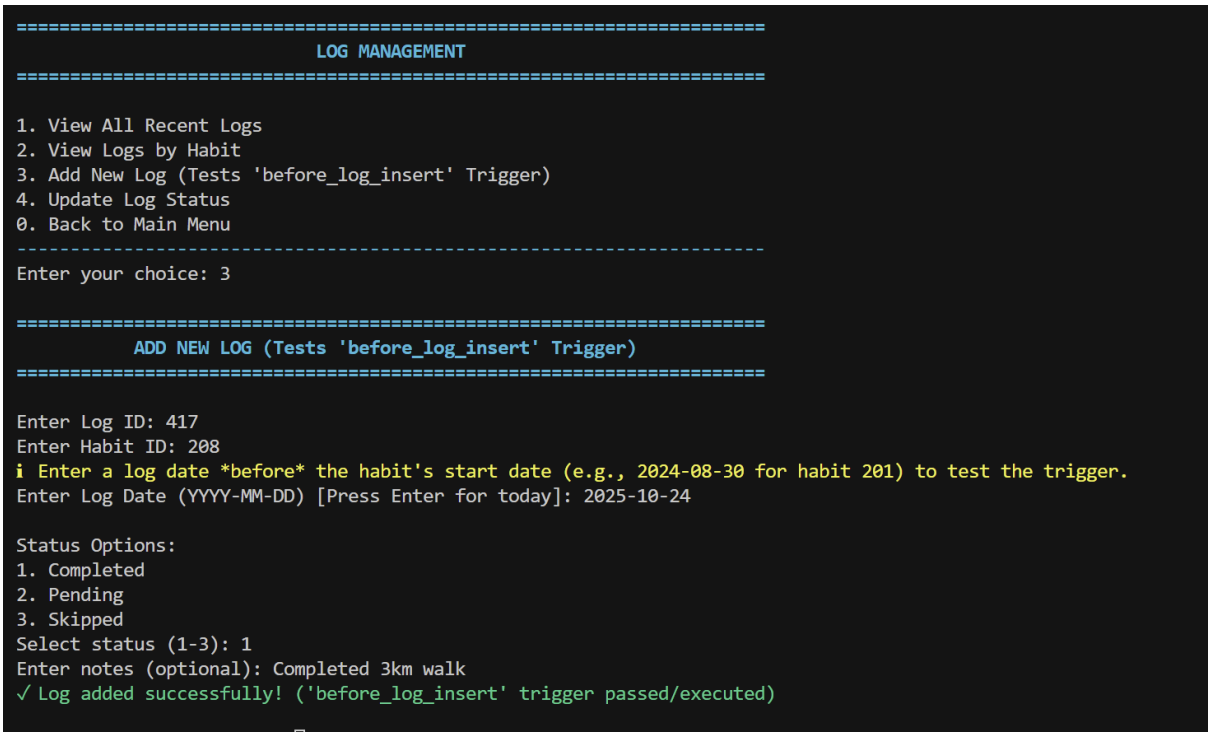
Screenshot:



Add Log

INSERT INTO Logs (log_id, habit_id, log_date, notes, status)
VALUES (417, 208, 2025-10-24, 'Completed 3km walk', 'Completed');

Screenshot:

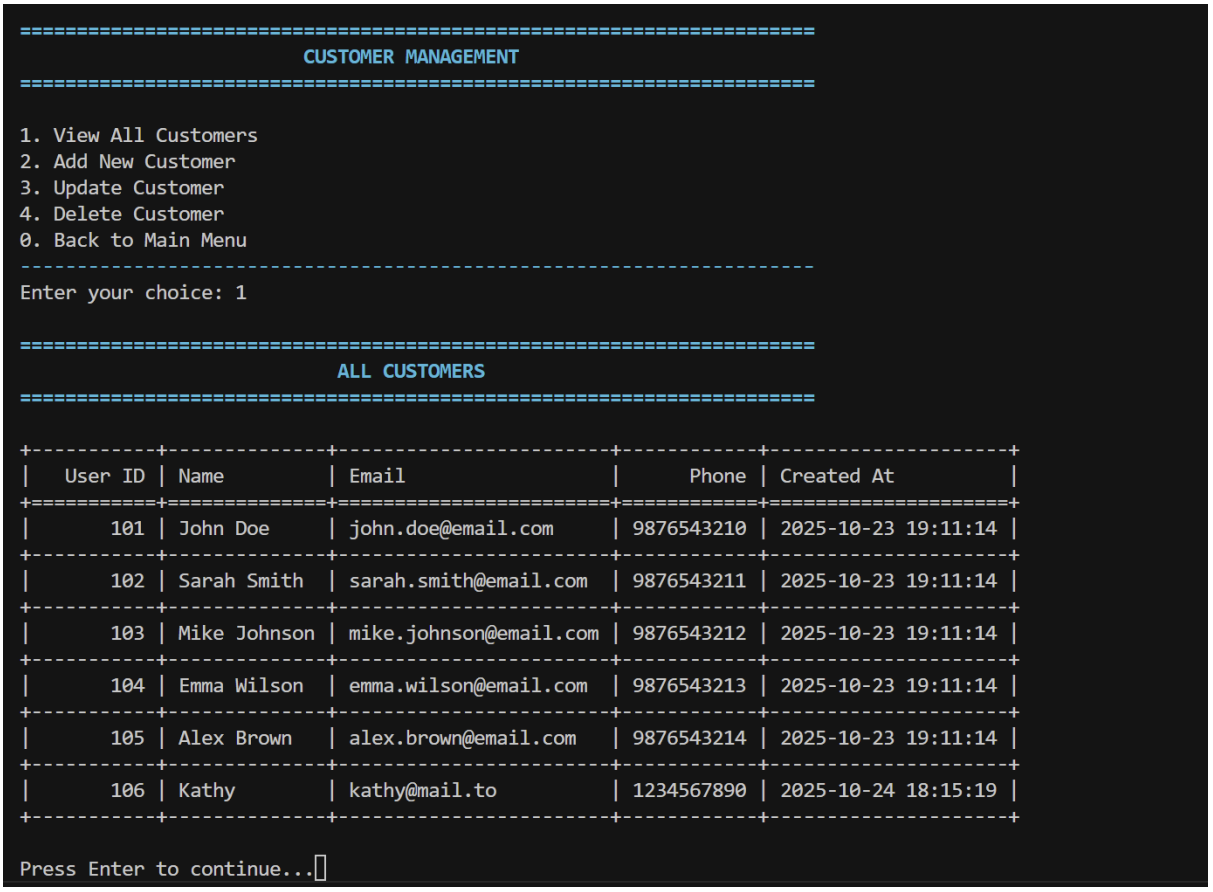


READ Operations

View All Customers

SELECT user_id, name, email, phone_no, created_at FROM Customer;

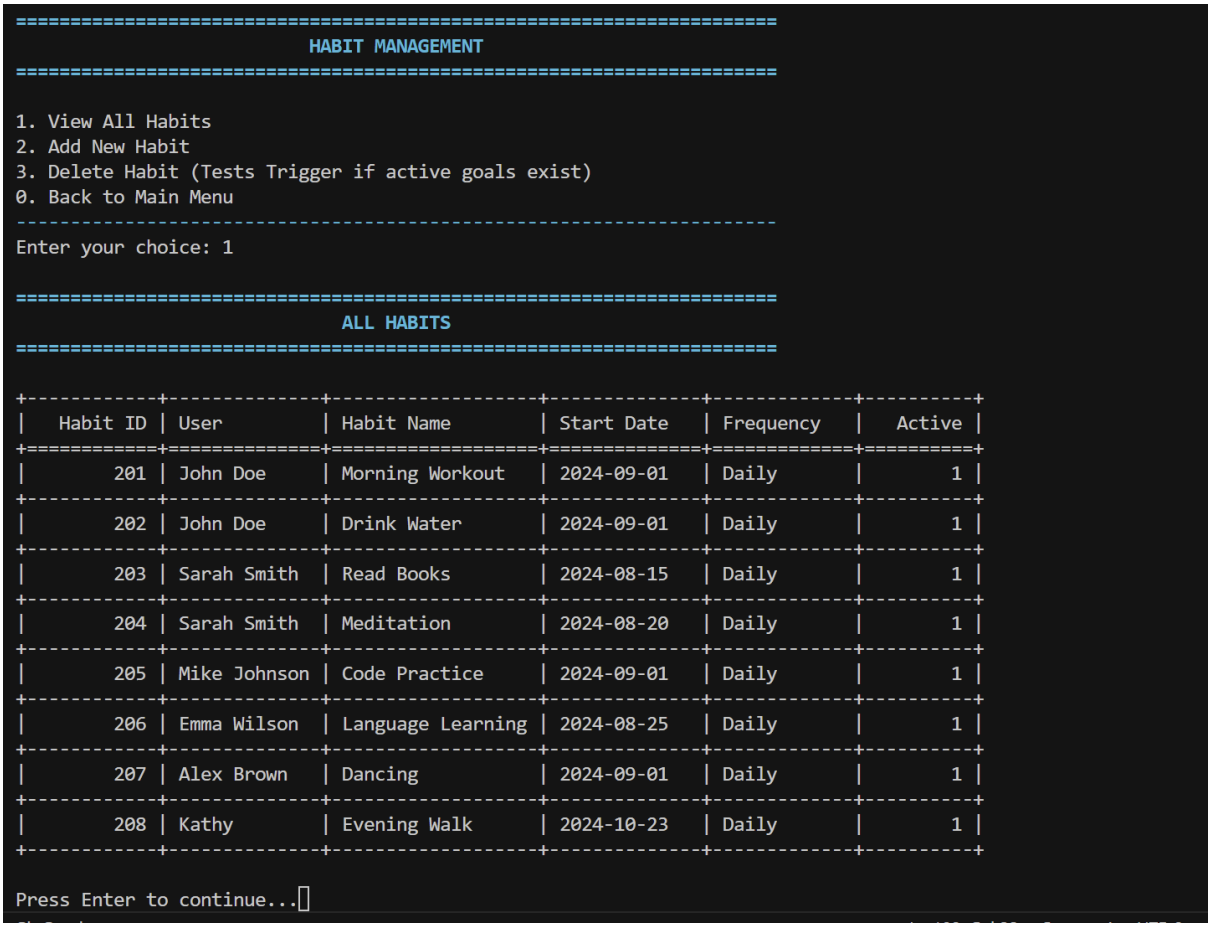
Screenshot:



View All Habits

```
SELECT h.habit_id, c.name, h.name, h.start_date, h.frequency, h.is_active
FROM Habit h
JOIN Customer c ON h.user_id = c.user_id;
```

Screenshot:

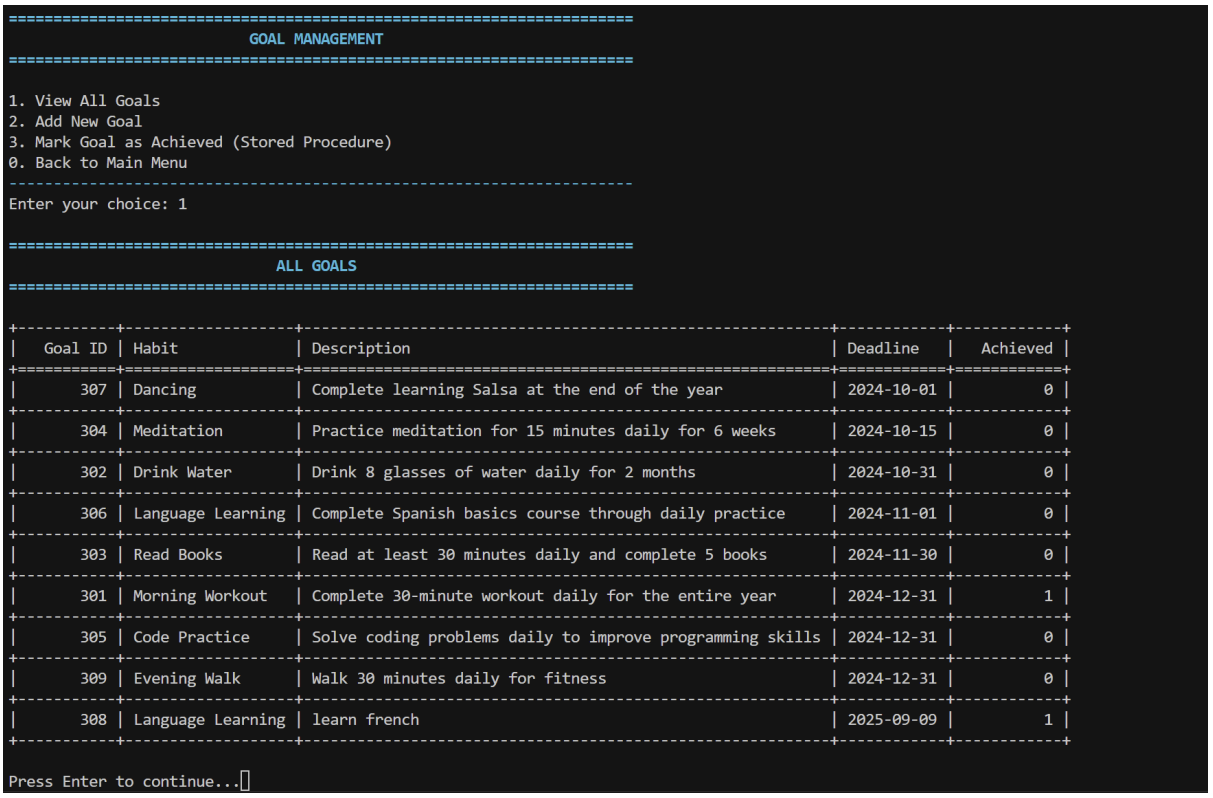


```
======================================================================
                        HABIT MANAGEMENT
======================================================================

1. View All Habits
2. Add New Habit
3. Delete Habit (Tests Trigger if active goals exist)
0. Back to Main Menu
----------------------------------------------------------------------
Enter your choice: 1

======================================================================
                          ALL HABITS
======================================================================

+------------+--------------+--------------------+--------------+-------------+----------+
|   Habit ID | User         | Habit Name         | Start Date   | Frequency   |   Active |
+============+==============+====================+==============+=============+==========+
|        201 | John Doe     | Morning Workout    | 2024-09-01   | Daily       |        1 |
+------------+--------------+--------------------+--------------+-------------+----------+
|        202 | John Doe     | Drink Water        | 2024-09-01   | Daily       |        1 |
+------------+--------------+--------------------+--------------+-------------+----------+
|        203 | Sarah Smith  | Read Books         | 2024-08-15   | Daily       |        1 |
+------------+--------------+--------------------+--------------+-------------+----------+
|        204 | Sarah Smith  | Meditation         | 2024-08-20   | Daily       |        1 |
+------------+--------------+--------------------+--------------+-------------+----------+
|        205 | Mike Johnson | Code Practice      | 2024-09-01   | Daily       |        1 |
+------------+--------------+--------------------+--------------+-------------+----------+
|        206 | Emma Wilson  | Language Learning  | 2024-08-25   | Daily       |        1 |
+------------+--------------+--------------------+--------------+-------------+----------+
|        207 | Alex Brown   | Dancing            | 2024-09-01   | Daily       |        1 |
+------------+--------------+--------------------+--------------+-------------+----------+
|        208 | Kathy        | Evening Walk       | 2024-10-23   | Daily       |        1 |
+------------+--------------+--------------------+--------------+-------------+----------+

Press Enter to continue...
```

View All Goals

```
SELECT g.goal_id, h.name, g.description, g.deadline, g.is_achieved
FROM Goal g
JOIN Habit h ON g.habit_id = h.habit_id;
```

Screenshot:



```
======================================================================
                        GOAL MANAGEMENT
======================================================================

1. View All Goals
2. Add New Goal
3. Mark Goal as Achieved (Stored Procedure)
0. Back to Main Menu
----------------------------------------------------------------------
Enter your choice: 1

======================================================================
                            ALL GOALS
======================================================================

+-----------+------------------+-------------------------------------------------------+------------+------------+
|   Goal ID | Habit            | Description                                           | Deadline   |   Achieved |
+===========+==================+=======================================================+============+============+
|       307 | Dancing          | Complete learning Salsa at the end of the year        | 2024-10-01 |          0 |
+-----------+------------------+-------------------------------------------------------+------------+------------+
|       304 | Meditation       | Practice meditation for 15 minutes daily for 6 weeks  | 2024-10-15 |          0 |
+-----------+------------------+-------------------------------------------------------+------------+------------+
|       302 | Drink Water      | Drink 8 glasses of water daily for 2 months           | 2024-10-31 |          0 |
+-----------+------------------+-------------------------------------------------------+------------+------------+
|       306 | Language Learning | Complete Spanish basics course through daily practice | 2024-11-01 |          0 |
+-----------+------------------+-------------------------------------------------------+------------+------------+
|       303 | Read Books       | Read at least 30 minutes daily and complete 5 books   | 2024-11-30 |          0 |
+-----------+------------------+-------------------------------------------------------+------------+------------+
|       301 | Morning Workout  | Complete 30-minute workout daily for the entire year  | 2024-12-31 |          1 |
+-----------+------------------+-------------------------------------------------------+------------+------------+
|       305 | Code Practice    | Solve coding problems daily to improve programming skills | 2024-12-31 |          0 |
+-----------+------------------+-------------------------------------------------------+------------+------------+
|       309 | Evening Walk     | Walk 30 minutes daily for fitness                     | 2024-12-31 |          0 |
+-----------+------------------+-------------------------------------------------------+------------+------------+
|       308 | Language Learning | learn french                                          | 2025-09-09 |          1 |
+-----------+------------------+-------------------------------------------------------+------------+------------+

Press Enter to continue...
```

View Recent Logs

```
SELECT l.log_id, h.name, l.log_date, l.status, l.notes
FROM Logs l
JOIN Habit h ON l.habit_id = h.habit_id
ORDER BY l.log_date DESC LIMIT 50;
```

Screenshot:

RECENT LOGS (Last 50)

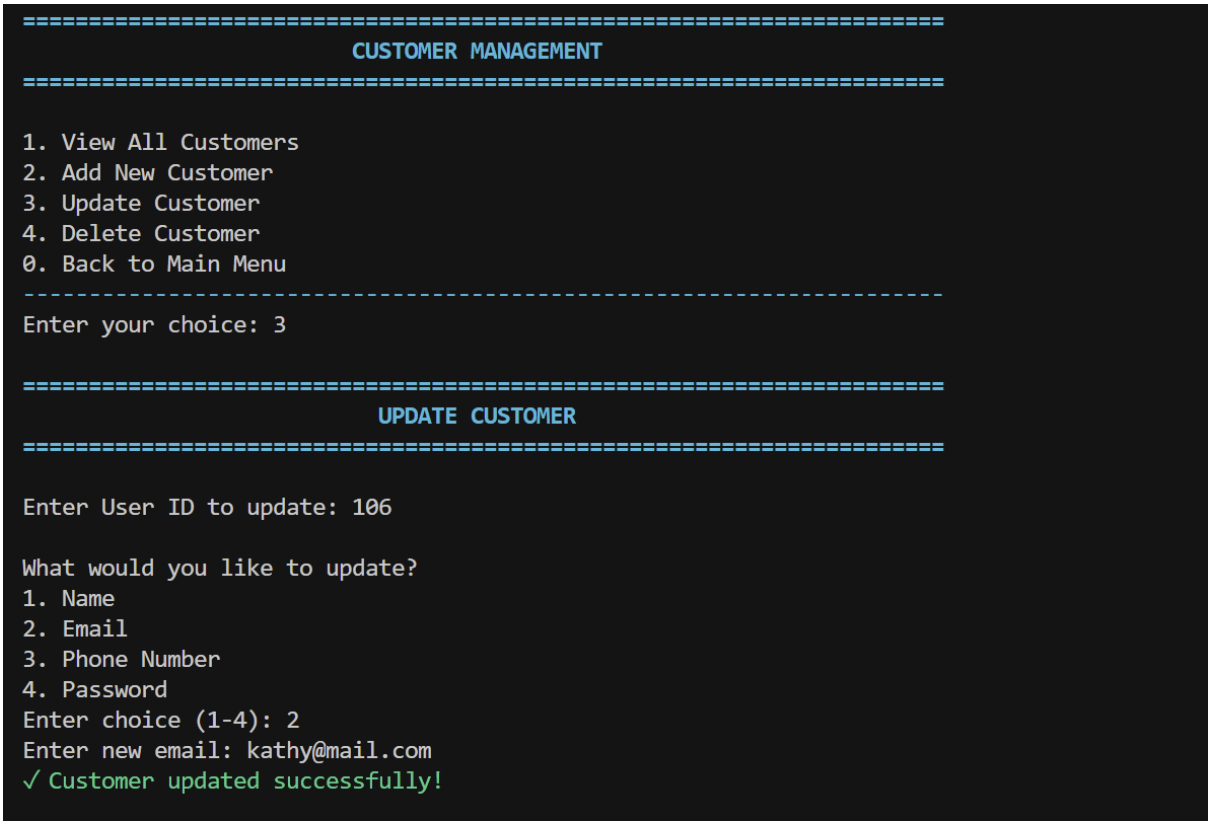
Log ID	Habit	Date	Status	Notes
417	Evening Walk	2025-10-24	Completed	Completed 3km walk
999	Morning Workout	2025-10-23	Completed	Trigger test pass
405	Morning Workout	2024-10-05	Completed	Full body workout completed successfully
404	Morning Workout	2024-10-04	Completed	Light yoga session due to tiredness
403	Morning Workout	2024-10-03	Skipped	Skipped due to illness
408	Drink Water	2024-10-03	Completed	Excellent hydration day
411	Read Books	2024-10-03	Pending	Finished reading productivity articles
414	Code Practice	2024-10-03	Completed	Working on dynamic programming concepts
402	Morning Workout	2024-10-02	Completed	Strength training focused on upper body
407	Drink Water	2024-10-02	Pending	Drank only 6 glasses, need to improve
410	Read Books	2024-10-02	Skipped	Too busy with work, could not read
413	Code Practice	2024-10-02	Completed	Practiced data structures - linked lists
401	Morning Workout	2024-10-01	Completed	Great cardio session with 20 minutes running
406	Drink Water	2024-10-01	Completed	Met daily water goal easily
409	Read Books	2024-10-01	Completed	Read Atomic Habits chapter on habit stacking
412	Code Practice	2024-10-01	Completed	Solved 3 algorithm problems on arrays
415	Language Learning	2024-10-01	Completed	Completed lesson on Spanish verb conjugation
416	Dancing	2024-10-01	Completed	Completed dancing lesson on Salsa
419	Language Learning	2024-08-30	Completed	Learned French

UPDATE Operations

Update Customer

UPDATE Customer SET email = 'kathy@mail.com' WHERE user_id = 106;

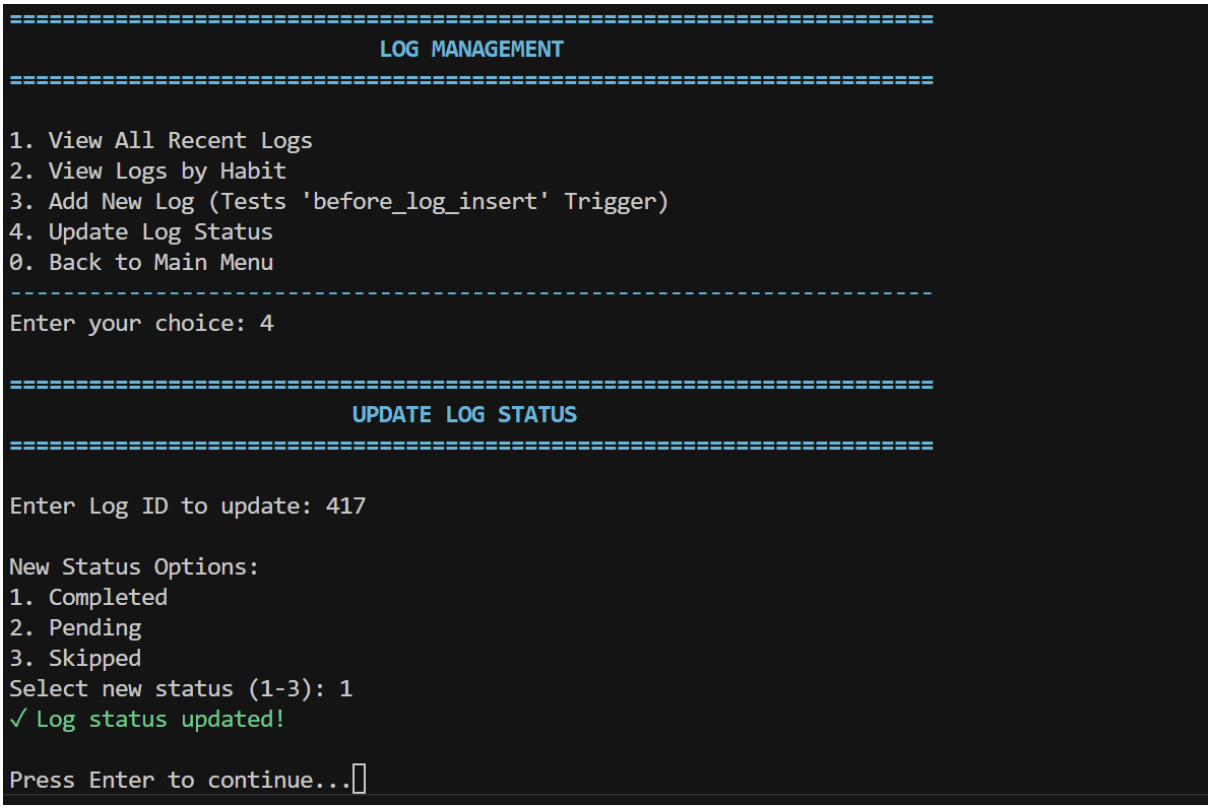
Screenshot:



Update Log Status

UPDATE Logs SET status = 'Completed' WHERE log_id = 417;

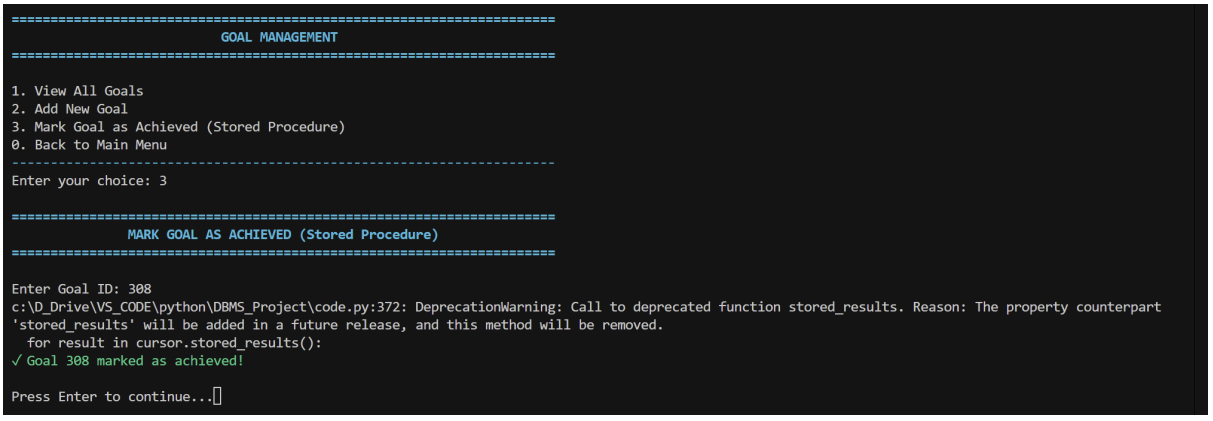
Screenshot:



Mark Goal Achieved (via Procedure)

CALL MarkGoalAchieved(308);

Screenshot:



DELETE Operations

Delete Customer

DELETE FROM Customer WHERE user_id = 106;

Screenshot:

```
================================================================================
                          CUSTOMER MANAGEMENT
================================================================================

1. View All Customers
2. Add New Customer
3. Update Customer
4. Delete Customer
0. Back to Main Menu
--------------------------------------------------------------------------------
Enter your choice: 4

================================================================================
                            DELETE CUSTOMER
================================================================================

Enter User ID to delete: 106
Are you sure you want to delete user 106? (yes/no): yes
✓ Customer deleted successfully!
```

Deleted Habit

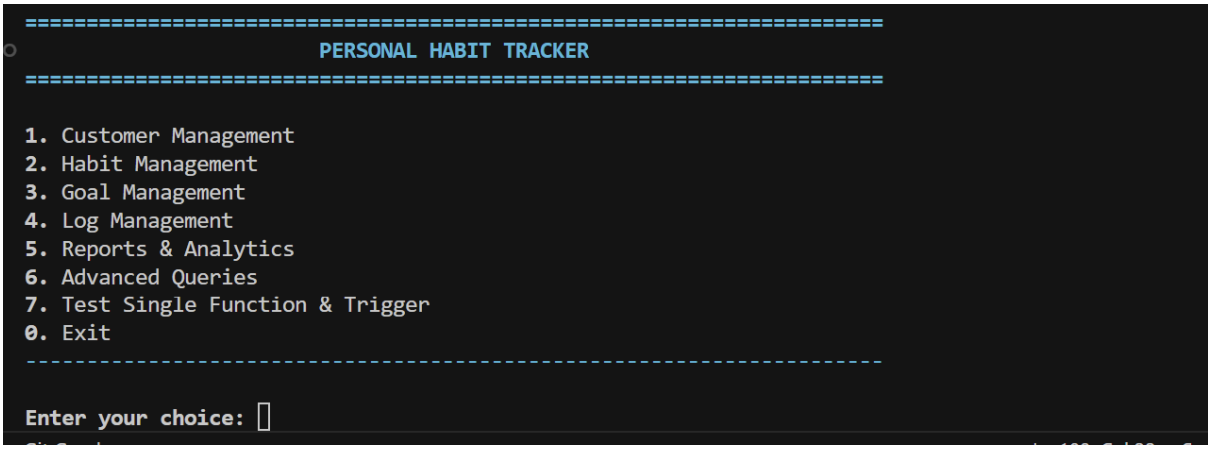
```
================================================================================
                          HABIT MANAGEMENT
================================================================================

1. View All Habits
2. Add New Habit
3. Delete Habit (Tests Trigger if active goals exist)
0. Back to Main Menu
--------------------------------------------------------------------------------
Enter your choice: 1

================================================================================
                             ALL HABITS
================================================================================

+------------+--------------+--------------------+--------------+-------------+----------+
|   Habit ID | User         | Habit Name         | Start Date   | Frequency   |   Active |
+============+==============+====================+==============+=============+==========+
|        201 | John Doe     | Morning Workout    | 2024-09-01   | Daily       |        1 |
+------------+--------------+--------------------+--------------+-------------+----------+
|        202 | John Doe     | Drink Water        | 2024-09-01   | Daily       |        1 |
+------------+--------------+--------------------+--------------+-------------+----------+
|        203 | Sarah Smith  | Read Books         | 2024-08-15   | Daily       |        1 |
+------------+--------------+--------------------+--------------+-------------+----------+
|        204 | Sarah Smith  | Meditation         | 2024-08-20   | Daily       |        1 |
+------------+--------------+--------------------+--------------+-------------+----------+
|        205 | Mike Johnson | Code Practice      | 2024-09-01   | Daily       |        1 |
+------------+--------------+--------------------+--------------+-------------+----------+
|        206 | Emma Wilson  | Language Learning  | 2024-08-25   | Daily       |        1 |
+------------+--------------+--------------------+--------------+-------------+----------+
|        207 | Alex Brown   | Dancing            | 2024-09-01   | Daily       |        1 |
+------------+--------------+--------------------+--------------+-------------+----------+

Press Enter to continue...
```

Habit gets deleted automatically as the user associated with it gets deleted here!!

9. Application Features and Screenshots

Main Menu

Screenshot:



Customer Management Module

- View Customers: Table display of all registered users
- Add Customer: Form with validation (email format, 10-digit phone)
- Update Customer: Field selection menu for profile updates
- Delete Customer: Confirmation dialog with cascade warning

Habit Management Module

- View Habits: List showing habits with owner names
- Add Habit: Frequency selection (Daily/Weekly/Monthly)
- Delete Habit: Removal with automatic cleanup

Goal Management Module

- View Goals: Table with deadline and achievement status
- Add Goal: Description input with date validation
- Mark Achieved: Stored procedure invocation

Log Management Module

- View All Logs: Recent 50 entries across habits
- View by Habit: Filtered log display

- Add Log: Status selection and notes input
- Update Status: Modify existing log entries

Reports and Analytics Module

- User Performance Summary: Aggregate statistics per user
- Habit Performance Report: Completion rates and breakdowns

Advanced Queries Module

- Above Average Users: Nested query results
- Habits with Goals: Join query display
- Overdue Goals: Date-based filtering

Testing Module

- Function Test: GetHabitCompletionRate demonstration
- Trigger Test: before_log_insert validation
- Procedure Test: MarkGoalAchieved execution

10. Database Objects

Stored Procedure

MarkGoalAchieved

- Purpose: Update goal achievement status
- Parameters: IN p_goal_id INT
- Operation: Sets is_achieved to TRUE for specified goal
- Returns: Confirmation message

```
CREATE PROCEDURE MarkGoalAchieved(IN p_goal_id INT)
BEGIN
   UPDATE Goal SET is_achieved = TRUE WHERE goal_id = p_goal_id;
   SELECT CONCAT('Goal ', p_goal_id, ' marked as achieved!') AS Message;
END;
```

Function

GetHabitCompletionRate

- Purpose: Calculate habit completion percentage
- Parameters: p_habit_id INT
- Returns: DECIMAL(5,2)
- Logic: Divides completed logs by total logs, handles zero division

```
CREATE FUNCTION GetHabitCompletionRate(p_habit_id INT)
RETURNS DECIMAL(5,2)
DETERMINISTIC READS SQL DATA
BEGIN
   DECLARE total_logs INT DEFAULT 0;
   DECLARE completed_logs INT DEFAULT 0;
   DECLARE completion_rate DECIMAL(5,2) DEFAULT 0.00;

   SELECT COUNT(*) INTO total_logs FROM Logs WHERE habit_id = p_habit_id;
   SELECT COUNT(*) INTO completed_logs
   FROM Logs WHERE habit_id = p_habit_id AND status = 'Completed';

   IF total_logs = 0 THEN RETURN 0.00; END IF;

   SET completion_rate = (completed_logs * 100.0) / total_logs;
   RETURN completion_rate;
END;
```

Trigger

before_log_insert

- Purpose: Validate log dates against habit start dates
- Timing: BEFORE INSERT
- Table: Logs
- Logic: Prevents insertion if log_date < habit start_date

```
CREATE TRIGGER before_log_insert
BEFORE INSERT ON Logs
FOR EACH ROW
BEGIN
   DECLARE habit_start DATE;
   SELECT start_date INTO habit_start FROM Habit WHERE habit_id = NEW.habit_id;
   IF NEW.log_date < habit_start THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Log date cannot be before habit start date';
   END IF;
END;
```

Nested Query

Users Above Average Performance

```
SELECT c.user_id, c.name,
      COUNT(CASE WHEN l.status = 'Completed' THEN 1 END) AS completed_logs
FROM Customer c
JOIN Habit h ON c.user_id = h.user_id
JOIN Logs l ON h.habit_id = l.habit_id
GROUP BY c.user_id, c.name
HAVING COUNT(CASE WHEN l.status = 'Completed' THEN 1 END) > (
   SELECT AVG(completed_count)
   FROM (
      SELECT COUNT(CASE WHEN status = 'Completed' THEN 1 END) AS
completed_count
      FROM Logs GROUP BY habit_id
   ) AS avg_table
)
ORDER BY completed_logs DESC;
```

Join Query

Habits with Goals

```
SELECT c.name AS user_name, h.name AS habit_name, h.frequency,
    g.description AS goal_description, g.deadline, g.is_achieved
FROM Customer c
JOIN Habit h ON c.user_id = h.user_id
LEFT JOIN Goal g ON h.habit_id = g.habit_id
ORDER BY c.name, h.name;
```

Aggregate Query

Habit Performance Statistics

```
SELECT h.name AS habit_name,
    COUNT(l.log_id) AS total_logs,
    SUM(CASE WHEN l.status = 'Completed' THEN 1 ELSE 0 END) AS completed,
    SUM(CASE WHEN l.status = 'Skipped' THEN 1 ELSE 0 END) AS skipped,
    SUM(CASE WHEN l.status = 'Pending' THEN 1 ELSE 0 END) AS pending,
    ROUND(AVG(CASE WHEN l.status = 'Completed' THEN 100 ELSE 0 END), 2) AS
completion_rate
FROM Habit h
LEFT JOIN Logs l ON h.habit_id = l.habit_id
GROUP BY h.habit_id, h.name
HAVING COUNT(l.log_id) > 0
ORDER BY completion_rate DESC;
```

11. Code Snippets for Invocation

Invoking Stored Procedure

Python Code:

```
def mark_goal_achieved(connection):
    goal_id = int(input("Enter Goal ID: "))
    cursor = connection.cursor()
    cursor.callproc('MarkGoalAchieved', [goal_id])
    for result in cursor.stored_results():
        print(result.fetchone()[0])
    connection.commit()
    cursor.close()
```

Direct SQL:

```
CALL MarkGoalAchieved(301);
```

Invoking Function

Python Code:

```
def test_completion_rate(connection):
    habit_id = int(input("Enter Habit ID: "))
    cursor = connection.cursor()
    cursor.execute("SELECT GetHabitCompletionRate(%s)", (habit_id,))
    result = cursor.fetchone()
    print(f"Completion Rate: {result[0]}%")
    cursor.close()
```

Direct SQL:

```
SELECT GetHabitCompletionRate(201) AS completion_rate;
```

Testing Trigger

Python Code:

```
def add_log(connection):
    log_id = int(input("Enter Log ID: "))
    habit_id = int(input("Enter Habit ID: "))
    log_date = input("Enter Log Date (YYYY-MM-DD): ")
    status = input("Status (Completed/Pending/Skipped): ")
```

```
    notes = input("Notes: ")

    cursor = connection.cursor()
    query = "INSERT INTO Logs (log_id, habit_id, log_date, notes, status) VALUES (%s, %s,
%s, %s, %s)"
    cursor.execute(query, (log_id, habit_id, log_date, notes, status))
    connection.commit()
    cursor.close()
```

Trigger Test (Should Fail):

```
-- Habit 201 starts on 2024-09-01
INSERT INTO Logs (log_id, habit_id, log_date, notes, status)
VALUES (999, 201, '2024-08-30', 'Test trigger', 'Completed');
-- Error: Log date cannot be before habit start date
```

Trigger Test (Should Pass):

```
INSERT INTO Logs (log_id, habit_id, log_date, notes, status)
VALUES (999, 201, CURRENT_DATE, 'Test trigger', 'Completed');
-- Success
```

12. SQL File Contents

The project.sql file contains:

1. **Database Creation**
    - CREATE DATABASE statement
    - USE statement
2. **Table Definitions**
    - Customer table with constraints
    - Habit table with foreign key and check constraint
    - Goal table with foreign key
    - Logs table with foreign key and check constraint
3. **Sample Data**
    - 5 customer records
    - 7 habit records
    - 7 goal records
    - 16 log records
4. **Stored Procedure**
    - MarkGoalAchieved definition
5. **Function**
    - GetHabitCompletionRate definition
6. **Trigger**
    - before_log_insert definition
7. **Complex Queries**
    - Join query for habits with goals
    - Nested query for above-average users
    - Aggregate query for habit performance
    - Date-based query for overdue goals
8. **Test Statements**
    - Procedure invocation test
    - Function invocation test
    - Trigger validation tests

13. GitHub Repository

Repository Name: Personal-Habit-Tracker

Repository URL: https://github.com/Kathitjoshi/Personal-Habit-Tracker

Repository Structure:

```
personal-habit-tracker/
├── project.sql            # Complete database schema and queries
├── code.py                # Python CLI application
├── README.md                  # Project documentation
├── Report.pdf               # Report
```

Key Files:

- project.sql: Contains all DDL, DML, stored procedures, functions, triggers, and queries
- code.py: Python application with database connectivity and CLI interface
- README.md: Comprehensive project documentation
- Report.pdf : This

Conclusion

The Personal Habit Tracker successfully demonstrates comprehensive database management concepts through a practical application. The system implements normalized schema design, referential integrity through foreign keys, data validation via triggers, reusable logic through stored procedures and functions, and complex data analysis using joins, nested queries, and aggregate functions. The Python-based CLI provides an intuitive interface for all operations with proper error handling and formatted outputs. The project serves as a complete example of database-driven application development from design to implementation.