**GAK: Integrated Student Academic and Lifestyle System**

21CSC205P Database Management Systems

A MINI PROJECT REPORT

*Submitted by*

D Gruhith [RA2411026010849]

G Manohar [RA2411026010848]

*Under the Guidance of*

**DR.KRUNAL DHANRAJ RANDIVE**

Assistant Professor, Department of Computational Intelligence

*in partial fulfillment of the requirements* *for the degree of*

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING

with specialization in (AL&ML)

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR- 603 203

MAY 2026



SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that Project report titled “**GAK: Integrated Student Academic and Lifestyle system** ” is the bonafide work of “**D Gruhith [RA2411026010849], G Manohar [RA2411026010848]”** who carried out the **21CSC205P Database Management Systems** mini project work under my supervision.

|  |  |
| --- | --- |
| **COURSE FACULTY** | **HEAD OF THE DEPARMENT** |
| DESIGNATION  Department | PROFESSOR AND HEAD  Department |

**ABSTRACT**

Gyaan–Karma–Ahara (GAK) is a unified decision-support system designed to integrate and manage student academic, fitness, and nutrition data within a single, normalized database framework. The system addresses the fragmentation of student information across academic records, timetables, attendance logs, physical activity tracking, and food intake monitoring by modeling them under a shared institutional structure.

The proposed design supports multi-university and multi-campus environments, enabling shared academic calendars and unified timetables while maintaining user-specific performance and lifestyle data. Attendance and marks are captured at a fine-grained level using timetable-based records, allowing accurate projections and decision support without modifying official data. Fitness activities, body metrics, and nutrition logs are incorporated to provide a holistic view of student behavior and well-being.

The database schema is fully normalized, scalable, and designed to handle large user populations efficiently. Derived insights such as attendance percentages, academic risk status, calorie summaries, and goal progress are computed dynamically from stored base data, ensuring data consistency and integrity. Overall, GAK demonstrates how structured data modeling can enable informed academic and lifestyle planning through a centralized, extensible decision-support system.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Title** | **Page no.** |
|  | ABSTRACT |  |
| 1 | Problem understanding, Identification of Entity and Relationships, Construction of DB using ER Model for their project |  |
| 2 | Design of Relational Schemas, Creation of Database and Tables for their project |  |
| 3 | 3.1Writing queries based on the concepts of Aggregate functions, constraints, sets |  |
| 3.2 Writing complex queries based on subqueries, joins and views |  |
| 3.3 Writing complex queries based on the concepts of functions, Triggers, Cursors and Exception Handling |  |
| 4 | 4.1 Analysing the pitfalls, identifying the dependencies, and applying Normalizations- 1NF, 2NF, 3NF |  |
| 4.2 Analysing the pitfalls, identifying the dependencies, and applying Normalizations- 4NF, 5NF |  |
| 5 | Implementation of concurrency control and recovery mechanisms. |  |
| 6 | Front-end And Back-End Implementation code |  |
| 7 | Results and Discussions |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Name** | **Page no.** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Name** | **Page no.** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |

**CHAPTER 1**

**PROBLEM UNDERSTANDING, IDENTIFICATION OF ENTITY AND RELATIONSHIPS, CONSTRUCTION OF DB USING ER MODEL FOR THE GAK : Integrated Student Academic and Lifestyle system**

* 1. **Introduction**

GAK (Gyaan–Ahara–Karma) is a unified, multi-source decision-support system designed to manage and analyze student academic and lifestyle data. It integrates academic information such as attendance, marks, timetables, and academic calendars with fitness activities and nutrition data. The system is designed as a centralized database-driven platform that supports structured data storage, efficient querying, and analytical decision-making rather than a simple application-centric approach

* 1. **Motivation**

The motivation for this project arises from the lack of an integrated system that combines academic performance tracking with lifestyle and health monitoring for university students.

* Academic data such as attendance and marks are stored separately from daily routines.
* Students struggle to understand the impact of missed classes, workouts, or poor nutrition on long-term performance.
* Existing systems focus on isolated analytics rather than holistic decision support.
* Manual tracking of fitness and food intake is inconsistent and error-prone.
* Institutional data such as academic calendars and timetables are duplicated or poorly utilized.

The GAK system is motivated by the need to centralize, normalize, and analyze multi-source student data to support informed academic and lifestyle decisions.

* 1. **Scope**

The scope of this project is limited to the design and implementation of a relational database-backed web system that integrates academic, fitness, and nutrition data for students across multiple campuses and academic units.

The system allows:

* Storage and reuse of shared institutional data such as academic calendars and unified timetables
* User-specific tracking of attendance, marks, goals, workouts, and food intake
* Integration readiness for external sources such as calendars and fitness platforms
* Decision-support features such as attendance prediction and academic risk identification

The project does not include real-time wearable device integration, online payments, or medical diagnostics. The primary focus remains on database design, normalization, and relational integrity.

* 1. **Problem Statement**

University students currently manage academic records, fitness activities, and dietary habits using disconnected systems. This fragmented approach makes it difficult to analyze trends, predict outcomes, or make informed decisions regarding attendance, academic performance, or personal health.

There is a need for a “centralized, normalized, and scalable database system “ that can integrate institutional academic data with user-specific lifestyle data while maintaining data integrity and reusability.

The GAK project aims to address this problem by modeling and implementing a unified decision-support system using a relational database approach.

* 1. **Project Requirements**

### Functional Requirements

* Store institutional academic data (universities, campuses, academic units)
* Maintain unified timetables and academic calendars shared across users
* Record user-specific attendance, marks, goals, fitness activities, and food intake
* Support analytical queries such as attendance percentage and academic risk status

### Non-Functional Requirements

* Data normalization to reduce redundancy
* Scalability to support multiple campuses and thousands of users
* Data integrity using primary keys, foreign keys, and constraints
* Efficient query performance using indexing

## Software Requirements

* Frontend: React 18, TypeScript, Vite, Tailwind CSS
* Backend: Node.js with Express.js or NestJS
* Database: MySQL or PostgreSQL (Relational DBMS)
* Runtime Environment: Node.js (version 18 or above)
* Development Tools: VS Code, MySQL Workbench
* Browser: Google Chrome / Firefox / Edge
  1. **Identification of Entity and Relationships**

The proposed **Gyaan–Karma–Ahara (GAK)** system is designed as a unified decision-support platform that integrates academic management, fitness tracking, and nutrition monitoring for students across multiple universities, campuses, and academic units. Based on the system requirements and the finalized database schema, the following entities and relationships have been identified.

### Identified Entities

| Entity | Description |
| --- | --- |
| University | Stores details of universities using the system |
| Campus | Represents different campuses under a university |
| Academic\_Unit | Represents academic divisions such as Engineering, Law, etc. |
| Academic\_Calendar | Stores shared academic dates, holidays, and day-order information |
| Unified\_Timetable | Represents official institution-level timetables shared across users |
| Timetable\_Entry | Stores individual class sessions derived from unified timetables |
| Faculty | Stores faculty details involved in teaching |
| Classroom | Stores classroom and building information |
| Subject | Represents academic subjects offered by academic units |
| Section | Represents sections/batches of students |
| User | Stores student account information |
| Academic\_Profile | Links users to their institutional and academic context |
| Attendance\_Record | Tracks per-class attendance of users |
| Marks\_Record | Stores assessment and evaluation data |
| Academic\_Goal | Stores academic targets set by users |
| Workout\_Plan | Stores fitness plans assigned to users |
| Workout\_Session | Stores individual workout sessions |
| Workout\_Action | Stores completion or skip status of workouts |
| Activity\_Log | Stores physical activities and calorie data |
| Body\_Metric | Stores physical attributes such as height and weight |
| Food\_Image | Stores uploaded food images |
| Detected\_Food\_Item | Stores AI-detected food items from images |
| Confirmed\_Food\_Item | Stores user-confirmed food items and nutritional values |
| Food\_Log | Stores daily aggregated nutrition data |
| Calendar\_Event | Stores user-specific personal events |
| Integration\_Status | Stores third-party integration details |

Table 1.1 : Entity Description Diagram

### Identified Relationships

* University ↔ Campus → One-to-Many (Has)
* Campus ↔ Academic\_Unit → One-to-Many (Contains)
* Academic\_Unit ↔ Academic\_Calendar → One-to-Many (Follows)
* Academic\_Unit ↔ Unified\_Timetable → One-to-Many (Uses)
* Unified\_Timetable ↔ Timetable\_Entry → One-to-Many (Contains)
* Timetable\_Entry ↔ Subject → Many-to-One (Scheduled For)
* Timetable\_Entry ↔ Faculty → Many-to-One (Taught By)
* Timetable\_Entry ↔ Classroom → Many-to-One (Conducted In)
* Timetable\_Entry ↔ Section → Many-to-One (Assigned To)
* User ↔ Academic\_Profile → One-to-One (Has)
* User ↔ Attendance\_Record → One-to-Many (Marks)
* Subject ↔ Attendance\_Record → One-to-Many (Tracked For)
* User ↔ Marks\_Record → One-to-Many (Receives)
* Subject ↔ Marks\_Record → One-to-Many (Evaluated By)
* User ↔ Academic\_Goal → One-to-Many (Sets)
* User ↔ Workout\_Plan → One-to-Many (Has)
* Workout\_Plan ↔ Workout\_Session → One-to-Many (Contains)
* Workout\_Session ↔ Workout\_Action → One-to-Many (Records Status)
* User ↔ Activity\_Log → One-to-Many (Performs)
* User ↔ Body\_Metric → One-to-Many (Records)
* User ↔ Food\_Image → One-to-Many (Uploads)
* Food\_Image ↔ Detected\_Food\_Item → One-to-Many (Contains)
* Detected\_Food\_Item ↔ Confirmed\_Food\_Item → One-to-One (Verifies)
* User ↔ Food\_Log → One-to-Many (Records)
* User ↔ Calendar\_Event → One-to-Many (Creates)
* User ↔ Integration\_Status → One-to-Many (Has)
  1. **Construction of DB Using ER Model for the GAK: Integrated Student Academic and Lifestyle system**

### 1.7.1 ER to Relational Mapping

The Entity–Relationship (ER) diagram designed for the Residential Accessibility and User Coordination system was systematically converted into a relational database schema. This conversion was carried out using standard ER-to-relational mapping rules to ensure data integrity and normalization.

The mapping process involved:

* Mapping each identified entity into a separate relational table
* Assigning primary keys to uniquely identify records in each table
* Implementing relationships between entities using foreign keys
* Creating associative (junction) tables to resolve many-to-many relationship

This approach ensures minimal redundancy and maintains consistency across the database.

### 1.7.2 Database Implementation

The database was implemented using MySQL, a widely used relational database management system.

The schema includes tables such as User, Admin, Flat, Area, Owner, Interest\_Request, Consultant\_Note, Flat\_Image, and Availability\_Status, which were created using CREATE TABLE statements.

Each table was defined with appropriate data types, primary keys, and foreign key constraints to accurately represent the relationships specified in the ER diagram.

1.7.3 Constraints Used

The following constraints were applied to maintain data accuracy and integrity:

* Primary Key – Ensures unique identification of each record
* Foreign Key – Maintains referential integrity between related tables
* Associative Keys – Used in junction tables to resolve many-to-many relationships between users and flats

These constraints help enforce valid relationships and prevent inconsistent data entries.

1.7.4 Data Population

Sample data was inserted into all tables using INSERT INTO statements.

This step was performed to validate entity relationships, test constraint enforcement, and ensure that the database supports the required system operations effectively.

1.7.5 Query Implementation

The system supports several essential queries, including:

* Retrieval of available flats based on area and availability status
* Display of flat details along with consultant notes and images
* Tracking and management of user interest requests
* Viewing the status of interest requests
* Accessing flat owner details through consultant-mediated control

These queries enable efficient information retrieval and support the core functionality of the system.

1.7.6 Views Creation

Database views were created to simplify frequently used and complex queries.

Examples of views include:

* Flat\_List\_View
* User\_Interest\_View
* Consultant\_Flat\_Details\_View

These views improve query readability, enhance reporting efficiency, and reduce the complexity of repeated query execution.

**ER Diagram:**

**Chapter 2**

**Design of Relational Schemas, Creation of Database and Tables for GAK: Integrated Student Academic and Lifestyle system**

**2.1 Relational Schema for GAK: Integrated Student Academic and Lifestyle system**

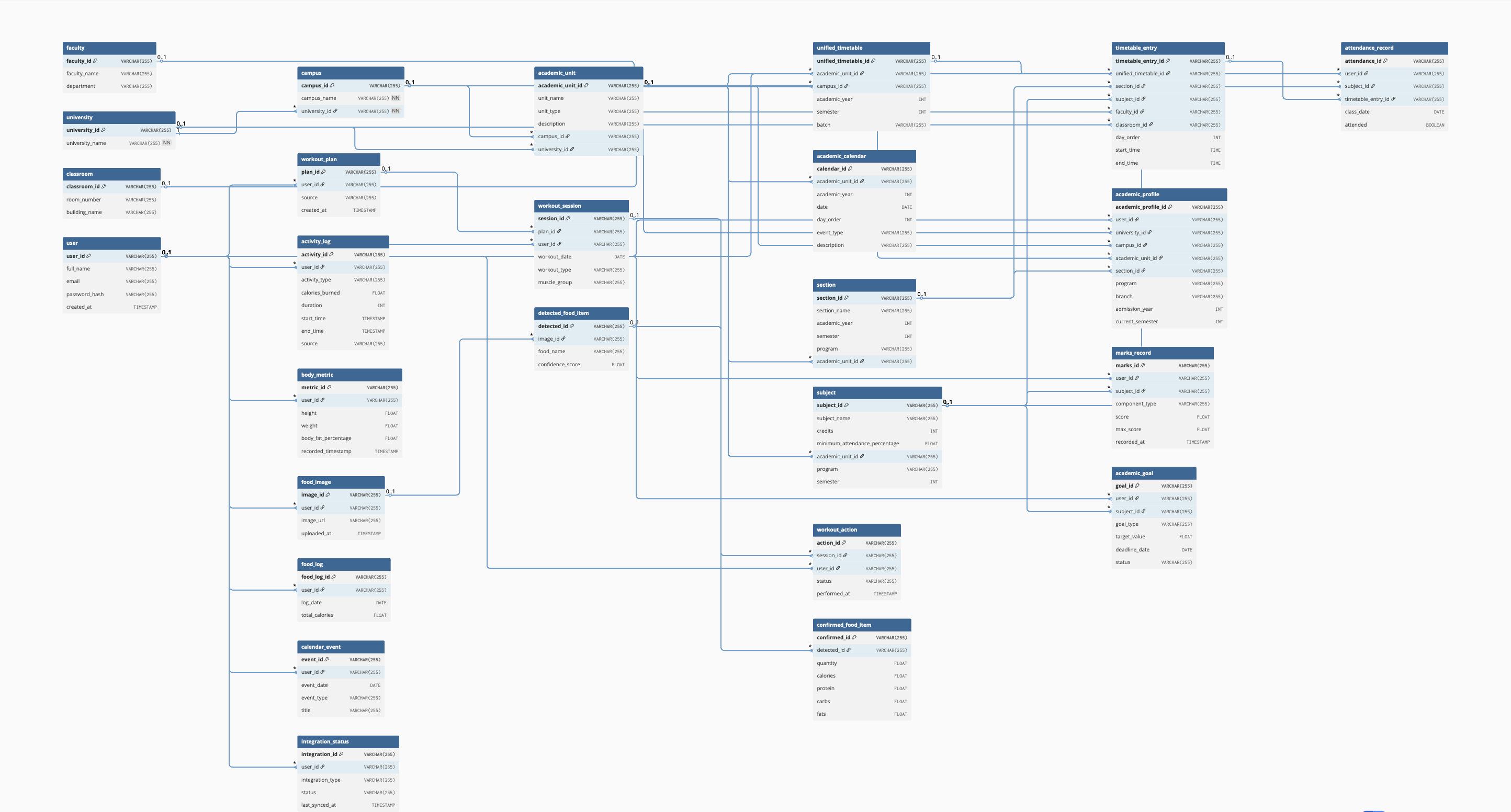


Figure 2.1: Relational Schema for GAK: Integrated Student Academic and Lifestyle system1.

Figure 2.1 depicts the relational schema for the GAK Multi-Campus Decision-Support System.The schema includes institutional lookup tables such as University, Campus, Academic\_Unit, Subject, Section, and Unified\_Timetable to manage shared academic structures efficiently, along with user-centric tables for attendance, marks, fitness, and nutrition data. Central transactional tables like Attendance\_Record, Marks\_Record, Activity\_Log, and Food\_Log store measurable outcomes and reference shared entities through foreign keys. This design achieves high normalization, avoids data redundancy, and supports efficient querying and decision-support analysis across academic and lifestyle domains.

**2.2 Description of Tables**

1.User Relation

User (user\_id, full\_name, email, password\_hash, created\_at)

The User table stores information about students registered in the GAK system. The attribute user\_id acts as the primary key and uniquely identifies each user. Other attributes such as full\_name, email, and password\_hash store authentication and identification details. This table serves as the base entity for all user-specific academic, fitness, and nutrition activities within the system.

2. Academic\_Profile Relation

Academic\_Profile (academic\_profile\_id, user\_id, university\_id, campus\_id, academic\_unit\_id, section\_id, program, branch, admission\_year, current\_semester)

The Academic\_Profile table stores institutional and academic details of users. The attribute academic\_profile\_id is the primary key, while user\_id is a foreign key referencing the User table. This table links users to their university, campus, academic unit, and section, enabling access to shared academic calendars and timetables without data duplication.

3. University Relation

University (university\_id, university\_name)

The University table stores details of universities supported by the system. The attribute university\_id is the primary key. This table enables scalability by allowing multiple universities to be modeled within the same system.

4. Campus Relation

Campus (campus\_id, campus\_name, university\_id)

The Campus table represents different campuses belonging to a university. The attribute campus\_id is the primary key, and university\_id is a foreign key referencing the University table. This structure allows separation of academic data across campuses.

5. Academic\_Unit Relation

Academic\_Unit (academic\_unit\_id, unit\_name, unit\_type, description, campus\_id, university\_id)

The Academic\_Unit table represents academic divisions such as Engineering or Law. The attribute academic\_unit\_id is the primary key. This table enables different academic units to have independent calendars, subjects, and timetables.

6. Academic\_Calendar Relation

Academic\_Calendar (calendar\_id, academic\_unit\_id, academic\_year, date, day\_order, event\_type, description)

The Academic\_Calendar table stores official academic events such as working days, holidays, and examinations. The attribute calendar\_id is the primary key, and academic\_unit\_id is a foreign key. This table acts as a shared institutional calendar for all users within an academic unit.

7. Unified\_Timetable Relation

Unified\_Timetable (unified\_timetable\_id, academic\_unit\_id, campus\_id, academic\_year, semester, batch)

The Unified\_Timetable table stores officially published timetables at the academic-unit level. The attribute unified\_timetable\_id is the primary key. This design avoids duplication by ensuring that timetables are not stored per user.

8. Timetable\_Entry Relation

Timetable\_Entry (timetable\_entry\_id, unified\_timetable\_id, section\_id, subject\_id, faculty\_id, classroom\_id, day\_order, start\_time, end\_time)

The Timetable\_Entry table represents individual class slots. The attribute timetable\_entry\_id is the primary key. Foreign keys link the entry to Unified\_Timetable, Section, Subject, Faculty, and Classroom, defining when and where a class occurs.

9. Subject Relation

Subject (subject\_id, subject\_name, credits, minimum\_attendance\_percentage, academic\_unit\_id, program, semester)

The Subject table stores course-related information. The attribute subject\_id is the primary key. This table defines attendance requirements and academic structure for each subject.

10. Section Relation

Section (section\_id, section\_name, academic\_year, semester, program, academic\_unit\_id)

The Section table represents student groupings within an academic unit. The attribute section\_id is the primary key. Sections link users to appropriate timetables.

11. Faculty Relation

Faculty (faculty\_id, faculty\_name, department)

The Faculty table stores details of instructors. The attribute faculty\_id is the primary key. Faculty members are linked to timetable entries to represent teaching assignments.

12. Classroom Relation

Classroom (classroom\_id, room\_number, building\_name)

The Classroom table stores information about physical class locations. The attribute classroom\_id is the primary key.

13. Attendance\_Record Relation

Attendance\_Record (attendance\_id, user\_id, subject\_id, timetable\_entry\_id, class\_date, attended)

The Attendance\_Record table tracks attendance of users per class session. The attribute attendance\_id is the primary key. Foreign keys link attendance to User, Subject, and Timetable\_Entry. Derived attributes such as attendance\_percentage are computed from this table.

14. Marks\_Record Relation

Marks\_Record (marks\_id, user\_id, subject\_id, component\_type, score, max\_score, recorded\_at)

The Marks\_Record table stores assessment details for users. The attribute marks\_id is the primary key. This table supports academic performance evaluation.

15. Academic\_Goal Relation

Academic\_Goal (goal\_id, user\_id, subject\_id, goal\_type, target\_value, deadline\_date, status)

The Academic\_Goal table stores academic targets set by users. The attribute goal\_id is the primary key. Progress toward goals is derived dynamically.

16. Workout\_Plan Relation

Workout\_Plan (plan\_id, user\_id, source, created\_at)

The Workout\_Plan table stores fitness plans assigned to users. The attribute plan\_id is the primary key.

17. Workout\_Session Relation

Workout\_Session (session\_id, plan\_id, user\_id, workout\_date, workout\_type, muscle\_group)

The Workout\_Session table stores scheduled workout sessions. The attribute session\_id is the primary key.

18. Activity\_Log Relation

Activity\_Log (activity\_id, user\_id, activity\_type, calories\_burned, duration, start\_time, end\_time, source)

The Activity\_Log table records physical activity data. The attribute activity\_id is the primary key.

19. Body\_Metric Relation

Body\_Metric (metric\_id, user\_id, height, weight, body\_fat\_percentage, recorded\_timestamp)

The Body\_Metric table stores health measurements. The attribute metric\_id is the primary key. BMI is a derived attribute.

20. Food\_Image Relation

Food\_Image (image\_id, user\_id, image\_url, uploaded\_at)

The Food\_Image table stores images uploaded by users for food analysis. The attribute image\_id is the primary key.

21. Detected\_Food\_Item Relation

Detected\_Food\_Item (detected\_id, image\_id, food\_name, confidence\_score)

The Detected\_Food\_Item table stores AI-detected food items. The attribute detected\_id is the primary key.

22. Confirmed\_Food\_Item Relation

Confirmed\_Food\_Item (confirmed\_id, detected\_id, quantity, calories, protein, carbs, fats)

The Confirmed\_Food\_Item table stores user-verified food details. The attribute confirmed\_id is the primary key.

23. Food\_Log Relation

Food\_Log (food\_log\_id, user\_id, log\_date, total\_calories)

The Food\_Log table stores daily nutrition summaries. The attribute food\_log\_id is the primary key. total\_calories is a derived, materialized attribute.

24. Calendar\_Event Relation

Calendar\_Event (event\_id, user\_id, event\_date, event\_type, title)

The Calendar\_Event table stores personal user events. The attribute event\_id is the primary key.

### 25. Integration\_Status Relation

Integration\_Status (integration\_id, user\_id, integration\_type, status, last\_synced\_at)

The Integration\_Status table tracks external integrations. The attribute integration\_id is the primary key.

**2.3 Creation of Database and Tables - DDL Commands**

GAK – Gyaan–Karma–Ahara Multi-Campus Decision Support System

DATABASE CREATION:

CREATE DATABASE IF NOT EXISTS GAK\_DB;

USE GAK\_DB;

1. INSTITUTIONAL STRUCTURE TABLE:

UNIVERSITY TABLE :

CREATE TABLE university (

university\_id VARCHAR(50) PRIMARY KEY,

university\_name VARCHAR(255) NOT NULL

);

CAMPUS TABLE :

CREATE TABLE campus (

campus\_id VARCHAR(50) PRIMARY KEY,

campus\_name VARCHAR(255) NOT NULL,

university\_id VARCHAR(50) NOT NULL,

FOREIGN KEY (university\_id) REFERENCES university(university\_id)

);

ACADEMIC UNIT TABLE :

CREATE TABLE academic\_unit (

academic\_unit\_id VARCHAR(50) PRIMARY KEY,

unit\_name VARCHAR(255),

unit\_type VARCHAR(100),

description VARCHAR(255),

campus\_id VARCHAR(50),

university\_id VARCHAR(50),

FOREIGN KEY (campus\_id) REFERENCES campus(campus\_id),

FOREIGN KEY (university\_id) REFERENCES university(university\_id)

);

2. ACADEMIC STRUCTURE TABLES

ACADEMIC CALENDAR TABLE :

CREATE TABLE academic\_calendar (

calendar\_id VARCHAR(50) PRIMARY KEY,

academic\_unit\_id VARCHAR(50),

academic\_year INT,

date DATE,

day\_order INT,

event\_type VARCHAR(100),

description VARCHAR(255),

FOREIGN KEY (academic\_unit\_id) REFERENCES academic\_unit(academic\_unit\_id),

UNIQUE (academic\_unit\_id, academic\_year, date)

);

UNIFIED TIMETABLE TABLE :

CREATE TABLE unified\_timetable (

unified\_timetable\_id VARCHAR(50) PRIMARY KEY,

academic\_unit\_id VARCHAR(50),

campus\_id VARCHAR(50),

academic\_year INT,

semester INT,

batch VARCHAR(50),

FOREIGN KEY (academic\_unit\_id) REFERENCES academic\_unit(academic\_unit\_id),

FOREIGN KEY (campus\_id) REFERENCES campus(campus\_id),

UNIQUE (academic\_unit\_id, academic\_year, semester, batch)

);

FACULTY TABLE :

CREATE TABLE faculty (

faculty\_id VARCHAR(50) PRIMARY KEY,

faculty\_name VARCHAR(255),

department VARCHAR(255)

);

CLASSROOM TABLE :

CREATE TABLE classroom (

classroom\_id VARCHAR(50) PRIMARY KEY,

room\_number VARCHAR(50),

building\_name VARCHAR(255)

);

SUBJECT TABLE :

CREATE TABLE subject (

subject\_id VARCHAR(50) PRIMARY KEY,

subject\_name VARCHAR(255),

credits INT,

minimum\_attendance\_percentage FLOAT,

academic\_unit\_id VARCHAR(50),

semester INT,

FOREIGN KEY (academic\_unit\_id) REFERENCES academic\_unit(academic\_unit\_id),

UNIQUE (academic\_unit\_id, subject\_name, semester)

);

SECTION TABLE :

CREATE TABLE section (

section\_id VARCHAR(50) PRIMARY KEY,

section\_name VARCHAR(100),

academic\_year INT,

semester INT,

program VARCHAR(100),

academic\_unit\_id VARCHAR(50),

FOREIGN KEY (academic\_unit\_id) REFERENCES academic\_unit(academic\_unit\_id),

UNIQUE (academic\_unit\_id, program, academic\_year, semester, section\_name)

);

TIMETABLE ENTRY TABLE :

CREATE TABLE timetable\_entry (

timetable\_entry\_id VARCHAR(50) PRIMARY KEY,

unified\_timetable\_id VARCHAR(50),

section\_id VARCHAR(50),

subject\_id VARCHAR(50),

faculty\_id VARCHAR(50),

classroom\_id VARCHAR(50),

day\_order INT,

start\_time TIME,

end\_time TIME,

FOREIGN KEY (unified\_timetable\_id) REFERENCES unified\_timetable(unified\_timetable\_id),

FOREIGN KEY (section\_id) REFERENCES section(section\_id),

FOREIGN KEY (subject\_id) REFERENCES subject(subject\_id),

FOREIGN KEY (faculty\_id) REFERENCES faculty(faculty\_id),

FOREIGN KEY (classroom\_id) REFERENCES classroom(classroom\_id),

UNIQUE (unified\_timetable\_id, section\_id, day\_order, start\_time)

);

3. USER & ACADEMIC PERFORMANCE TABLES:

USER TABLE :

CREATE TABLE user (

user\_id VARCHAR(50) PRIMARY KEY,

full\_name VARCHAR(255),

email VARCHAR(255),

password\_hash VARCHAR(255),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

ACADEMIC PROFILE TABLE :

CREATE TABLE academic\_profile (

academic\_profile\_id VARCHAR(50) PRIMARY KEY,

user\_id VARCHAR(50) UNIQUE,

university\_id VARCHAR(50),

campus\_id VARCHAR(50),

academic\_unit\_id VARCHAR(50),

section\_id VARCHAR(50),

program VARCHAR(100),

branch VARCHAR(100),

admission\_year INT,

current\_semester INT,

FOREIGN KEY (user\_id) REFERENCES user(user\_id),

FOREIGN KEY (university\_id) REFERENCES university(university\_id),

FOREIGN KEY (campus\_id) REFERENCES campus(campus\_id),

FOREIGN KEY (academic\_unit\_id) REFERENCES academic\_unit(academic\_unit\_id),

FOREIGN KEY (section\_id) REFERENCES section(section\_id)

);

ATTENDANCE RECORD TABLE :

CREATE TABLE attendance\_record (

attendance\_id VARCHAR(50) PRIMARY KEY,

user\_id VARCHAR(50),

subject\_id VARCHAR(50),

timetable\_entry\_id VARCHAR(50),

class\_date DATE,

attended BOOLEAN,

FOREIGN KEY (user\_id) REFERENCES user(user\_id),

FOREIGN KEY (subject\_id) REFERENCES subject(subject\_id),

FOREIGN KEY (timetable\_entry\_id) REFERENCES timetable\_entry(timetable\_entry\_id),

UNIQUE (user\_id, timetable\_entry\_id, class\_date)

);

MARKS RECORD TABLE :

CREATE TABLE marks\_record (

marks\_id VARCHAR(50) PRIMARY KEY,

user\_id VARCHAR(50),

subject\_id VARCHAR(50),

component\_type VARCHAR(100),

score FLOAT,

max\_score FLOAT,

recorded\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES user(user\_id),

FOREIGN KEY (subject\_id) REFERENCES subject(subject\_id),

UNIQUE (user\_id, subject\_id, component\_type)

);

ACADEMIC GOAL TABLE :

CREATE TABLE academic\_goal (

goal\_id VARCHAR(50) PRIMARY KEY,

user\_id VARCHAR(50),

subject\_id VARCHAR(50),

goal\_type VARCHAR(100),

target\_value FLOAT,

deadline\_date DATE,

status VARCHAR(100),

FOREIGN KEY (user\_id) REFERENCES user(user\_id),

FOREIGN KEY (subject\_id) REFERENCES subject(subject\_id)

);

4. FITNESS (KARMA) TABLES

CREATE TABLE workout\_plan (

plan\_id VARCHAR(50) PRIMARY KEY,

user\_id VARCHAR(50),

source VARCHAR(100),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES user(user\_id)

);

CREATE TABLE workout\_session (

session\_id VARCHAR(50) PRIMARY KEY,

plan\_id VARCHAR(50),

user\_id VARCHAR(50),

workout\_date DATE,

workout\_type VARCHAR(100),

muscle\_group VARCHAR(100),

FOREIGN KEY (plan\_id) REFERENCES workout\_plan(plan\_id),

FOREIGN KEY (user\_id) REFERENCES user(user\_id),

UNIQUE (plan\_id, workout\_date)

);

CREATE TABLE workout\_action (

action\_id VARCHAR(50) PRIMARY KEY,

session\_id VARCHAR(50),

user\_id VARCHAR(50),

status VARCHAR(100),

performed\_at TIMESTAMP,

FOREIGN KEY (session\_id) REFERENCES workout\_session(session\_id),

FOREIGN KEY (user\_id) REFERENCES user(user\_id)

);

CREATE TABLE activity\_log (

activity\_id VARCHAR(50) PRIMARY KEY,

user\_id VARCHAR(50),

activity\_type VARCHAR(100),

calories\_burned FLOAT,

duration INT,

start\_time TIMESTAMP,

end\_time TIMESTAMP,

source VARCHAR(100),

FOREIGN KEY (user\_id) REFERENCES user(user\_id)

);

CREATE TABLE body\_metric (

metric\_id VARCHAR(50) PRIMARY KEY,

user\_id VARCHAR(50),

height FLOAT,

weight FLOAT,

body\_fat\_percentage FLOAT,

recorded\_timestamp TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES user(user\_id),

UNIQUE (user\_id, recorded\_timestamp)

);

5. NUTRITION (AHARA) TABLES

CREATE TABLE food\_image (

image\_id VARCHAR(50) PRIMARY KEY,

user\_id VARCHAR(50),

image\_url VARCHAR(255),

uploaded\_at TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES user(user\_id)

);

CREATE TABLE detected\_food\_item (

detected\_id VARCHAR(50) PRIMARY KEY,

image\_id VARCHAR(50),

food\_name VARCHAR(255),

confidence\_score FLOAT,

FOREIGN KEY (image\_id) REFERENCES food\_image(image\_id),

UNIQUE (image\_id, food\_name)

);

CREATE TABLE confirmed\_food\_item (

confirmed\_id VARCHAR(50) PRIMARY KEY,

detected\_id VARCHAR(50) UNIQUE,

quantity FLOAT,

calories FLOAT,

protein FLOAT,

carbs FLOAT,

fats FLOAT,

FOREIGN KEY (detected\_id) REFERENCES detected\_food\_item(detected\_id)

);

CREATE TABLE food\_log (

food\_log\_id VARCHAR(50) PRIMARY KEY,

user\_id VARCHAR(50),

log\_date DATE,

total\_calories FLOAT,

FOREIGN KEY (user\_id) REFERENCES user(user\_id),

UNIQUE (user\_id, log\_date)

);

6. SYSTEM TABLES

CREATE TABLE calendar\_event (

event\_id VARCHAR(50) PRIMARY KEY,

user\_id VARCHAR(50),

event\_date DATE,

event\_type VARCHAR(100),

title VARCHAR(255),

FOREIGN KEY (user\_id) REFERENCES user(user\_id)

);

CREATE TABLE integration\_status (

integration\_id VARCHAR(50) PRIMARY KEY,

user\_id VARCHAR(50),

integration\_type VARCHAR(100),

status VARCHAR(100),

last\_synced\_at TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES user(user\_id),

UNIQUE (user\_id, integration\_type)

);

**2.4 Insertion of tuples into the table – DML commands**

**Insert SQL (for all tables)**

INSERT INTO university (university\_id, university\_name) VALUES

('UNI01', 'SRM Institute of Science and Technology');

+---------------+-------------------------------------------+

| university\_id | university\_name |

+---------------+-------------------------------------------+

| UNI01 | SRM Institute of Science and Technology |

+---------------+-------------------------------------------+

INSERT INTO campus (campus\_id, campus\_name, university\_id) VALUES

('CAMP01', 'Kattankulathur', 'UNI01');

+-----------+-------------------+---------------+

| campus\_id | campus\_name | university\_id |

+-----------+-------------------+---------------+

| CAMP01 | Kattankulathur | UNI01 |

+-----------+-------------------+---------------+

INSERT INTO academic\_unit (academic\_unit\_id, unit\_name, unit\_type, description, campus\_id, university\_id) VALUES

('AU01', 'Engineering', 'School', 'Engineering and Technology', 'CAMP01', 'UNI01');

+------------------+-------------+-----------+---------------------------+-----------+---------------+

| academic\_unit\_id | unit\_name | unit\_type | description | campus\_id | university\_id |

+------------------+-------------+-----------+---------------------------+-----------+---------------+

| AU01 | Engineering | School | Engineering and Technology| CAMP01 | UNI01 |

+------------------+-------------+-----------+---------------------------+-----------+---------------+

INSERT INTO subject (subject\_id, subject\_name, credits, minimum\_attendance\_percentage, academic\_unit\_id, program, semester) VALUES

('SUB01', 'Database Management Systems', 4, 75, 'AU01', 'B.Tech CSE', 4),

('SUB02', 'Operating Systems', 3, 75, 'AU01', 'B.Tech CSE', 4);

+------------+------------------------------+---------+-------------------------------+------------------+-------------+----------+

| subject\_id | subject\_name | credits | minimum\_attendance\_percentage | academic\_unit\_id | program | semester |

+------------+------------------------------+---------+-------------------------------+------------------+-------------+----------+

| SUB01 | Database Management Systems | 4 | 75 | AU01 | B.Tech CSE | 4 |

| SUB02 | Operating Systems | 3 | 75 | AU01 | B.Tech CSE | 4 |

+------------+------------------------------+---------+-------------------------------+------------------+-------------+----------+

INSERT INTO section (section\_id, section\_name, academic\_year, semester, program, academic\_unit\_id) VALUES

('SEC01', 'CSE-A', 2026, 4, 'B.Tech CSE', 'AU01');

+------------+--------------+---------------+----------+-------------+------------------+

| section\_id | section\_name | academic\_year | semester | program | academic\_unit\_id |

+------------+--------------+---------------+----------+-------------+------------------+

| SEC01 | CSE-A | 2026 | 4 | B.Tech CSE | AU01 |

+------------+--------------+---------------+----------+-------------+------------------+

INSERT INTO faculty (faculty\_id, faculty\_name, department) VALUES

('FAC01', 'Dr. Kumar', 'CSE Department');

+------------+-------------+----------------+

| faculty\_id | faculty\_name| department |

+------------+-------------+----------------+

| FAC01 | Dr. Kumar | CSE Department |

+------------+-------------+----------------+

INSERT INTO classroom (classroom\_id, room\_number, building\_name) VALUES

('CR01', '301', 'Tech Block');

+--------------+------------+---------------+

| classroom\_id | room\_number| building\_name |

+--------------+------------+---------------+

| CR01 | 301 | Tech Block |

+--------------+------------+---------------+

INSERT INTO unified\_timetable (unified\_timetable\_id, academic\_unit\_id, campus\_id, academic\_year, semester, batch) VALUES

('UT01', 'AU01', 'CAMP01', 2026, 4, 'Batch1');

+--------------------+------------------+-----------+---------------+----------+--------+

| unified\_timetable\_id| academic\_unit\_id | campus\_id | academic\_year | semester | batch |

+---------------------+------------------+-----------+---------------+----------+--------+

| UT01 | AU01 | CAMP01 | 2026 | 4 | Batch1 |

+---------------------+------------------+-----------+---------------+----------+--------+

INSERT INTO timetable\_entry (timetable\_entry\_id, unified\_timetable\_id, section\_id, subject\_id, faculty\_id, classroom\_id, day\_order, start\_time, end\_time) VALUES

('TE01', 'UT01', 'SEC01', 'SUB01', 'FAC01', 'CR01', 1, '09:00:00', '10:00:00');

+-------------------+---------------------+------------+------------+------------+--------------+----------+------------+----------+

| timetable\_entry\_id| unified\_timetable\_id| section\_id | subject\_id | faculty\_id | classroom\_id | day\_order| start\_time | end\_time |

+-------------------+---------------------+------------+------------+------------+--------------+----------+------------+----------+

| TE01 | UT01 | SEC01 | SUB01 | FAC01 | CR01 | 1 | 09:00:00 | 10:00:00 |

+-------------------+---------------------+------------+------------+------------+--------------+----------+------------+----------+

INSERT INTO user (user\_id, full\_name, email, password\_hash, created\_at) VALUES

('U01', 'Rahul Sharma', 'rahul@srm.edu', 'hash123', NOW());

+---------+---------------+------------------+--------------+---------------------+

| user\_id | full\_name | email | password\_hash| created\_at |

+---------+---------------+------------------+--------------+---------------------+

| U01 | Rahul Sharma | rahul@srm.edu | hash123 | 2026-02-10 10:00:00 |

+---------+---------------+------------------+--------------+---------------------+

INSERT INTO academic\_profile (academic\_profile\_id, user\_id, university\_id, campus\_id, academic\_unit\_id, section\_id, program, branch, admission\_year, current\_semester) VALUES

('AP01', 'U01', 'UNI01', 'CAMP01', 'AU01', 'SEC01', 'B.Tech', 'CSE', 2023, 4);

+--------------------+--------+---------------+-----------+------------------+------------+---------+--------+---------------+------------------+

| academic\_profile\_id| user\_id | university\_id | campus\_id | academic\_unit\_id | section\_id | program | branch | admission\_year| current\_semester |

+--------------------+---------+---------------+-----------+------------------+------------+---------+--------+---------------+------------------+

| AP01 | U01 | UNI01 | CAMP01 | AU01 | SEC01 | B.Tech | CSE | 2023 | 4 |

+--------------------+---------+---------------+-----------+------------------+------------+---------+--------+---------------+------------------+

INSERT INTO attendance\_record (attendance\_id, user\_id, subject\_id, timetable\_entry\_id, class\_date, attended) VALUES

('ATT01', 'U01', 'SUB01', 'TE01', '2026-02-01', TRUE);

+--------------+---------+------------+-------------------+------------+----------+

| attendance\_id| user\_id | subject\_id | timetable\_entry\_id| class\_date | attended |

+--------------+---------+------------+-------------------+------------+----------+

| ATT01 | U01 | SUB01 | TE01 | 2026-02-01 | TRUE |

+--------------+---------+------------+-------------------+------------+----------+

INSERT INTO marks\_record (marks\_id, user\_id, subject\_id, component\_type, score, max\_score, recorded\_at) VALUES

('MR01', 'U01', 'SUB01', 'Internal', 42, 50, NOW());

+----------+---------+------------+----------------+-------+-----------+---------------------+

| marks\_id | user\_id | subject\_id | component\_type | score | max\_score | recorded\_at |

+----------+---------+------------+----------------+-------+-----------+---------------------+

| MR01 | U01 | SUB01 | Internal | 42 | 50 | 2026-02-10 10:00:00 |

+----------+---------+------------+----------------+-------+-----------+---------------------+

INSERT INTO academic\_goal (goal\_id, user\_id, subject\_id, goal\_type, target\_value, deadline\_date, status) VALUES

('G01', 'U01', 'SUB01', 'Marks', 90, '2026-04-30', 'Active');

+---------+--------+------------+----------+-------------+--------------+--------+

| goal\_id | user\_id | subject\_id | goal\_type| target\_value| deadline\_date| status |

+---------+---------+------------+----------+-------------+--------------+--------+

| G01 | U01 | SUB01 | Marks | 90 | 2026-04-30 | Active |

+---------+---------+------------+----------+-------------+--------------+--------+

INSERT INTO workout\_plan (plan\_id, user\_id, source, created\_at) VALUES

('WP01', 'U01', 'AI', NOW());

+---------+---------+--------+---------------------+

| plan\_id | user\_id | source | created\_at |

+---------+---------+--------+---------------------+

| WP01 | U01 | AI | 2026-02-10 10:00:00 |

+---------+---------+--------+---------------------+

INSERT INTO workout\_session (session\_id, plan\_id, user\_id, workout\_date, workout\_type, muscle\_group) VALUES

('WS01', 'WP01', 'U01', '2026-02-10', 'Cardio', 'Full Body');

+------------+---------+---------+-------------+-------------+--------------+

| session\_id | plan\_id | user\_id | workout\_date| workout\_type| muscle\_group |

+------------+---------+---------+-------------+-------------+--------------+

| WS01 | WP01 | U01 | 2026-02-10 | Cardio | Full Body |

+------------+---------+---------+-------------+-------------+--------------+

INSERT INTO workout\_action (action\_id, session\_id, user\_id, status, performed\_at) VALUES

('WA01', 'WS01', 'U01', 'Completed', NOW());

+-----------+------------+---------+-----------+---------------------+

| action\_id | session\_id | user\_id | status | performed\_at |

+-----------+------------+---------+-----------+---------------------+

| WA01 | WS01 | U01 | Completed | 2026-02-10 10:00:00 |

+-----------+------------+---------+-----------+---------------------+

INSERT INTO activity\_log (activity\_id, user\_id, activity\_type, calories\_burned, duration, start\_time, end\_time, source) VALUES

('ACT01', 'U01', 'Running', 300, 30, NOW(), NOW(), 'Manual');

+------------+---------+---------------+----------------+----------+---------------------+---------------------+--------+

| activity\_id| user\_id | activity\_type | calories\_burned| duration | start\_time | end\_time | source |

+------------+---------+---------------+----------------+----------+---------------------+---------------------+--------+

| ACT01 | U01 | Running | 300 | 30 | 2026-02-10 10:00:00 | 2026-02-10 10:00:00 | Manual |

+------------+---------+---------------+----------------+----------+---------------------+---------------------+--------+

INSERT INTO body\_metric (metric\_id, user\_id, height, weight, body\_fat\_percentage, recorded\_timestamp) VALUES

('BM01', 'U01', 1.75, 70, 18, NOW());

+----------+---------+--------+--------+----------------------+---------------------+

| metric\_id| user\_id | height | weight | body\_fat\_percentage | recorded\_timestamp |

+----------+---------+--------+--------+----------------------+---------------------+

| BM01 | U01 | 1.75 | 70 | 18 | 2026-02-10 10:00:00 |

+----------+---------+--------+--------+----------------------+---------------------+

INSERT INTO food\_image (image\_id, user\_id, image\_url, uploaded\_at) VALUES

('IMG01', 'U01', 'meal.jpg', NOW());

+----------+---------+-----------+---------------------+

| image\_id | user\_id | image\_url | uploaded\_at |

+----------+---------+-----------+---------------------+

| IMG01 | U01 | meal.jpg | 2026-02-10 10:00:00 |

+----------+---------+-----------+---------------------+

INSERT INTO detected\_food\_item (detected\_id, image\_id, food\_name, confidence\_score) VALUES

('DF01', 'IMG01', 'Rice', 0.95);

+-------------+----------+-----------+------------------+

| detected\_id | image\_id | food\_name | confidence\_score |

+-------------+----------+-----------+------------------+

| DF01 | IMG01 | Rice | 0.95 |

+-------------+----------+-----------+------------------+

INSERT INTO confirmed\_food\_item (confirmed\_id, detected\_id, quantity, calories, protein, carbs, fats) VALUES

('CF01', 'DF01', 2, 200, 4, 45, 1);

+--------------+------------+----------+----------+---------+-------+------+

| confirmed\_id | detected\_id| quantity | calories | protein | carbs | fats |

+--------------+------------+----------+----------+---------+-------+------+

| CF01 | DF01 | 2 | 200 | 4 | 45 | 1 |

+--------------+------------+----------+----------+---------+-------+------+

INSERT INTO food\_log (food\_log\_id, user\_id, log\_date, total\_calories) VALUES

('FL01', 'U01', '2026-02-10', 400);

+-------------+---------+------------+---------------+

| food\_log\_id | user\_id | log\_date | total\_calories|

+-------------+---------+------------+---------------+

| FL01 | U01 | 2026-02-10 | 400 |

+-------------+---------+------------+---------------+

INSERT INTO calendar\_event (event\_id, user\_id, event\_date, event\_type, title) VALUES

('EV01', 'U01', '2026-02-15', 'Personal', 'Exam Preparation');

+----------+---------+------------+------------+-------------------+

| event\_id | user\_id | event\_date | event\_type | title |

+----------+---------+------------+------------+-------------------+

| EV01 | U01 | 2026-02-15 | Personal | Exam Preparation |

+----------+---------+------------+------------+-------------------+

INSERT INTO integration\_status (integration\_id, user\_id, integration\_type, status, last\_synced\_at) VALUES

('INT01', 'U01', 'Google Fit', 'Connected', NOW());

+---------------+---------+----------------+-----------+---------------------+

| integration\_id| user\_id | integration\_type| status | last\_synced\_at |

+---------------+---------+----------------+-----------+---------------------+

| INT01 | U01 | Google Fit | Connected | 2026-02-10 10:00:00 |

+---------------+---------+----------------+-----------+---------------------+

**CHAPTER 3**

**Complex queries based on the concepts of constraints, sets, joins, views, Triggers and Cursors.**

**(min 3 questions with queries for all topics)**

**< create complex queries based on the sub-topic with questions and SQL statement with output>**

* 1. **Adding Constraints and queries based on constraints**

**Question:**

**SQL Statement:**

**Output:**

* 1. **Queries based on Aggregate Functions**

**Question:**

**SQL Statement:**

**Output:**

* 1. **Complex queries based on Sets**

**Question:**

**SQL Statement:**

**Output:**

* 1. **Complex queries based on Subqueries**

**Question:**

**SQL Statement:**

**Output:**

* 1. **Complex queries based on Joins**

**Question:**

**SQL Statement:**

**Output:**

* 1. **Complex queries based on views**

**Question:**

**SQL Statement:**

**Output:**

* 1. **Complex queries based on Triggers**

**Question:**

**SQL Statement:**

**Output:**

* 1. **Complex queries based on Cursors**

**Question:**

**SQL Statement:**

**Output:**

**CHAPTER 4**

**ANALYZING THE PITFALLS, IDENTIFYING THE DEPENDENCIES, AND APPLYING NORMALIZATIONS**

**(show the table before and after respective normalization)**

**4.1 Analyse the Pitfalls in Relations**

**4.2 First Normal Form**

**4.2.1: Identify Dependency**

**4.2.2: Apply Normalization to 1NF**

**4.3 Second Normal Form**

**4.3.1: Identify Dependency**

**4.3.2: Apply Normalization to 2NF**

**4.4 Third Normal Form**

**4.4.1: Identify Dependency**

**4.4.2: Apply Normalization to 3NF**

**4.5 BCNF**

**4.5.1: Identify Dependency**

**4.5.2: Apply Normalization to BCNF**

**4.6 Fourth Normal Form**

**4.6.1: Identify Dependency**

**4.6.2: Apply Normalization to 4NF**

**4.7 Fifth Normal Form**

**4.7.1: Identify Dependency**

**4.7.2: Apply Normalization to 5NF**

**CHAPTER 5**

**IMPLEMENTATION OF CONCURRENCY CONTROL AND RECOVERY MECHANISMS**

**5.1 Introduction to Transactions**

**5.1.1 Properties**

**5.1.2 States**

**5.2 Transaction Control Language (TCL)**

**5.2.1 Save point**

**5.2.2 Commit**

**5.2.3 Rollback**

**5.3 create 5 transactions for your project and execute**

**5.3.1 Transaction1 code with output (min 5 transactions)**

**Use savepoints, commit, rollback**

**< Sample:**

**-- Step 1: Start transaction by updating book availability**

UPDATE Books

SET available\_copies = available\_copies – 1 WHERE book\_id = 101;

**-- Step 2: Set a savepoint after first change**

SAVEPOINT after\_first\_update;

**-- Step 3: Another update**

UPDATE Books SET available\_copies = available\_copies – 1 WHERE book\_id = 102;

**-- Step 4: Oops! Something's wrong with second update, rollback to savepoint**

ROLLBACK TO after\_first\_update;

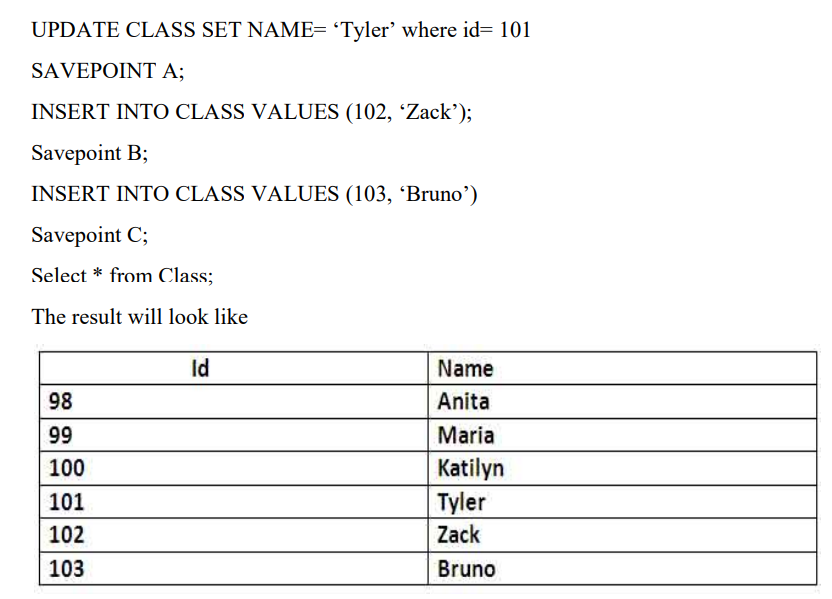
**-- Step 5: Continue with a safe update**

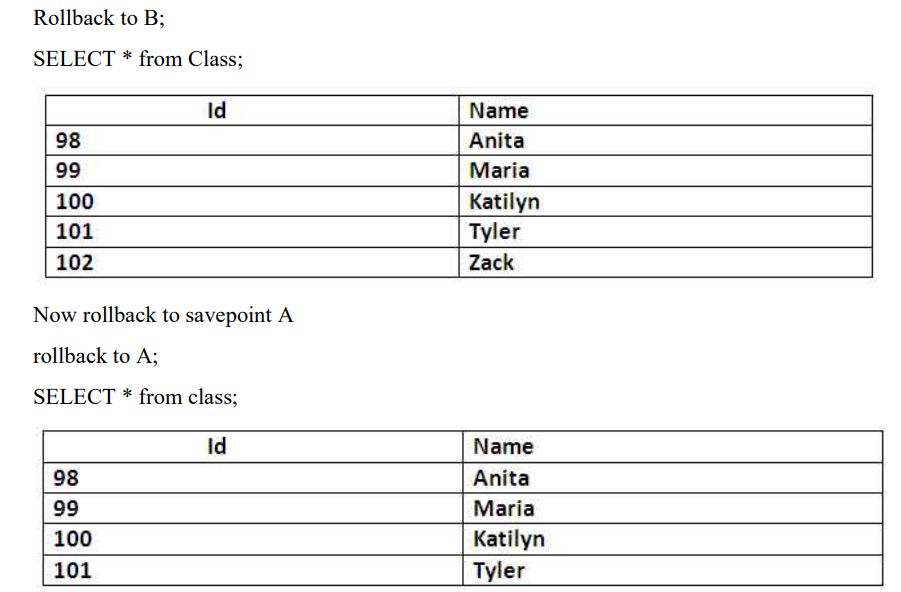
UPDATE Books SET available\_copies = available\_copies + 1 WHERE book\_id = 101;

**-- Step 6: Commit all changes since savepoint**

COMMIT; >

**Sample 2:**

****

****

**5.3 Concurrency control**

**5.3.1 Concurrency control Algorithms**

**5.3.1 Locking commands**

**a. Row-Level Locking – SELECT ... FOR UPDATE**

**b. Table-Level Locking – LOCK TABLE**

Lock Modes

| **Lock Mode** | **Description** |
| --- | --- |
| ROW SHARE | Allows concurrent access; prevents other sessions from locking the table exclusively. |
| ROW EXCLUSIVE | Prevents other sessions from locking in share mode. Used by default for DML. |
| SHARE | Allows queries but not updates or deletes. |
| SHARE ROW EXCLUSIVE | A mix; more restrictive than SHARE. |
| EXCLUSIVE | Prevents all other access — full table lock. |

**c. COMMIT – Release All Locks**

**d. ROLLBACK – Undo Changes & Release Locks**

**5.3.2 Example (1 example for your project)**

**CHAPTER 6**

**FRONT-END AND BACK-END CODE OF <PROJECT TITLE>**

**6.1 Front–End Module codes**

**6.2 Database connectivity**

**CHAPTER 7**

**RESULTS AND DISCUSSIONS**

7.1 Screenshots of front-end

7.2 Screenshots of Database

**References**