

FORT BEND Independent School District

Database Management System



[Fort Bend School Database Design]

Group No:

| Student Name | Student ID |
|--------------------|------------|
| Sabrina Mobassirin | 2280077 |
| Florine Nswal | 1901623 |
| MD. Yousuf | 2231282 |

Table of Contents

| | |
|---|-----------|
| PART I: INTRODUCTION | 4 |
| 1.1 ABSTRACT | 4 |
| 1.2 MISSION STATEMENT | 4 |
| 1.3 MISSION OBJECTIVE | 4 |
| PART II: ANALYSIS | 5 |
| 2 PROBLEM DEFINITION AND DATA REQUIREMENTS | 5 |
| 2.1 PROBLEM DESCRIPTION | 5 |
| 2.2 DATA REQUIREMENTS | 5 |
| 2.3 WORKING RULES | 6 |
| 2.4 INTENDED OUTPUT OF THE SYSTEM. | 8 |
| PART III: DB DESIGN | 9 |
| 3 ER DIAGRAM DESIGN | 9 |
| 3.1 ER DIAGRAM | 9 |
| 4 ER-TO-LOGICAL SCHEMA MAPPING | 10 |
| 4.1 MAPPING OF REGULAR ENTITY TYPES | 10 |
| 4.2 MAPPING OF WEAK ENTITY TYPES | 13 |
| 4.3 MAPPING OF BINARY 1-1 RELATIONSHIP TYPES | 13 |
| 4.4 MAPPING OF BINARY 1-N RELATIONSHIP TYPES | 14 |
| 4.5 MAPPING OF BINARY M-N RELATIONSHIP TYPES | 16 |
| 4.6 MAPPING OF MULTIVALUED ATTRIBUTES | 17 |
| 4.7 MAPPING OF N-ARRAY RELATIONSHIP TYPES | 17 |
| 4.8 SCHEMA DIAGRAM | 17 |
| 5 NORMALIZATION | 19 |
| 5.1 FIRST NORMAL FORM | 19 |
| 5.2 SECOND NORMAL FORM | 19 |
| 5.3 THIRD NORMAL FORM | 19 |
| 6 FINAL DB SCHEMA DIAGRAM | 20 |
| PART IV: IMPLEMENTATION | 21 |
| 7 TABLE CREATION SCRIPT | 21 |
| 7.1 <EMPLOYEE> TABLE | 21 |
| 7.2 <EMP_PHONE> TABLE | 21 |
| 7.3 <TEACHER> TABLE | 21 |
| 7.4 <IT_WORKER> TABLE | 22 |
| 7.5 <CLEANING_STAFF> TABLE | 22 |
| 7.6 <LIBRARIAN> TABLE | 22 |
| 7.7 <LIB_MANAGERS> TABLE | 22 |
| 7.8 <LIBRARY> TABLE | 23 |
| 7.9 <BOOKS> TABLE | 23 |
| 7.10 <MEMBERS> TABLE | 23 |
| 7.11 <STUDENTS> TABLE | 23 |
| 7.12 <COURSES> TABLE | 24 |
| 7.13 <ENROLLMENT> TABLE | 24 |
| 7.14 <LEGAL_GUARDIAN> TABLE | 24 |
| 7.15 <LG_INFO> TABLE | 24 |
| 7.16 <LG_PHONES> TABLE | 25 |
| 7.17 <GRADES> TABLE | 25 |

| | |
|---|-----------|
| 7.18 < <i>Exams</i> > TABLE | 25 |
| 7.19 < <i>Rooms</i> > TABLE | 25 |
| 7.20 < <i>Class</i> > TABLE | 26 |
| 8 USE CASE LIST | 26 |
| 9 USE CASE IMPLEMENTATION | 32 |
| 10 QUERIES TESTING | 38 |
| 10.1 < <i>YEAR_WORKS LIBRARY MEMBERS</i> > | 38 |
| 10.2 < <i>COVID-19 STUDENTS</i> > | 39 |
| 10.3 < <i>NON-PHYSICS 1 TEACHERS</i> > | 40 |
| 10.4 < <i>ALL FAILED STUDENTS IN 2021</i> > | 40 |
| 10.5 < <i>FUNDAMENTAL YEAR STUDENTS</i> > | 41 |
| 11 CONCLUSION | 42 |
| 12 REFERENCES | 42 |
| APPENDIX | 42 |

PART I: Introduction

1.1 Abstract

FORT BEND Independent School District is one of the most highly regarded school districts in Texas. Every school under this district must keep track of their employees, students, courses and classes capacity. We have decided to build a database system for the school district to facilitate access to search, add, and modify records containing all information. The database system will monitor entities such as employees, students, guardians, courses and books. In addition, it will allow the user to update or remove information about students or employees, add new courses or books from the database, etc.

1.2 Mission Statement

The mission is to provide a robust and secure database management system that enables school districts to manage and organize student records, employee information, teacher's information, school performance, Legal Guardian information and Library information efficiently. A platform that allows educators to access, update, and analyze data quickly and accurately, empowering them to make informed decisions that benefit students and the community.

1.3 Mission Objective

To maintain (enter, update, and delete) data on employees.

To maintain (enter, update, and delete) data on students.

To maintain (enter, update, and delete) data on guardians.

To maintain (enter, update, and delete) data on courses.

To maintain (enter, update, and delete) data on books.

To perform searches on employees.

To perform searches on students.

To perform searches on guardians.

To perform searches on courses.

To perform searches on books.

To track the status of employees.

To track the status of students.

To track the status of guardians .

To track the status of courses.

To track the status of books.

To report on employees.

To report on students.

To report on guardians.

To report on courses.

To report on books.

PART II: Analysis

2 Problem Definition and Data Requirements

2.1 Problem Description

Dealing with a massive amount of data requires a lot of organization and to be able to do that you need help from a computer. Dumping all data in files and some excel sheets is inefficient and slow. Furthermore, most schools are having a lot of issues dealing with this massive data. However, schools need management software to keep all members' data such as students, all employees including teachers, IT-Workers, and even cleaning staff members. In addition to that, we need to manage the courses and which student took which course and when plus the grades he got. So, we have decided to solve this problem by creating a full database system that includes all the massive data that any school needs.

2.2 Data Requirements

❖ Employees Entity:

- EmployeeID: a unique ID for every employee.
- Name: a composite attribute that consists of the first name (Fname), middle name (Mname), and last name (Lname).
- Sex: the gender of the employee.
- Salary: the salary of the employee.
- EMP_Phone: The phone number of the employee.
- BDate: the birth date of the employee.

Note: Employee is a superclass with 4 subclasses: Teacher, Cleaning staff,

IT-Worker and Librarian.

❖ Teacher Entity:

- Specialization: The Specialization of the teacher.

❖ IT-Worker:

- Specialization: The Specialization of the IT-Worker.

❖ Students Entity:

- StudSSN: a unique ID for every student.
- Name: a composite attribute that consists of the first name (Fname), middle name (Mname), and last name (Lname).
- BDate: the birth date of the student.
- STU_Phone: The phone number of the student.

❖ Class entity:

- Class_ID: a unique id for every class.
- Period: the time that takes by each class.
- Start_time: the start time of the class.

❖ Courses Entity:

- Course_ID: a unique ID for every course.
- Course_Name: the name of the course.
- prerequisite: the course required for registration in a new course.

❖ Enrollment Entity:

- Quarter: the name of the quarter.
- Year: the year of the enrollment.

❖ Exams Entity:

- Exam_ID: a unique ID for every exam.

- *Exam_Mark*: the mark of the exam.
- *Exam_Date*: the date of the exam.
- *Exam_Time*: the time of the exam.

❖ **Grades Entity:**

- *Grade_ID*: a unique ID for every grade.
- *Attend*: the attendance grade.
- *Behaviour*: the behaviour grade.
- *Final_Mark*: the final mark for each course, which is a derived attribute.

❖ **Legal Guardian Entity:**

- *Name*: the name of the Guardian.
- *Bdate*: The Birth date of the Guardian.
- *Relationship*: the relationship between Student and his guardian.
- *Sex*: the gender of the guardian
- *LG_Phone*: Legal Guardian's Phone number.

Note: the Legal Guardian Entity is a weak entity.

❖ **Rooms Entity:**

- *Room_No*: a unique ID for every room.
- *Floor_NO*: the number of the floor.
- *Capacity*: the capacity for each room.

❖ **Library Entity:**

- *Lib_ID*: a unique ID for every library.
- *Lib_Name*: the name of the library.

❖ **Members Entity:**

- *Mem_ID*: a unique ID for every Library members.
- *Mem_Name*: name of the member.

❖ **Books Entity:**

- *Book_ID*: a unique ID for every book.
- *Book_Name*: the name of the book.
- *Author*: the author of the book

2.3 Working Rules

A school job is to hire employees on the Employees tables, accept students from the Student table, and store their data and manipulate it. The school offers courses for the students that are stored in the Courses table, the courses are taught by the employees. In addition, a school has an Id, name and address, every school has its own Employees, students, and library although they teach the same courses.

Employee

The employees are the people who are going to run and manage the school. The school has different categories of employees such as teachers, supervisors, working staff, every one of them is important to get the school going. Managers will be responsible for the teachers and student affairs; the teachers will oversee subjects to teach the students. The IT workers will oversee managing the data of students and employees, the cleaning staff must keep the school clean, and finally there are the librarians who will oversee the library.

Employee/Teacher -

A teacher [GIVES] a class based on his specialty.

Employee/Cleaning_staff -

All cleaning staff are responsible for cleaning the school.

Employee/IT_Worker -

Each IT worker is responsible for technical things going in school based on his specialty such as database admin, data entry, data analysis.

Employee/Librarian -

Each library has one library manager. Using **Emp_ID** as FK

Library -

The students can go to the library which is managed by the librarian employee to have access to books that can help them, whether in their courses or just for fun.

Class -

A class is the session where teachers teach courses and it does have **Teacher_ID** as a foreign key.

Students -

First, each student must provide their data and their parents info, after their acceptance in the school, a **Stud_ID** is automatically generated.

Legal guardian -

The table which stores parents or people who are responsible for each student.

Courses -

The school defines what courses are taught by which professor. The courses have exams that determine whether or not the students are going to pass the course. The course will be given a classroom with a specific time that does not conflict with other courses.

Enrollment -

It is the table where all semester's data is stored such as, enrollment year, and the status of passing. It has **Stud_ID**, **Course_ID** as a foreign key.

Grades -

Includes all marks for **Stud_SSN** such as, behavior, attendance, year works.

2.4 Intended Output of the system.

Output & Queries:

- Display all employees with their jobs type.
- Display all students with their courses.
- Display weekly courses schedule.
- Display all students with their dependence.
- Display report for a specific Student.
- Search for a Student by ID.
- Search for available courses students can enroll in.
- Search for a teacher by ID.
- Display grades of specific students.
- etc...

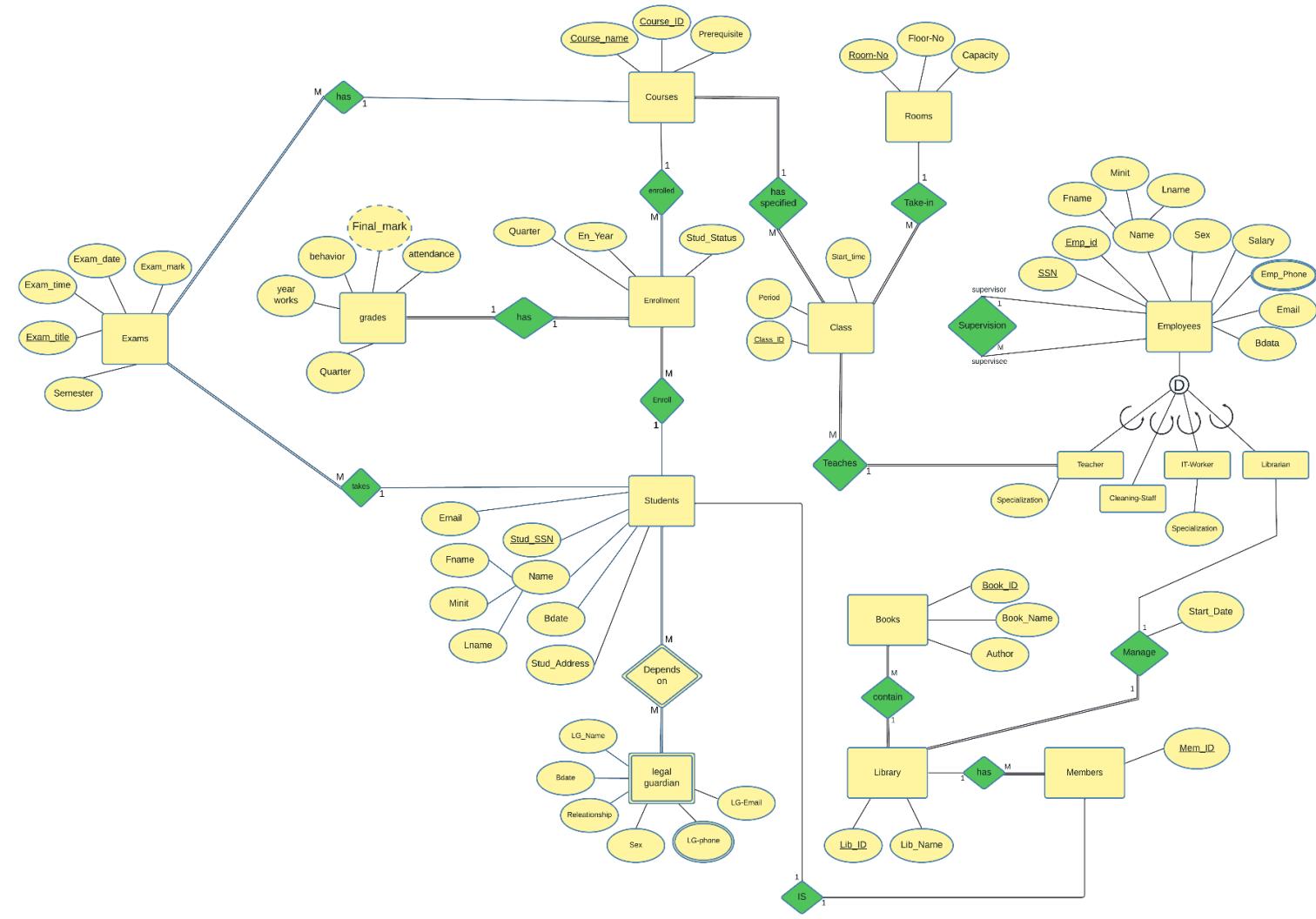
Transactions:

- Insert a new student.
- Update supervisor of employees.
- Assign a teacher to a specified course.
- delete a teacher.
- Assign room to specified course.

PART III: DB DESIGN

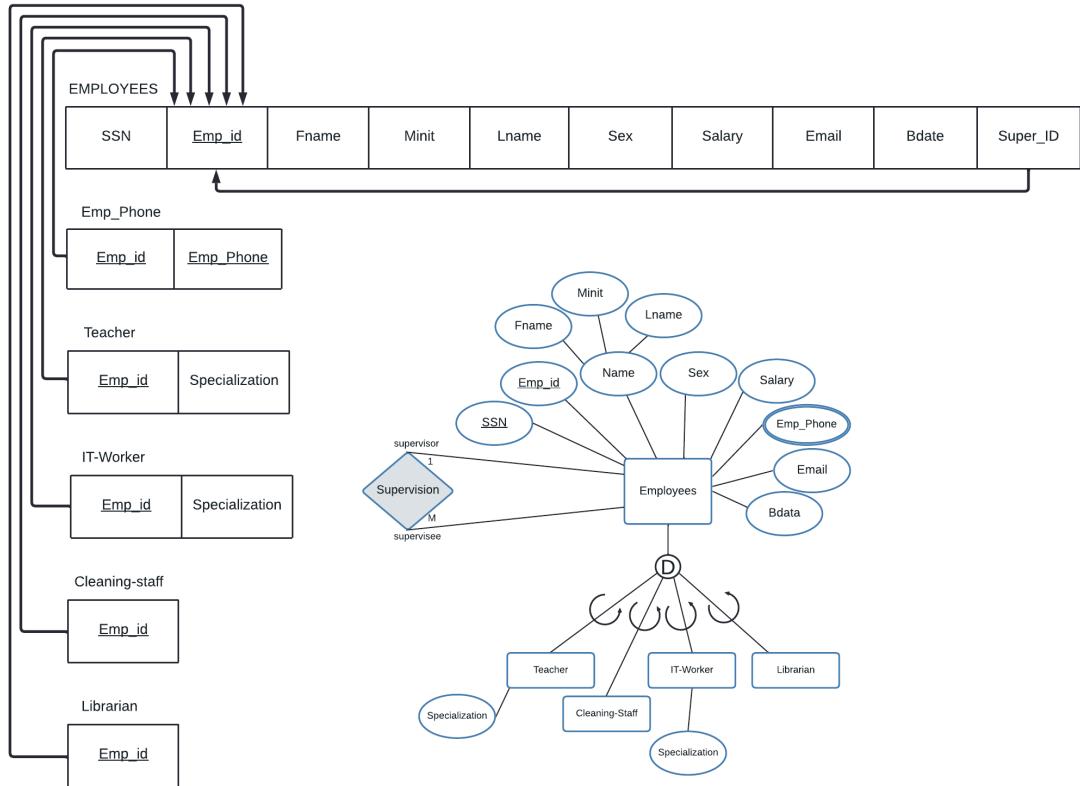
3 ER Diagram Design

3.1 ER diagram



4 ER-to-logical schema mapping

4.1 Mapping of Regular Entity Types

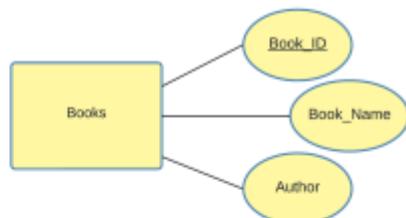


Class

| Teacher_ID | Course_ID | Semester | C_Period | Room_No | Start_time |
|------------|-----------|----------|----------|---------|------------|
|------------|-----------|----------|----------|---------|------------|

Books

| Book_ID | Book_Name | Author | Lib_ID |
|---------|-----------|--------|--------|
| | | | |



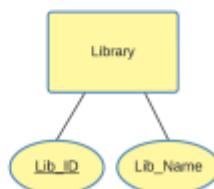
Members

| Mem_ID | Lib_ID |
|--------|--------|
| | |



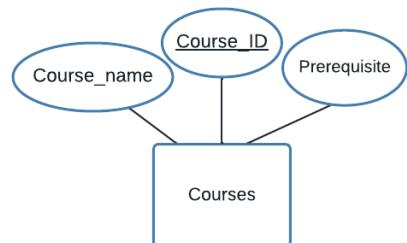
Library

| Lib_ID | Lib_Name |
|--------|----------|
| | |



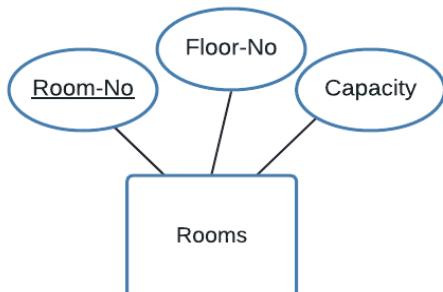
Courses

| Course_ID | Course_name | Prerequisite |
|-----------|-------------|--------------|
| | | |



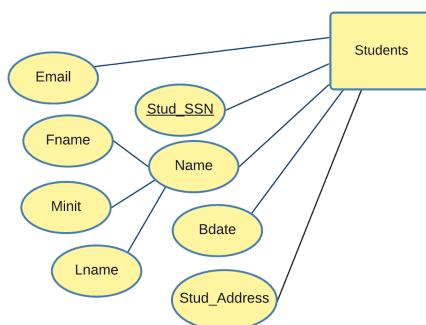
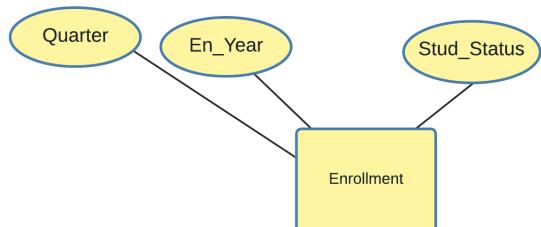
Rooms

| Room-No | Floor-No | Capacity |
|---------|----------|----------|
| | | |



enrollment

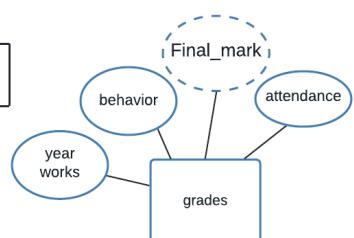
| Stud_SSN | Course_ID | En_year | Stud_status | Quarter |
|----------|-----------|---------|-------------|---------|
|----------|-----------|---------|-------------|---------|



| Stud_SSN | Fname | Minit | Lname | Bdate | Stud_Address | Email |
|----------|-------|-------|-------|-------|--------------|-------|
|----------|-------|-------|-------|-------|--------------|-------|

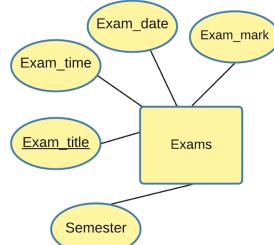
Grades

| Stud_SSN | Course_ID | year works | behavior | attendance |
|----------|-----------|------------|----------|------------|
|----------|-----------|------------|----------|------------|



Exams

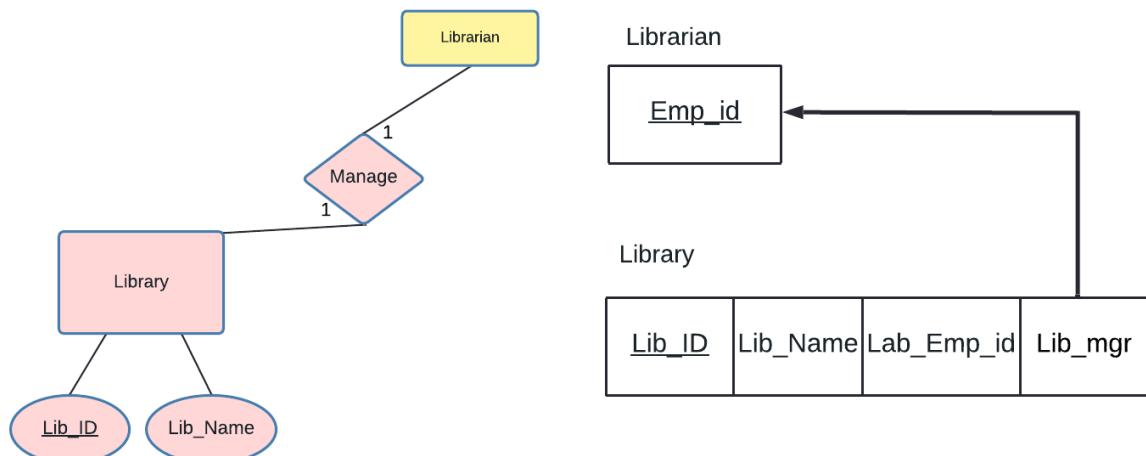
| Exam_title | Stud_SSN | Course_ID | Semester | Exam_date | Exam_mark | exam_time |
|------------|----------|-----------|----------|-----------|-----------|-----------|
|------------|----------|-----------|----------|-----------|-----------|-----------|

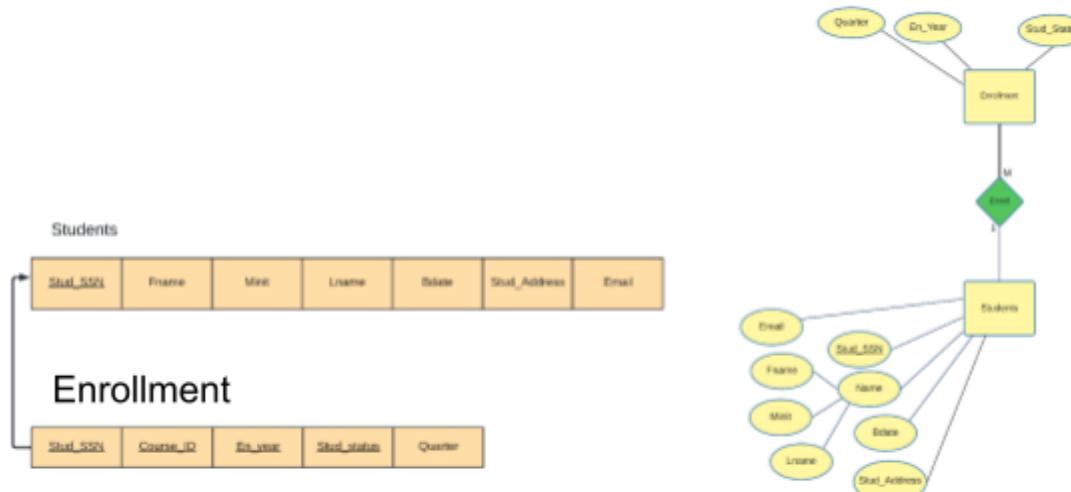
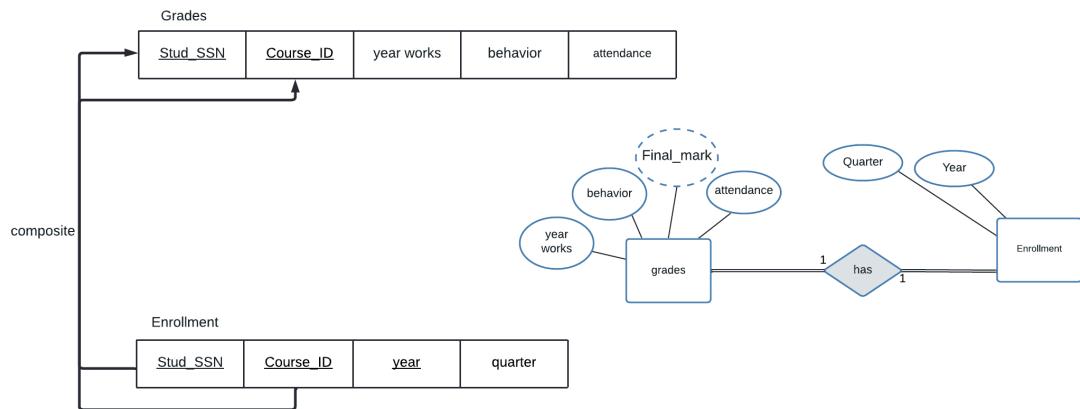


4.2 Mapping of Weak Entity Types

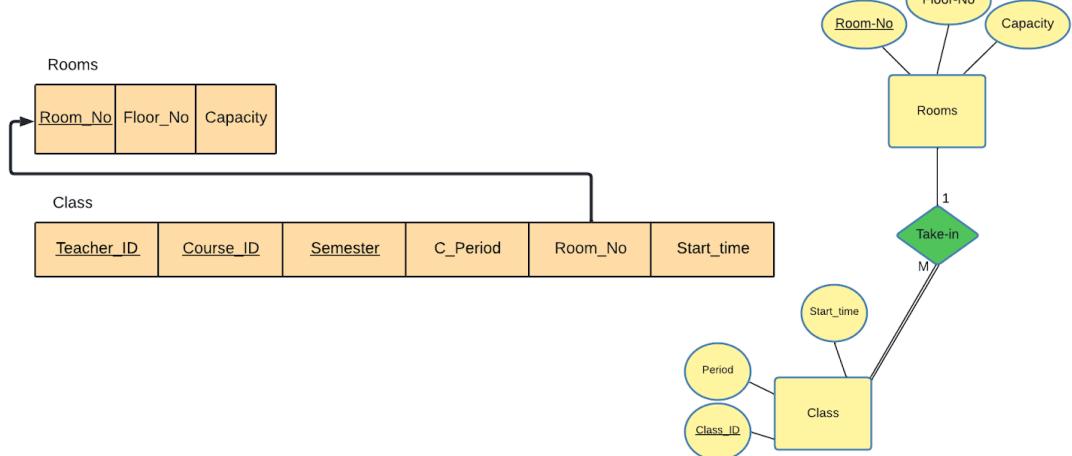


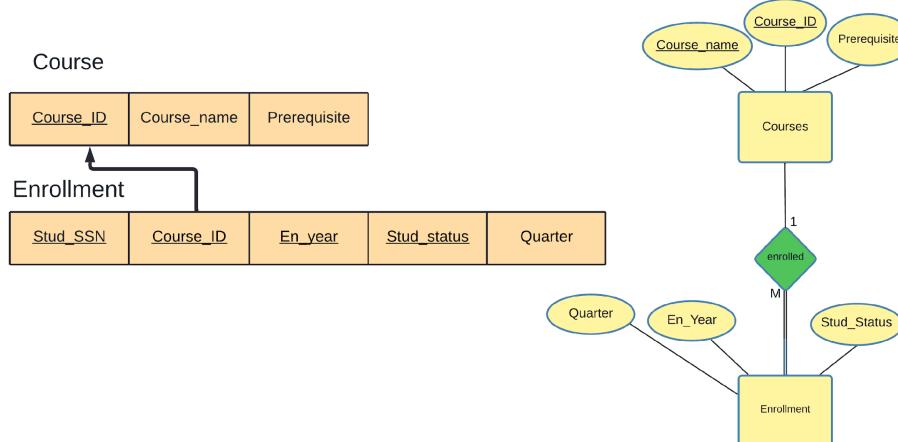
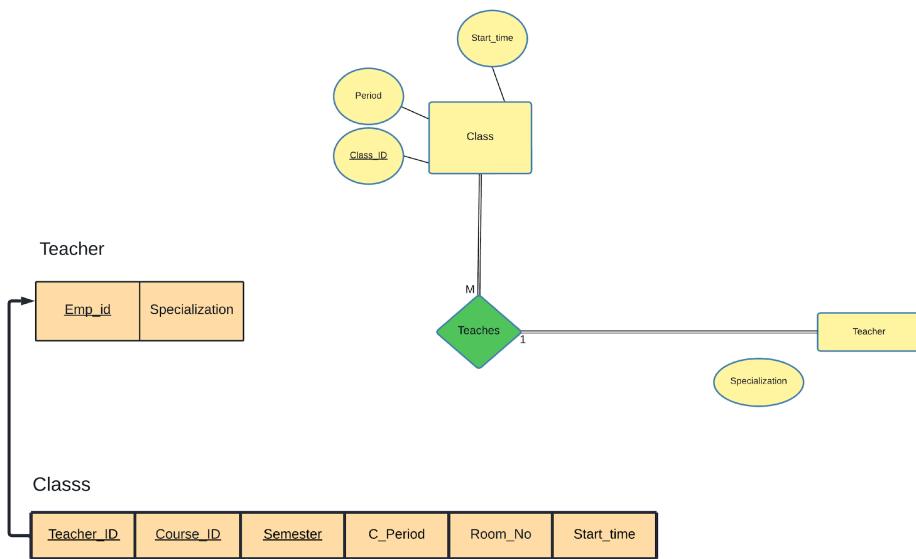
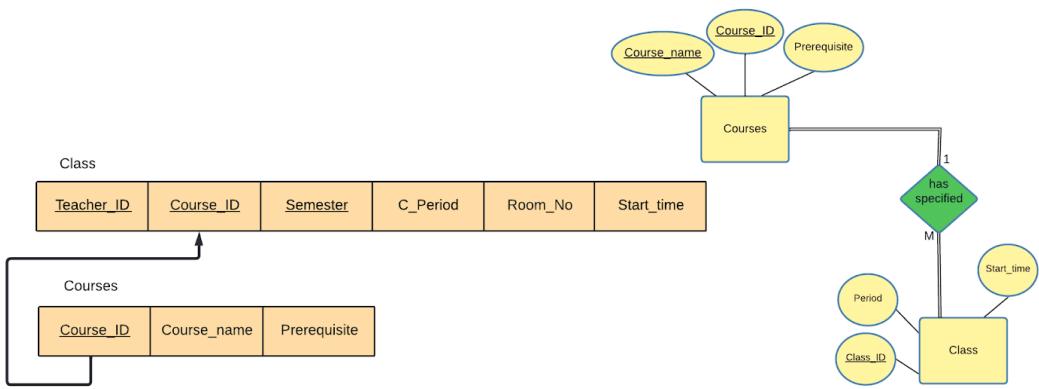
4.3 Mapping of binary 1-1 relationship types

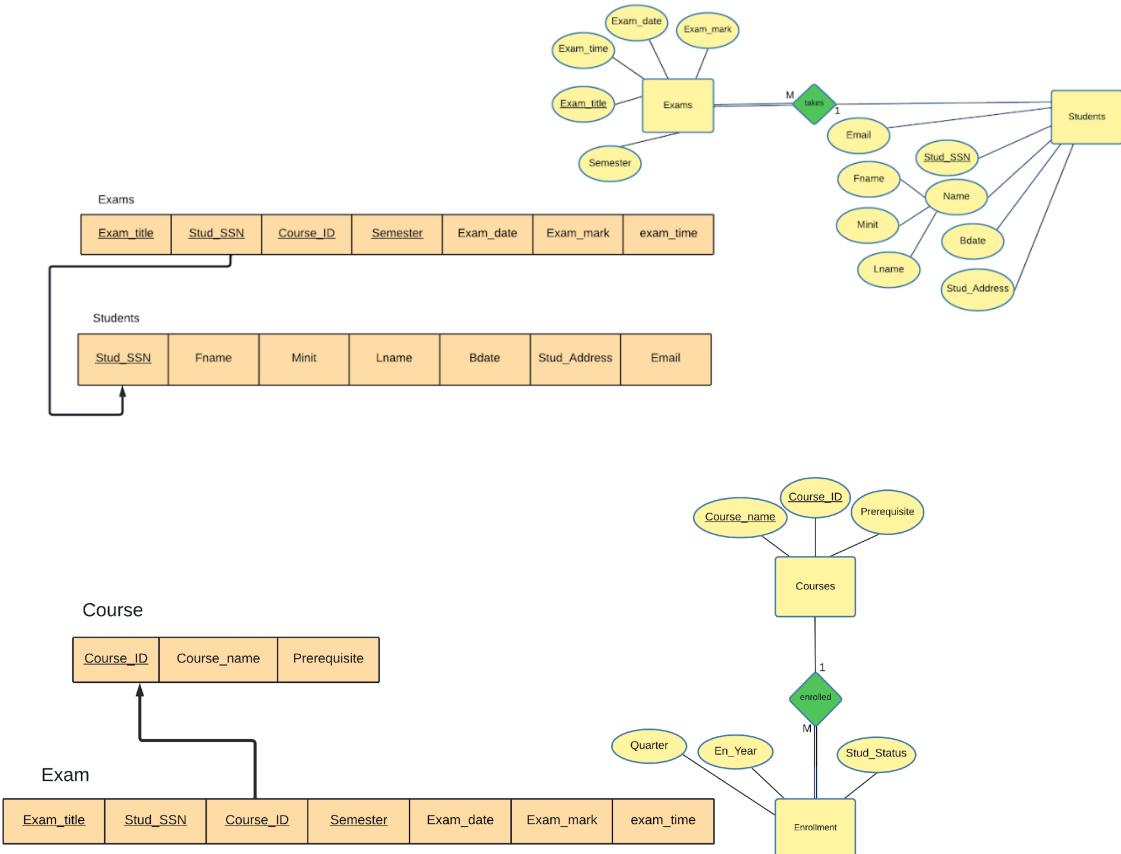
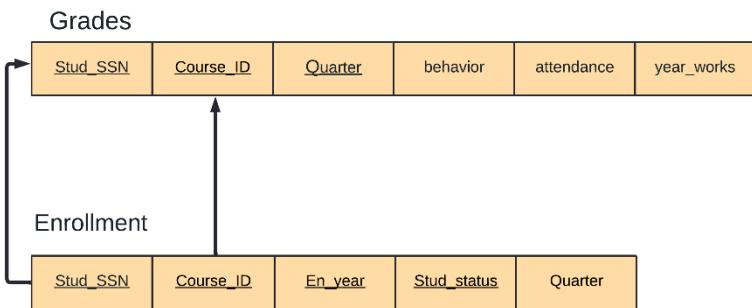
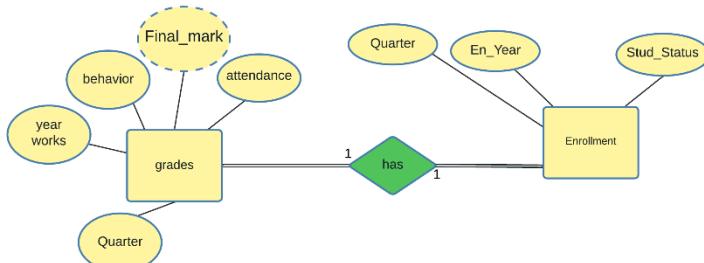




4.4 Mapping of binary 1-N relationship types

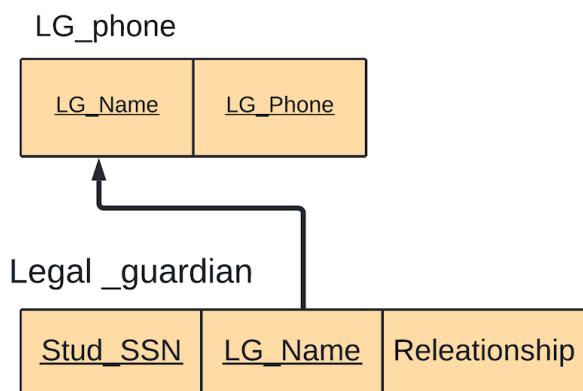
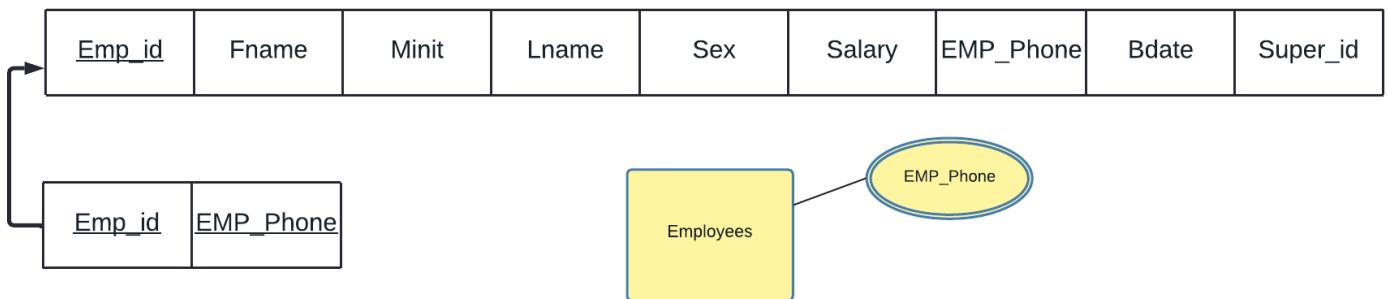






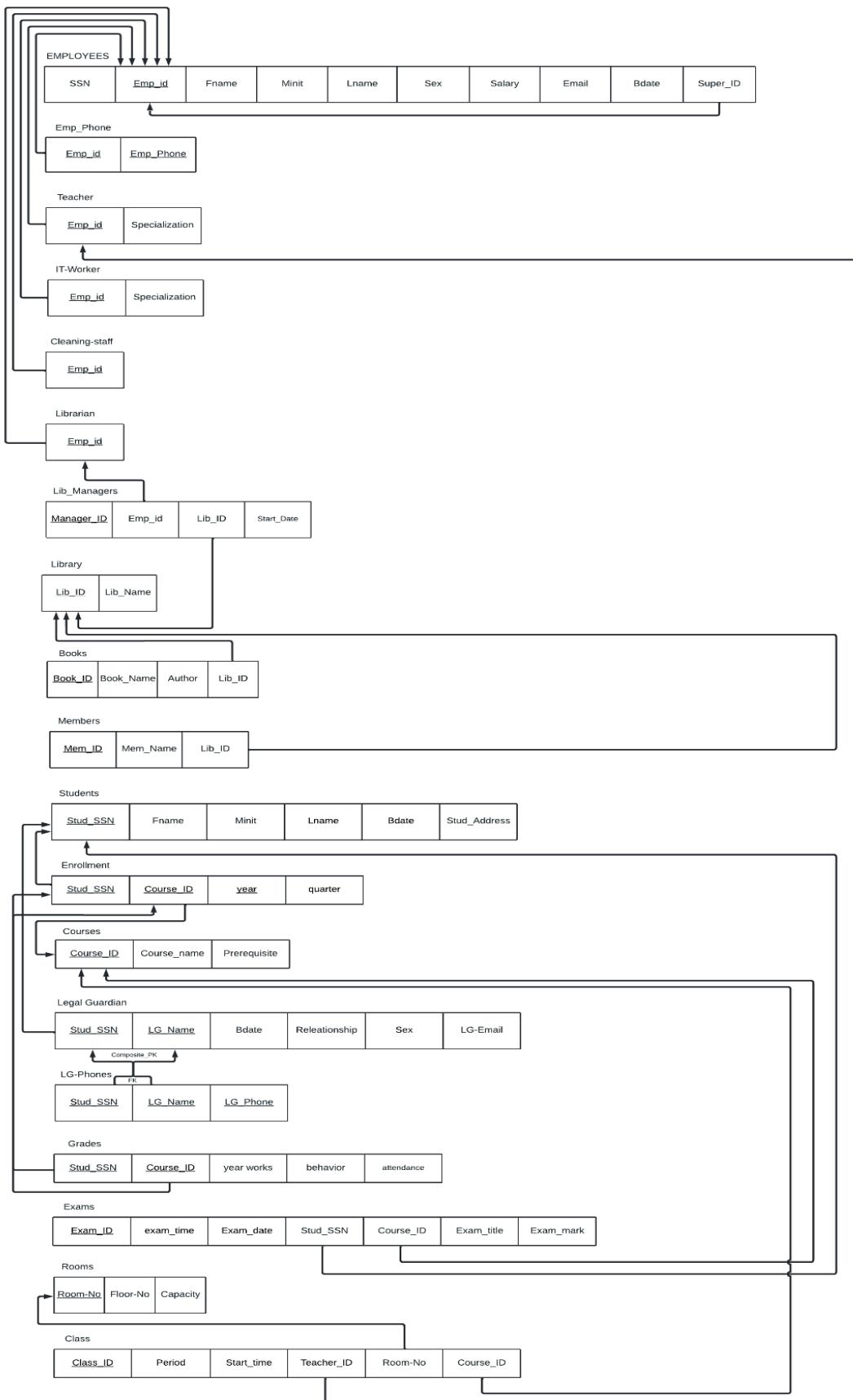
4.5 Mapping of binary M-N relationship types [NONE]

4.6 Mapping of multivalued attributes



4.7 Mapping of n-array relationship types [NONE]

4.8 Schema Diagram



5 Normalization

5.1 First Normal Form

First normal form requires not to have a composite and multivalued attributes. We have two multivalued attributes in the employee entity and in the Legal guardian entity. Since we transformed the mentioned attributes to a relation, we no longer have anything that goes against the guidelines for the first normalization Form.

| EMPLOYEES | | | | | | | | | | | | | | |
|----------------|---------|-------|----------|---------------|-----|--------|----------|-------|-----------|--|--|--|--|--|
| SSN | Emp_id | Fname | Minit | Lname | Sex | Salary | Email | Bdate | Super_ID | | | | | |
| Emp_Phone | | | | | | | | | | | | | | |
| Emp_id | | | | | | | | | Emp_Phone | | | | | |
| Legal Guardian | | | | | | | | | | | | | | |
| Stud_SSN | LG_Name | | Bdate | Releationship | | Sex | LG-Email | | | | | | | |
| LG-Phones | | | | | | | | | | | | | | |
| Stud_SSN | LG_Name | | LG_Phone | | | | | | | | | | | |

5.2 Second Normal Form

To make our relational schema in Second normal form, every non-prime attribute should be fully functionally dependent on the primary key. All our relations are in the 2NF except Legal Guardian Relation:

| Legal Guardian | | | | | |
|----------------|---------|-------|---------------|-----|----------|
| Stud_SSN | LG_Name | Bdate | Releationship | Sex | LG-Email |
| | | | | | |

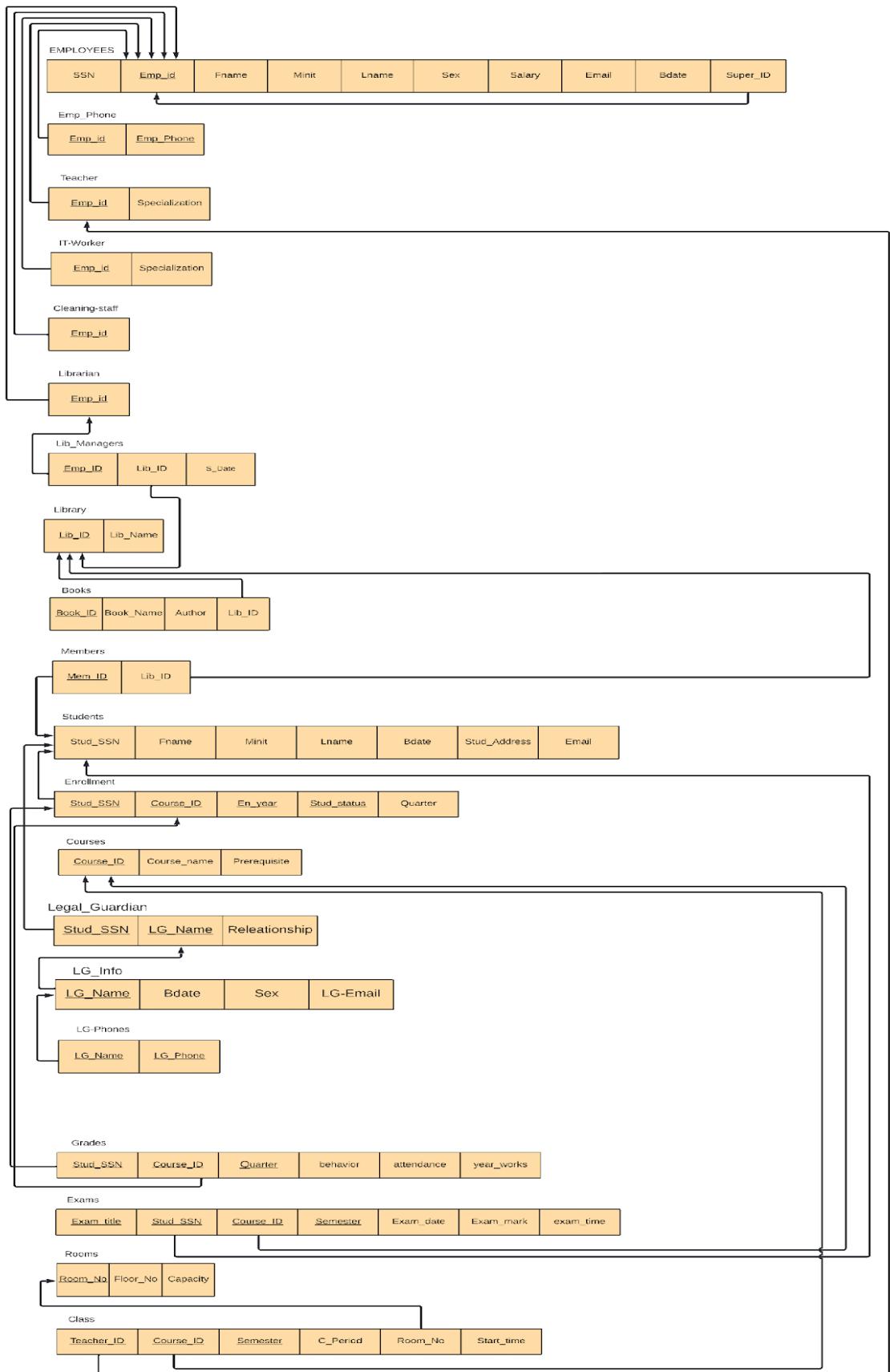
| Legal Guardian | | |
|----------------|---------|---------------|
| Stud_SSN | LG_Name | Releationship |
| | | |

| LG_Info | | | |
|---------|-------|-----|----------|
| LG_Name | Bdate | Sex | LG-Email |
| | | | |

5.3 Third Normal Form

[NONE]

6 Final DB Schema Diagram



PART III: IMPLEMENTATION

7 Table Creation Script

7.1 <Employee> TABLE

```
⊖ CREATE TABLE Employee(
    `SSN` BIGINT,
    `Emp_ID` BIGINT,
    `Fname` VARCHAR(15),
    `Minit` VARCHAR(15),
    `Lname` VARCHAR(15) NOT NULL,
    `Sex` char,
    `Salary` DECIMAL,
    `Email` VARCHAR(30),
    `Bdata` DATE,
    `Super_ID` BIGINT,
    CONSTRAINT Employee_Pk PRIMARY KEY (Emp_ID),
    CONSTRAINT Employee_Employee_Fk FOREIGN KEY (Super_ID) REFERENCES
        Employee(Emp_ID)
);
```

7.2 <Emp_Phone> TABLE

```
⊖ CREATE TABLE Emp_phone(
    `Emp_ID` BIGINT,
    `Emp_Phone` VARCHAR(15),
    PRIMARY KEY (Emp_ID, Emp_Phone),
    CONSTRAINT Emp_Phone_Fk FOREIGN KEY (Emp_ID) REFERENCES Employee(Emp_ID)
);
```

7.3 <Teacher> TABLE

```
⊖ CREATE TABLE Teacher (
    `Emp_ID` BIGINT,
    `Specialization` varchar(15),
    PRIMARY KEY (Emp_ID),
    CONSTRAINT Teacher_Fk FOREIGN KEY (Emp_ID) REFERENCES Employee(Emp_ID)
);
```

7.4 <IT_Worker> TABLE

```
⊖ create table IT_Worker(
    `Emp_ID` BIGINT,
    `Specialization` varchar(15),
    CONSTRAINT IT_Worker_PK PRIMARY KEY (Emp_ID),
    CONSTRAINT IT_worker_Fk FOREIGN KEY(Emp_ID) REFERENCES Employee(Emp_ID)
);
```

7.5 <Cleaning_Staff> TABLE

```
⊖ CREATE TABLE Cleaning_Staff (
    `Emp_ID` BIGINT,
    CONSTRAINT Cleaning_Staff_PK PRIMARY KEY (Emp_ID),
    CONSTRAINT Cleaning_Staff_FK FOREIGN KEY (Emp_ID) REFERENCES Employee(Emp_ID)
);
```

7.6 <librarian> TABLE

```
⊖ create table librarian (
    Emp_ID BIGINT ,
    CONSTRAINT librarian_PK PRIMARY KEY (Emp_ID),
    CONSTRAINT librarian_Fk FOREIGN KEY (Emp_ID) REFERENCES Employee (Emp_ID)
);
```

7.7 <lib_managers> TABLE

```
⊖ create table lib_managers (
    Emp_ID BIGINT UNIQUE,
    Lib_ID BIGINT UNIQUE,
    Strt_Date DATE,
    CONSTRAINT lib_managers_PK PRIMARY KEY (Emp_ID,Lib_ID),
    CONSTRAINT lib_managers_FK1 FOREIGN KEY(Emp_ID) REFERENCES librarian(Emp_ID),
    CONSTRAINT lib_managers_FK2 FOREIGN KEY(Lib_ID) REFERENCES Library(Lib_ID)
);
```

7.8 <Library> TABLE

```
CREATE TABLE Library(
    Lib_ID BIGINT,
    Lib_Name VARCHAR(15),
    CONSTRAINT Library_PK PRIMARY KEY (Lib_ID)
);
```

7.9 <Books> TABLE

```
CREATE TABLE Books(
    Book_ID BIGINT,
    Book_Name VARCHAR(30),
    Author varchar(15),
    Lib_ID BIGINT,
    CONSTRAINT Books_PK PRIMARY KEY (Book_ID),
    CONSTRAINT Books_FK FOREIGN KEY(Lib_ID) REFERENCES Library(Lib_ID)
);
```

7.10 <Members> TABLE

```
CREATE TABLE Members(
    Memb_id BIGINT,
    Lib_ID BIGINT,
    CONSTRAINT Members_PK PRIMARY KEY (Memb_id),
    CONSTRAINT Members_FK FOREIGN KEY(Lib_ID) REFERENCES Library(Lib_ID),
    CONSTRAINT Members_FK2 FOREIGN KEY(Memb_id) REFERENCES Students(Stud_SSN)
);
```

7.11 <Students> TABLE

```
CREATE TABLE Students(
    Stud_SSN BIGINT,
    Fname VARCHAR(15),
    Minit VARCHAR(15),
    Lname varchar(15),
    Bdata DATE,
    lib_managers,
    Stud_Address varchar(30) NULL,
    Email VARCHAR(25),
    CONSTRAINT Students_PK PRIMARY KEY (Stud_SSN)
);
```

7.12 <courses> TABLE

```
create table courses(
    Course_ID BIGINT,
    Course_Name varchar(15),
    Prerequisite varchar(15),
    CONSTRAINT courses_PK PRIMARY KEY (Course_ID)
);
```

7.13 <Enrollment> TABLE

```
CREATE TABLE Enrollment(
    Stud_SSN BIGINT,
    Course_ID BIGINT,
    En_year varchar(15),
    Stud_Status varchar(15),
    Quarter VARCHAR(15),
    CONSTRAINT Enrollment_PK PRIMARY KEY (Stud_SSN , Course_ID , En_year , Stud_Status),
    CONSTRAINT Enrollment_FK1 FOREIGN KEY(Stud_SSN) REFERENCES Students(Stud_SSN),
    CONSTRAINT Enrollment_FK2 FOREIGN KEY(Course_ID) REFERENCES courses(Course_ID)
);
```

7.14 <Legal_Guardian> TABLE

```
create table Legal_Guardian(
    Stud_SSN BIGINT UNIQUE ,
    LG_Name varchar(15) UNIQUE,
    Relationship varchar(15),
    CONSTRAINT Legal_Guardian_PK PRIMARY KEY (Stud_SSN,LG_Name),
    CONSTRAINT Legal_Guardian_FK FOREIGN KEY(Stud_SSN) REFERENCES Students(Stud_SSN)
);
```

7.15 <LG_Info> TABLE

```
create table LG_Info (
    LG_Name varchar(15),
    Bdata DATE,
    Sex CHAR,
    LG_Email VARCHAR(25),
    CONSTRAINT LG_Info_PK PRIMARY KEY (LG_Name),
    CONSTRAINT LG_Info_FK FOREIGN KEY(LG_Name) REFERENCES Legal_Guardian(LG_Name)
);
```

7.16 <LG_Phones> TABLE

```
⊖ create table LG_Phones(
    LG_Name VARCHAR(15),
    LG_Phone BIGINT,
    CONSTRAINT LG_Phones_PK PRIMARY KEY (LG_Name,LG_Phone),
    CONSTRAINT LG_Phones_FK FOREIGN KEY(LG_Name) REFERENCES LG_Info(LG_Name)
);
```

7.17 <Grades> TABLE

```
DROP TABLE IF EXISTS grades;
⊖ CREATE TABLE grades (
    Stud_SSN BIGINT,
    Course_ID BIGINT,
    Quarter VARCHAR(15),
    Behavior BIGINT,
    Attendance BIGINT,
    Year_Works BIGINT,
    final_mark INT GENERATED ALWAYS AS ((Behavior/10) + (Attendance/10) + Year_Works)
    VIRTUAL,
    PRIMARY KEY (Stud_SSN, Course_ID, Quarter),
    CONSTRAINT Grades_FK1 FOREIGN KEY (Stud_SSN) REFERENCES Students(Stud_SSN),
    CONSTRAINT Grades_FK2 FOREIGN KEY (Course_ID) REFERENCES courses(Course_ID)
);
```

7.18 <Exams> TABLE

```
⊖ create table Exams(
    Exam_Title VARCHAR(15),
    Stud_SSN BIGINT,
    Course_ID BIGINT,
    Semester varchar(30),
    Exame_Date DATE ,
    Exame_Mark BIGINT,
    Exame_time VARCHAR(8),
    CONSTRAINT Exams_PK PRIMARY KEY (Exam_Title,Stud_SSN,Course_ID,Semester),
    CONSTRAINT Exams_FK1 FOREIGN KEY(Stud_SSN) REFERENCES Students(Stud_SSN),
    CONSTRAINT Exams_FK2 FOREIGN KEY(Course_ID) REFERENCES courses(Course_ID)
);
```

7.19 <Rooms> TABLE

```
⊖ CREATE table Rooms (
    Room_No BIGINT,
    Floor_No BIGINT,
    Capacity BIGINT,
    CONSTRAINT Rooms PRIMARY KEY (Room_No)
);
```

7.20 <Class> TABLE

```
create table Class (
    Teacher_ID BIGINT,
    Course_ID BIGINT,
    Semester varchar(15),
    C_Period varchar(15),
    Room_No BIGINT,
    Start_time VARCHAR(8),
    CONSTRAINT Class_PK PRIMARY KEY (Teacher_ID,Course_ID),
    CONSTRAINT Class_FK1 FOREIGN KEY(Teacher_ID) REFERENCES Teacher(Emp_ID),
    CONSTRAINT Class_FK2 FOREIGN KEY(Course_ID) REFERENCES courses(Course_ID)
);
```

8 Use Case List

⇒ EMPLOYEE_DEMOGRAPHIC:

- ◆ Title: Add new Employee_ID

Description: Add new employee records to the database

Primary Actor: Administrator

Step 1: Actor clicks on “New employee add” button

Step 2: A new EMPLOYEE_ID is apparent.

Step 3: Prompt to enter teacher information: FIRST_NAME ,LAST_NAME , AGE, PHONE_NUMBER, EMAIL, ZIP_CODE , GENDER, EMPLOYEE_ID, COURSE_ID

Step 4: All information is displayed for new employee. Ask for Confirmation.

Step 5: User clicks “Confirm.” New employee is added.

- ◆ Title: Update employee information

Description: Update employee Email, PHONE_ NUMBER

Primary Actor: Administrator

Step 1: Actor clicks on “Edit Employee info” button.

Step 2: Prompt to select EMPLOYEE_ID to be edited.

Step 3: Prompt to edit teacher information: FIRST_NAME, LAST_NAME , AGE, PHONE_NUMBER, EMAIL, ZIP_CODE , GENDER, TEACHER_ID, COURSE_ID

Step 4: All information is displayed for edited Teacher info. Ask for Confirmation.

Step 5: User clicks “Confirm.” Teacher information is updated.

◆ Title: Delete Teacher

Description: Delete specific TEACHER_ID and its related info.

Primary Actor: Administrator

Step 1: Actor clicks on “Delete Employee” button.

Step 2: Prompt to select Employee to be deleted.

Step 3: User selects which Employee_ID to be deleted.

Step 4: Ask for Confirmation.

Step 5: User clicks “Confirm.” Selected EMPLOYEE_ID is deleted.

⇒ **STUDENT_DEMOGRAPHICS:**

◆ Title: Add new student

Description: Add new student records to the database

Primary Actor: Administrator

Step 1: Actor clicks on “New Student add” button

Step 2: A new STUDENT_ID is apparent.

Step 3: Prompt to enter Student information: FIRST_NAME ,LAST_NAME , AGE, PHONE_NUMBER, EMAIL, ZIP_CODE , GENDER, STUDENT_ID, COURSE_ID.

Step 4: All information is displayed for new Student. Ask for Confirmation.

Step 5: User clicks “Confirm.” New Student is added.

◆ Title: Update Student information

Description: Update student EMAIL, PHONE_NUMBER

Primary Actor: Administrator

Step 1: Actor clicks on “Edit Student_Info” button.

Step 2: User selects which STUDENT_ID to be edited.

Step 3: Prompt to edit Student information: FIRST_NAME, LAST_NAME , AGE, PHONE_NUMBER, EMAIL, ZIP_CODE , GENDER, STUDENT_ID, COURSE_ID.

Step 4: All information is displayed for edited Student_Info. Ask for Confirmation.

Step 5: User clicks “Confirm.” Student information is updated.

- ◆ Title: Delete Student Description: Delete STUDENT_ID and its related info.

Primary Actor: Administrator

Step 1: Actor clicks on “Delete Student” button.

Step 2: Prompt to select STUDENT_ID to be deleted.

Step 3: Ask for Confirmation.

Step 4: User clicks “Confirm.” Selected Student is deleted.

⇒ **LEGAL_GUARDIAN_DEMOGRAPHICS:**

- ◆ Title: Add new Legal guardian

Description: Add new guardian records to the database

Primary Actor: Administrator

Step 1: Actor clicks on “New Guardian add” button

Step 2: A new LG_ID is apparent.

Step 3: Prompt to enter Student information: FIRST_NAME ,LAST_NAME , AGE, PHONE_NUMBER, EMAIL, ZIP_CODE , GENDER, LG_ID, STUDENT_ID.

Step 4: All information is displayed for new Student. Ask for Confirmation.

Step 5: User clicks “Confirm.” New Legal Guardian is added.

- ◆ Title: Update Legal Guardian information

Description: Update EMAIL, PHONE_NUMBER

Primary Actor: Administrator

Step 1: Actor clicks on “Edit LG_Info” button.

Step 2: User selects which LG_ID to be edited.

Step 3: Prompt to edit Student information: FIRST_NAME, LAST_NAME , AGE, PHONE_NUMBER, EMAIL, ADDRESS, GENDER, LG_ID, STUDENT_ID.

Step 4: All information is displayed for edited Student_Info. Ask for Confirmation.

Step 5: User clicks “Confirm.” Legal Guardian information is updated.

- ◆ Title: Delete Legal Guardian Description: Delete LG_ID and its related info.

Primary Actor: Administrator

Step 1: Actor clicks on “Delete LEGAL GUARDIAN” button.

Step 2: Prompt to select LG_ID to be deleted.

Step 3: Ask for Confirmation.

Step 4: User clicks “Confirm.” Selected Legal Guardian is deleted.

⇒ **COURSE DEMOGRAPHIC :**

- ◆ Insert new Course

Title: Create a COURSE_ID

Description: Create course along with detailed course details.

Primary Actor: administrator

Step 1: Actor clicks on “New Course” button.

Step 2: A new COURSE_ID.

Step 3: Prompt to enter Course information: COURSE_ID, COURSE_DESCRIPTION, COURSE_DURATION, CLASS_SCHEDULE, ENROLLMENT_CAPACITY, GRADE_COURSE_ID.

Step 4: All information is displayed for the new course. Ask for Confirmation. Step 5: User clicks “Confirm.” New Course is created.

- ◆ Title: Update Course

data Description: Update COURSE_DURATION, CLASS_SCHEDULE Primary

Actor: Administrator

Step 1: Actor clicks on “Edit Course” button

Step 2: Prompt to select Course _ID to be edited.

Step 3: User selects which course to be edited.

Step 4: Prompt to edit Course information: COURSE_ID, COURSE_DESCRIPTION, COURSE_DURATION, CLASS_SCHEDULE, ENROLLMENT_CAPACITY, GRADE_COURSE_ID.

Step 5: All information is displayed for the edited Course. Ask for Confirmation.

Step 6: User clicks “Confirm.” Course information is updated.

◆ Title: Delete Course

Description: Delete specific COURSE_ID and its related details.

Primary Actor: Administrator

Step 1: Actor clicks on “Delete Course” button

Step 2: Prompt to select COURSE_ID of the Course to be deleted.

Step 3: User selects which COURSE_ID to be deleted.

Step 4: Ask for Confirmation.

Step 5: User clicks “Confirm.” Selected Course is deleted.

⇒ BOOK DEMOGRAPHIC :

◆ Insert new BOOK.

Title: Create a BOOK_ID

Description: Create book entry along with detailed book details.

Primary Actor: administrator

Step 1: Actor clicks on “New Book” button.

Step 2: A new BOOK_ID.

Step 3: Prompt to enter BOOK information: BOOK_ID, BOOK_NAME, AUTHOR_NAME, LIBRARY_ID

Step 4: All information is displayed for the new book. Ask for Confirmation.

Step 5: User clicks “Confirm.” New Book is created.

◆ Title: Update Book

Description: Update BOOK_NAME, AUTHOR_NAME

Primary Actor: Administrator

Step 1: Actor clicks on “Edit Book” button.

Step 2: Prompt to select Book_ID to be edited.

Step 3: User selects which book to be edited.

Step 4: Prompt to edit Book information: BBOK_ID, BOOK_NAME, AUTHOR_NAME, LIBRARY_ID

Step 5: All information is displayed for the edited Book. Ask for Confirmation.

Step 6: User clicks "Confirm." Book information is updated.

◆ **Title: Delete Books**

Description: Delete specific BOOK_ID and its related details.

Primary Actor: Administrator

Step 1: Actor clicks on "Book CLASS" button

Step 2: Prompt to select BOOK_ID of the BOOK to be deleted.

Step 3: User selects which BOOK_ID to be deleted.

Step 4: Ask for Confirmation.

Step 5: User clicks "Confirm." Selected BOOK is deleted.

9 Use Case Implementation

Use Case: `student` table

◆ **Insert Statement**

Inserting a new row into the `Students` table:

```
```sql
```

```
INSERT INTO Students (Stud_SSN, Fname, Minit, Lname, Bdata, Stud_Address, Email)
```

```
VALUES (1112223340, 'Ali', 'Fahad', 'Al-Ahmad', '01-Jan-2000', '123 Street Name',
'ali.ahmad@example.com');
```

```
...
```

◆ **Delete Statement**

Deleting a specific student from the `Students` table by `Stud\_SSN`:

```
```sql
```

```
DELETE FROM Students WHERE Stud_SSN = 1112223340;
```

```
```
```

#### ◆ Update Statement

Updating the email address for a student in the `Students` table:

```
```sql
```

```
UPDATE Students
```

```
SET Email = 'new.email@example.com'
```

```
WHERE Stud_SSN = 1112223330;
```

```
```
```

USE CASE: `grades` Table

#### ◆ Insert

Inserting a new grade record for a student:

```
```sql
```

```
INSERT INTO grades (Stud_SSN, Course_ID, Quarter, Behavior, Attendance, Year_Works)
```

```
VALUES (1112223340, 102, 'Fall 2022', 90, 95, 80);
```

```
```
```

#### ◆ Update

Updating the attendance score for a specific student in a course:

```
```sql
```

```
UPDATE grades
```

```
SET Attendance = 98
```

```
WHERE Stud_SSN = 1112223330 AND Course_ID = 101;
```

```

#### ◆ Delete

Deleting a grade record for a student from a specific course:

```sql

```
DELETE FROM grades  
WHERE Stud_SSN = 1112223340 AND Course_ID = 102;  
```
```

#### ◆ Aggregate – Average

Calculating the average final mark for a course:

```sql

```
SELECT AVG(final_mark)  
FROM grades  
WHERE Course_ID = 101;  
```
```

Use Case: `Members` Table

#### ◆ Insert

Adding a new library member:

```sql

```
INSERT INTO Members (Memb_id, Lib_ID)  
VALUES (1112223341, 10);  
```
```

#### ◆ Update

Updating a member's library ID (if they switch libraries for some reason):

```sql

```
SET Lib_ID = 11  
  
WHERE Memb_id = 1112223330;  
  
'''
```

◆ Delete

Removing a member from the library system:

```
```sql
DELETE FROM Members
WHERE Memb_id = 1112223341;
```

```

◆ Aggregate - Count

Counting the total number of members in a specific library:

```
```sql
SELECT COUNT(*)
FROM Members
WHERE Lib_ID = 10;
````
```

Use Case: `Employee` Table

◆ **Insert**

Adding a new employee record:

```
```sql
```

```
INSERT INTO Employee (SSN, Emp_ID, Fname, Minit, Lname, Sex, Salary, Email, Bdata, Super_ID)
VALUES (1192240497, 00011, 'Laila', 'A', 'Al-Ghamdi', 'F', 18000.00, 'laila.gh@example.com',
'1985-05-05', 00003);
```

```
```
```

◆ Update

Updating the salary for an employee:

```
```sql
```

```
UPDATE Employee
SET Salary = 19000.00
WHERE Emp_ID = 00011;
```

```
```
```

◆ Delete

Removing an employee record:

```
```sql
```

```
DELETE FROM Employee
WHERE Emp_ID = 00011;
```

```
```
```

◆ Aggregate - Sum

Calculating the total salary expenditure for the organization:

```
```sql
```

```
SELECT SUM(Salary)
FROM Employee;
```

```
```
```

```
# some use case for using aggregate functions
```

◆ Aggregate Functions

Count

Counting the number of students enrolled in a specific course ('101' for example):

```
```sql
```

```
SELECT COUNT(*)
FROM Enrollment
WHERE Course_ID = 101;
```
```

◆ Sum

Calculating the total salary of all employees:

```
```sql
```

```
SELECT SUM(Salary)
FROM Employee;
```
```

◆ Average

Calculating the average salary of all employees:

```
```sql
```

```
SELECT AVG(Salary)
FROM Employee;
```
```

◆ Min/Max

Finding the minimum and maximum salary among all employees:

```
```sql
```

```
-- Minimum salary
```

```
SELECT MIN(Salary)
```

```
FROM Employee;
```

```
-- Maximum salary
```

```
SELECT MAX(Salary)
```

```
FROM Employee;
```

```
```
```

◆ Group By and Having

Calculating the total number of students enrolled per course and filtering to show only those with more than 2 enrollments:

```
```sql
```

```
SELECT Course_ID, COUNT(*) AS NumOfStudents
```

```
FROM Enrollment
```

```
GROUP BY Course_ID
```

```
HAVING COUNT(*) > 2;
```

```
```
```

| Course_ID | NumOfStudents |
|-----------|---------------|
| 101 | 8 |
| 102 | 8 |
| 103 | 3 |

Result 30

10 Queries Testing

10.1 <Year_ Works library members>

Query in natural language (ENGLISH)

Select all library members and order by their years of work from high to low

SQL script

```

1   -- select all library members and order by their year-work high to low
2 •  select s.Fname, s.Lname,g.final_mark as Grades
3   from Students s, grades g
4   where s.stud_SSN= g.stud_SSN
5   ⊖ and s.stud_SSN IN (select m.Memb_id
6     from members m
7     inner join
8       grades on Memb_id = grades.stud_SSN)
9   ORDER BY g.final_mark DESC;

```

Caption of the first five rows of the output

| Fname | Lname | Grades |
|--------|-----------|--------|
| Saleh | Al-Malik | 80 |
| Naif | Al-Bassam | 80 |
| Ahmad | Al-Khaled | 79 |
| Ahmad | Al-Sultan | 76 |
| Salman | Al-Turki | 75 |
| Faisal | Al-Salman | 74 |

10.2 <Covid-19 students>

Query in natural language (ENGLISH)

- List the names of the students who took the same subjects in the year 2020

SQL script

```

-- Covid-19 students
•  select distinct Fname
  FROM Enrollment enrl, Students s, courses c
  ⊖ where Fname IN (select Fname
    from Enrollment enrl, Students s, courses c
    where enrl.Stud_SSN = s.Stud_SSN)
    AND
  ⊖ Fname In(select Fname
    From Enrollment enrl, Students s, courses c
    where c.Course_ID = enrl.Course_ID AND En_year = 2020
  );

```

Caption of the first five rows of the output

| Fname |
|-----------|
| Ahmad |
| Saleh |
| Salman |
| Faisal |
| Sultan |
| Osama |
| Naif |
| Abdulaziz |

10.3 <Non-Physics 1 teachers>

Query in natural language (ENGLISH)

Print the Teacher_ID, first name, last name, of those who do not give a PHYS1 course.

SQL script

```
-- Non-physics teachers
Select e.emp_id, e.fname, e.lname
From Employee e, Teacher t
where e.emp_ID = t.emp_ID
And e.emp_id NOT IN (select t.emp_id
From class c, Courses co, teacher t
Where t.emp_id = c.Teacher_ID and
c.Course_ID = co.Course_ID and co.course_name= 'PHYS1');
```

Caption of the first five rows of the output

| emp_id | fname | lname |
|--------|--------|-----------|
| 3 | Bader | Al-Salem |
| 5 | Fahd | Al-Harthi |
| 6 | Faisal | Al-Mahri |
| 7 | Majed | Al-Zhrani |

10.4 <All failed students in 2021>

Query in natural language (ENGLISH)

Print all the students who fail in 2021

SQL script

```
-- students who fail in 2021
select s.Stud_ssn,s.Fname, s.Lname, e.stud_status
from Students s, Enrollment e
where
s.Stud_ssn = e.Stud_SSN and e.Stud_status = 'Fail' and e.En_year = '2021';
```

Caption of the first five rows of the output

| Stud_ssn | Fname | Lname | stud_status |
|------------|--------|-----------|-------------|
| 1112223336 | Osama | Al-Majed | Fail |
| 1112223331 | Saleh | Al-Malik | Fail |
| 1112223337 | Naif | Al-Bassam | Fail |
| 1112223331 | Saleh | Al-Malik | Fail |
| 1112223333 | Faisal | Al-Turki | Fail |
| 1112223330 | Ahmad | Al-Khaled | Fail |

10.5 <Fundamental year students>

Query in natural language (ENGLISH)

- list all students who took MATH1 and in 2021

SQL script

```
-- all the students who took MATH1 in 2021

select s.Stud_SSN, s.Fname, s.Lname
from Students s, Enrollment e
where s.stud_ssn = e.stud_ssn and
e.course_id = 103 and
e.En_year = '2021';
```

Caption of the first five rows of the output

| | Stud_SSN | Fname | Lname | |
|--|------------|--------|----------|--|
| | 1112223331 | Saleh | Al-Malik | |
| | 1112223333 | Faisal | Al-Turki | |

11 Conclusion

In the School Database Management System which we created, we tried to add all the essential tables and diagrams. We have more than six strong entities and other connected and referencing tables. We continue our presentation with steps for each instance of the use case for each entity, then with the use case Implementation using MySQL commands. Here we have represented our ER model and we broken down the model into relationship and normalization. Finally, we showed the DB schema and as this Database is going to be used for Fort Bend School system apps, there will be more tables /entities added in the future like the cafeteria, different club, medical system for students, sports etc. And finally, this database will be added to the application of Fort Bend ISD system where both user and administration can access for edit, search, and report about their queries easily.

12 References

<https://www.fortbendisd.com/domain/83>

<https://www.wikipedia.org/>

<https://openai.com/>

APPENDIX

Employee Table

| SSN | EMP_ID | FNAME | MINIT | LNAME | S | SALARY | EMAIL |
|------------|--------|-----------|----------|-----------|---|--------|-----------------------------|
| 1114048696 | 1 | Saad | hamed | Al-Khaldi | M | 15000 | saadkk18@gmail.com |
| 1195040296 | 2 | Ahmed | Saleh | Al-Qhtani | M | 20000 | Ahmedqq50@gmail.com |
| 1122048724 | 3 | Bader | Mohammed | Al-Salem | M | 16000 | saadkk18@gmail.com |
| 1195040496 | 4 | Mohammed | Saud | Al-Hmdan | M | 20000 | MohammedHmdan@gmail.com |
| 1112048386 | 5 | Fahd | Ahmed | Al-Harthi | M | 14000 | FahdAAA@gmail.com |
| 1114048096 | 6 | Faisal | Nawaf | Al-Mahri | M | 19000 | FaijalMahri@gmail.com |
| 1112228096 | 7 | Majed | Bader | Al-Zhrani | M | 18000 | Majed0k18@gmail.com |
| 1192240496 | 8 | Abdalulah | Sadd | Al-Gamdi | M | 15000 | AbdalulahAl-Gamdi@gmail.com |
| 1112228303 | 9 | Osama | Mohammed | Al-Faisal | M | 17000 | Osama0k18@gmail.com |
| 1146158386 | 10 | Ishaq | Ibrahim | Mohameed | M | 3000 | AhmedBdran@gmail.com |

Emp_Phone

| EMP_ID | EMP_PHONE |
|-------------------|-----------------|
| 1 | 0557781014 |
| 1 | 0594664521 |
| 2 | 0507790007 |
| EMP_ID | SPECIALIZATION |
| 3 | Math |
| 4 | Phys |
| EMP_ID | SPECIALIZATION |
| 8 | Database Admin |
| 9 | Hardware expert |
| EMP_ID | EMP_PHONE |
| 6 | 0544458712 |
| 7 | 0555582212 |
| 7 | 0564648781 |
| 8 | 0564455662 |
| 8 | 0592774193 |
| 9 | 0547411112 |
| 9 | 0557700054 |
| 10 | 0555000558 |
| 10 | 0584445112 |
| 20 rows selected. | |

Teacher

IT_Worker

Cleaning_Staff

| EMP_ID |
|--------|
| 10 |

Librarian

| EMP_ID |
|--------|
| 1 |
| 2 |

lib_managers

| EMP_ID | LIB_ID | STRT_DATE |
|--------|--------|-----------|
| 1 | 12345 | 08-MAY-00 |

Library

| LIB_ID | LIB_NAME |
|--------|----------|
| 10 | Iqra |

Books

| BOOK_ID | BOOK_NAME | AUTHOR | LIB_ID |
|---------|--------------------------|----------------|--------|
| 211 | chess stratgies | Jeremy Silman | 10 |
| 212 | Mein Kampf | adolf hetler | 10 |
| 323 | dance with life | hadia almosa | 10 |
| 434 | platos republic | oxford | 10 |
| 545 | 1980 | gorge orel | 10 |
| 656 | long shadow | hean webster | 10 |
| 437 | this winter | Alice oseman | 10 |
| 358 | the art of a simple life | shinmu manzu | 10 |
| 639 | chaser | hassan aljundi | 10 |
| 5310 | how to pull money | joseph murfy | 10 |
| 5311 | animal farm | goerge orel | 10 |

11 rows selected.

Members

| MEMB_ID | LIB_ID |
|------------|--------|
| L112223330 | 10 |
| L112223331 | 10 |
| L112223332 | 10 |
| L112223333 | 10 |
| L112223335 | 10 |
| L112223337 | 10 |
| L112223339 | 10 |

7 rows selected.

Students

| STUD_SSN | FNAME | MINIT | LNAME | BDATA | STUD_ADDRESS | EMAIL |
|------------|-----------|----------|-----------|-----------|--------------|-----------------------|
| l112223330 | Ahmad | Abdullah | Al-Khaled | 05-APR-05 | | AhmadAK10@gmail.com |
| l112223331 | Saleh | Hussam | Al-Malik | 10-OCT-07 | | Saleh21@gmail.com |
| l112223332 | Salman | Samy | Al-Turki | 15-SEP-06 | | Salman19@gmail.com |
| l112223333 | Faisal | Saleh | Al-Turki | 26-OCT-05 | | Faisal8@gmail.com |
| l112223334 | Sultan | Fahad | Al-Sultan | 29-APR-05 | | Sultan23@gmail.com |
| l112223335 | Faisal | Majed | Al-Salman | 01-SEP-07 | | Faisal98@gmail.com |
| l112223336 | Osama | Ibrahim | Al-Majed | 03-MAY-05 | | Osama45@gmail.com |
| l112223337 | Naif | Nawaf | Al-Bassam | 07-APR-06 | | Naif99@gmail.com |
| l112223338 | Abdulaziz | Mussab | Al-Mussab | 09-OCT-07 | | Abdulaziz17@gmail.com |
| l112223339 | Ahmad | Ali | Al-Sultan | 01-MAY-07 | | Ahmad999@gmail.com |

10 rows selected.

Courses

| COURSE_ID | COURSE_NAME | PREREQUISITE |
|-----------|-------------|--------------|
| 101 | MATH1 | |
| 102 | COMP1 | |
| 103 | MATH2 | MATH1 |
| 104 | COMP2 | COMP1 |
| 105 | CHEM1 | |
| 106 | PHYS1 | |
| 107 | CHEM2 | CHEM1 |
| 108 | PHYS2 | PHYS1 |
| 109 | ISLM1 | |
| 110 | ARAB1 | |
| 111 | ISLM2 | ISLM1 |

11 rows selected.

Enrollment

| STUD_SSN | COURSE_ID | EN_YEAR | STUD_STATUS | QUARTER |
|------------|-----------|---------|-------------|---------|
| 1112223330 | 101 | 2021 | Pass | 1 |
| 1112223330 | 102 | 2021 | Pass | 1 |
| 1112223330 | 106 | 2021 | Fail | 1 |
| 1112223331 | 101 | 2020 | Pass | 1 |
| 1112223331 | 103 | 2021 | Fail | 1 |
| 1112223331 | 102 | 2021 | Fail | 1 |
| 1112223332 | 101 | 2020 | Pass | 1 |
| 1112223332 | 105 | 2020 | Pass | 1 |
| 1112223332 | 102 | 2020 | Pass | 1 |
| 1112223333 | 101 | 2020 | Fail | 1 |
| 1112223333 | 101 | 2020 | Pass | 2 |
| STUD_SSN | COURSE_ID | EN_YEAR | STUD_STATUS | QUARTER |
| 1112223333 | 103 | 2021 | Fail | 1 |
| 1112223334 | 101 | 2020 | Pass | 2 |
| 1112223334 | 102 | 2020 | Pass | 2 |
| 1112223334 | 105 | 2020 | Pass | 2 |
| 1112223335 | 101 | 2020 | Pass | 2 |
| 1112223335 | 102 | 2020 | Pass | 1 |
| 1112223335 | 103 | 2020 | Pass | 2 |
| 1112223336 | 101 | 2021 | Fail | 1 |
| 1112223337 | 102 | 2021 | Fail | 1 |
| 1112223338 | 102 | 2021 | Pass | 1 |
| 1112223339 | 102 | 2021 | Pass | 1 |

22 rows selected.

Legal_Guardian

| STUD_SSN | LG_NAME | RELEATIONSHIP |
|------------|----------|---------------|
| 1112223330 | Abdullah | Father |
| 1112223331 | Basma | Mother |
| 1112223332 | Samy | Father |
| 1112223333 | Saleh | Father |
| 1112223334 | KHlod | Mother |
| 1112223335 | Majed | Father |
| 1112223336 | Afnan | Mother |
| 1112223337 | Nawaf | Father |
| 1112223338 | Mussab | Father |
| 1112223339 | Rehab | Mother |

10 rows selected.

LG_Info

| LG_NAME | BDATE | S | LG_EMAIL |
|----------|-----------|---|-----------------------|
| Abdullah | 25-MAY-81 | M | Abdullah112@gmail.com |
| Basma | 02-OCT-89 | F | Basma113@gmail.com |
| Samy | 01-SEP-85 | M | Samy114@gmail.com |
| Saleh | 10-JUL-84 | M | Saleh115@gmail.com |
| KHlod | 15-SEP-87 | F | KHlod116@gmail.com |
| Majed | 29-MAY-80 | M | Majed117@gmail.com |
| Afnan | 14-MAY-79 | F | Afnan118@gmail.com |
| Nawaf | 16-JUL-75 | M | Nawaf119@gmail.com |
| Mussab | 11-SEP-71 | M | Mussab120@gmail.com |
| Rehab | 05-JUL-90 | F | Rehab121@gmail.com |

10 rows selected.

LG_Phones

| LG_NAME | LG_PHONE |
|----------|-----------|
| Abdullah | 545557770 |
| Afnan | 545557776 |
| Basma | 545557771 |
| KHlod | 545557774 |
| Majed | 545557775 |
| Mussab | 545557778 |
| Nawaf | 545557777 |
| Rehab | 545557779 |
| Saleh | 545557773 |
| Samy | 545557772 |

10 rows selected.

Grades

| STUD_SSN | COURSE_ID | QUARTER | BEHAVIOR | ATTENDANCE | YEAR_WORKS |
|------------|-----------|-------------|----------|------------|------------|
| 1112223330 | 101 | summer 2022 | 100 | 100 | 60 |
| 1112223331 | 101 | summer 2022 | 100 | 100 | 60 |
| 1112223332 | 103 | summer 2022 | 100 | 100 | 60 |
| 1112223333 | 106 | summer 2022 | 100 | 100 | 60 |
| 1112223334 | 108 | summer 2022 | 100 | 100 | 60 |
| 1112223335 | 104 | summer 2022 | 100 | 100 | 60 |
| 1112223336 | 111 | summer 2022 | 100 | 100 | 60 |
| 1112223337 | 105 | summer 2022 | 100 | 100 | 60 |
| 1112223338 | 109 | summer 2022 | 100 | 100 | 60 |
| 1112223339 | 107 | summer 2022 | 100 | 100 | 60 |

10 rows selected.

Exams

| EXAM_TITLE | STUD_SSN | COURSE_ID | SEMESTER | EXAME_DAT | EXAME_MARK | EXAME_TI |
|------------|------------|-----------|-------------|-----------|------------|----------|
| Math | 1112223330 | 101 | summer 2022 | 05-JUL-22 | 100 | 00:09:00 |
| Math | 1112223331 | 101 | summer 2022 | 05-JUL-22 | 100 | 00:09:00 |
| Math | 1112223332 | 103 | summer 2022 | 05-JUL-22 | 100 | 00:09:00 |
| comp | 1112223330 | 102 | summer 2022 | 06-JUL-22 | 100 | 00:10:00 |
| comp | 1112223334 | 102 | summer 2022 | 06-JUL-22 | 100 | 00:10:00 |
| comp | 1112223335 | 104 | summer 2022 | 08-JUL-22 | 100 | 00:09:00 |
| comp | 1112223337 | 104 | summer 2022 | 08-JUL-22 | 100 | 00:09:00 |
| CHEM | 1112223337 | 105 | summer 2022 | 19-JUL-22 | 100 | 00:09:00 |
| CHEM | 1112223339 | 107 | summer 2022 | 20-JUL-22 | 100 | 00:09:00 |
| Phys | 1112223333 | 106 | summer 2022 | 20-JUL-22 | 100 | 00:08:00 |
| Phys | 1112223334 | 108 | summer 2022 | 20-JUL-22 | 100 | 00:09:00 |
| EXAM_TITLE | STUD_SSN | COURSE_ID | SEMESTER | EXAME_DAT | EXAME_MARK | EXAME_TI |
| ISLM | 1112223338 | 109 | summer 2022 | 26-JUL-22 | 100 | 00:12:00 |
| ISLM | 1112223336 | 111 | summer 2022 | 29-JUL-22 | 100 | 00:13:00 |
| ARAB | 1112223332 | 110 | summer 2022 | 29-JUL-22 | 100 | 00:13:00 |

14 rows selected.

Rooms

| ROOM_NO | FLOOR_NO | CAPACITY |
|---------|----------|----------|
| 0 | 0 | 20 |
| 1 | 0 | 20 |
| 2 | 0 | 20 |
| 3 | 0 | 20 |
| 4 | 1 | 25 |
| 5 | 1 | 25 |
| 6 | 1 | 25 |
| 7 | 2 | 12 |
| 8 | 2 | 20 |
| 9 | 2 | 25 |
| 10 | 3 | 25 |

11 rows selected.

Class

| TEACHER_ID | COURSE_ID | SEMESTER | C_PERIOD | ROOM_NO | START TI |
|------------|-----------|-------------|----------|---------|-------------|
| 3 | 101 | summer 2022 | 45 | | 0 00:08:00 |
| 6 | 102 | summer 2022 | 45 | | 1 00:09:00 |
| 3 | 103 | summer 2022 | 45 | | 2 00:10:00 |
| 6 | 104 | summer 2022 | 45 | | 1 00:11:00 |
| 7 | 105 | summer 2022 | 45 | | 3 00:12:00 |
| 4 | 106 | summer 2022 | 45 | | 10 00:13:00 |
| 7 | 107 | summer 2022 | 45 | | 3 00:14:00 |
| 4 | 108 | summer 2022 | 45 | | 4 00:15:00 |
| 5 | 109 | summer 2022 | 45 | | 6 00:16:00 |
| 5 | 111 | summer 2022 | 45 | | 2 00:17:00 |

10 rows selected.

