Student Database Management

shawkyebrahim2514

DATABASE SCHEME

```
create database student_db;

use student_db;


create table unregisteredStudents

(

    id int(8) not null

);


create table students

(

    id       int(8) primary key not null,

    password varchar(255)       not null

);


create table personalData

(

    studentID int         not null,

    firstName varchar(50) not null,

    lastName  varchar(50) not null,

    birthday  timestamp   not null,

    gender    varchar(6)  not null,

    foreign key (studentID) references students (id) on delete cascade

);


create table contactData

(

    studentID   int          not null,

    email       varchar(255) not null,

    phoneNumber char(11)     not null,

    address     varchar(255) not null,

    foreign key (studentID) references students (id) on delete cascade

);


create table academicData

(

    studentID int           not null,
    level     int(1)        not null default 1,
    GPA       decimal(3, 2) not null default 0,
    foreign key (studentID) references students (id) on delete cascade
);

create table courses
(
    id   int auto_increment primary key,
    name varchar(255) not null,
    code char(5)      not null
);

create table studentCourse
(
    id        int auto_increment primary key,
    studentID int    not null,
    courseID  int    not null,
    grade     int(3) not null,
    level     int(1) not null,
    semester  int(1) not null,
    foreign key (studentID) references students (id) on delete cascade,
    foreign key (courseID) references courses (id) on delete cascade,
    unique (studentID, courseID)
);

create table notes
(
    id              int auto_increment primary key,
    studentCourseID int not null,
    title           varchar(255),
    content         varchar(3000),
    foreign key (studentCourseID) references studentCourse (id) on delete cascade
);
```

FCAI-CU COURSES

```
insert into courses (code, name)
values ('HU332', 'Creative Thinking'),
       ('DS411', 'Decision and Game Theory'),
       ('MA112', 'Discrete Mathematics'),
       ('HU111', 'English'),
       ('HU323', 'Fundamentals of Accounting'),
       ('HU333', 'Mass Communication'),
       ('MA111', 'Mathematics-1'),
       ('MA113', 'Mathematics-2'),
       ('MA214', 'Mathematics-3'),
       ('MA315', 'Mathematics-4'),
       ('ST121', 'Probability and Statistics-1'),
       ('ST122', 'Probability and Statistics-2'),
       ('HU334', 'Professional Ethics'),
       ('HU112', 'Scientific & Technical Report Writing'),
       ('CS316', 'Algorithms'),
       ('CS361', 'Artificial Intelligence'),
       ('CS318', 'Assembly Language'),
       ('CS419', 'Compilers'),
       ('CS443', 'Computer Arabization'),
       ('CS322', 'Computer Architecture and Organization'),
       ('CS317', 'Concepts of Programming Languages'),
       ('CS214', 'Data Structures'),
       ('CS215', 'File Organization and Processing'),
       ('CS464', 'Genetic Algorithms'),
       ('CS453', 'Human Computer Interfaces'),
       ('CS111', 'Introduction to Computers'),
       ('CS465', 'Knowledge Base Systems'),
       ('CS221', 'Logic Design'),
       ('CS467', 'Machine Learning'),
       ('CS466', 'Multi-Agent Systems'),
       ('CS462', 'Natural Languages Processing'),
       ('CS463', 'Neural Networks'),
       ('CS241', 'Operating System-1'),
       ('CS342', 'Operating Systems-2'),
       ('CS471', 'Parallel Processing'),
('CS112', 'Programming-1'),
('CS213', 'Programming-2'),
('CS498', 'Project'),
('CS495', 'Selected Topics in Computer Science-1'),
('CS496', 'Selected Topics in Computer Science-2'),
('CS251', 'Software Engineering-1'),
('CS352', 'Software Engineering-2'),
('IS352', 'Analysis and Design of Information Systems-2'),
('IS332', 'Business Functions Classification'),
('IS421', 'Data Mining'),
('IS313', 'Data Storage and Retrieval'),
('IS422', 'Data Warehouses'),
('IS414', 'Database Design'),
('IS211', 'Database Systems 1'),
('IS312', 'Database Systems 2'),
('IS416', 'Distributed Databases'),
('IS442', 'E-Commerce'),
('IS231', 'Fundamentals of Information Systems'),
('IS443', 'Geographical Informatiopn Systems'),
('IS435', 'Information Centres Management'),
('IS453', 'Information Systems Development Methodologies'),
('IS441', 'Intelligent Information Systems'),
('IS345', 'Internet Applications'),
('IS446', 'Internet Information Systems'),
('IS333', 'Management Information Systems'),
('IS444', 'Multimedia Information Systems'),
('IS415', 'Object Oriented Databases'),
('IS498', 'Project'),
('IS434', 'Quality Assurance of Information Systems and programming'),
('IS495', 'Selected Topics in Information systems-1'),
('IS496', 'Selected Topics in Information systems-2'),
('IT321', 'Communication Technology'),
('IT432', 'Computer Animation'),
('IT311', 'Computer Architecture'),
('IT331', 'Computer Graphics-1'),
('IT332', 'Computer Graphics-2'),
('IT313', 'Computer Interfaces'),
('IT322', 'Computer Network-2'),
```

('IT222', 'Computer Networks-1'),

('IT444', 'Computer Vision'),

('IT221', 'Data Communication'),

('IT453', 'Digital Library'),

('IT341', 'Digital Signals Processing'),

('IT411', 'Distributed and Parallel Computer Systems'),

('IT451', 'E-Business'),

('IT452', 'E-Learning'),

('IT111', 'Electronic-1'),

('IT112', 'Electronics-2'),

('IT414', 'Embedded Systems'),

('IT413', 'Fault Tolerant Computer Systems'),

('IT441', 'Image Processing-1'),

('IT442', 'Image Processing-2'),

('IT423', 'Information and Computer Networks Security'),

('IT454', 'Information Engineering'),

('IT445', 'Intelligent and Quantum Computers'),

('IT223', 'Internet Technology'),

('IT312', 'Microprocessors'),

('IT433', 'Multimedia'),

('IT342', 'Pattern Recognition'),

('IT421', 'Planning and Design of Information Networks'),

('IT498', 'Project'),

('IT412', 'Real Time Systems'),

('IT415', 'Robotics'),

('IT495', 'Selected Topics in Information Technology-1 (Fuzzy)'),

('IT496', 'Selected Topics in Information Technology-2 (Network)'),

('IT241', 'Signals and Systems'),

('IT443', 'Speech processing'),

('IT431', 'Virtual Reality'),

('IT422', 'Wireless and Mobile Networks'),

('DS426', 'Advanced Project Management'),

('DS451', 'Advanced Topics in Intelligent Computational'),

('DS351', 'Computational Intelligence in Decision Support'),

('DS342', 'Computer Languages for Modeling'),

('DS443', 'Computer Simulation Languages'),

('DS432', 'Data Management in Decision Support'),

('DS332', 'Decision Support Systems and Applications'),

```
('DS331', 'Decision Support Tools and Techniques'),
('DS491', 'Elective Course'),
('DS492', 'Elective Course'),
('DS493', 'Elective Course'),
('DS121', 'Fundamentals of Economics'),
('DS122', 'Fundamentals of Management'),
('DS431', 'Geographic Information Systems for Decision Support'),
('DS211', 'Introduction to Decision Support and Systems'),
('DS424', 'Inventory Control and Production Management'),
('DS433', 'Knowledge Base Decision Support systems'),
('DS311', 'Linear and Integer Programming'),
('DS425', 'Logistics Management'),
('DS241', 'Modeling and Simulation'),
('DS414', 'Multi-Objective Programming'),
('DS413', 'Networks Optimization'),
('DS312', 'Non-Linear and Dynamic Programming'),
('DS313', 'Optimizations Techniques'),
('DS498', 'Project'),
('DS321', 'Projects Management'),
('DS423', 'Quantitative Models for Services'),
('DS422', 'Quantitative Models in Economics and Management'),
('DS415', 'Risk Management'),
('DS495', 'Selected Topics in Decision Support'),
('DS444', 'Simulation Games'),
('DS442', 'Simulation Models in management and Economics'),
('DS461', 'Statistical Analysis in Decision Support'),
('DS361', 'Stochastic Models in Operations Research and Decision Support'),
('DS462', 'Stochastic Programming'),
('DS412', 'Strategic and Crisis Management'),
('DS441', 'System Analysis and Modeling');
```

Students

- Register students

```
-- inserting into students table
INSERT INTO students (id, password)
values (${data.studentID}, '${data.password}');
```

```
-- inserting into personalData table
INSERT INTO personalData (studentID, firstName, lastName, birthday, gender)
values (${data.studentID}, '${data.firstName}', '${data.lastName}', '${data.birthday}',
        '${data.gender}');
```

```
-- inserting into contactData table
INSERT INTO contactData (studentID, email, phoneNumber, address)
values (${data.studentID}, '${data.email}', '${data.phoneNumber}', '${data.address}');
```

```
-- inserting into academicData table
INSERT INTO academicData (studentID, level, GPA)
values (${data.studentID}, ${data.level}, ${data.gpa});
```

```
-- deleting from unregisteredStudents table
delete from unregisteredStudents
where id = ${data.studentID};
```

- login

```
-- collect student ID and password to verify the login
select *
from students
where id = ${studentID};
```

```
-- collect all student data to show it in his profile page
select students.id              as studentID,
       students.password        as password,
       personalData.firstName   as firstName,
       personalData.lastName    as lastName,
       personalData.birthday    as birthday,
       personalData.gender      as gender,
       contactData.email        as email,
       contactData.phoneNumber as phoneNumber,
       contactData.address     as address,
       academicData.level      as level,
       academicData.GPA        as gpa
from students
        join personalData on students.id = personalData.studentID
        join contactData on students.id = contactData.studentID
        join academicData on students.id = academicData.studentID
where students.id = ${studentID};
```

Update students’ data

- update student’s password

```
update students
set password = ‘${data.password}’
where id = ${data.studentID};
```

- Update student’s personal data

```
update personalData
set firstName = ‘${data.firstName}’,
    lastName  = ‘${data.lastName}’
where studentID = ${data.studentID};
```

- Update student’s contact data

```
update contactData
set email       = ‘${data.email}’,
    phoneNumber = ‘${data.phoneNumber}’,
    address     = ‘${data.address}’
where studentID = ${data.studentID};
```

- Update student’s academic data

```
update academicData
set level = ‘${data.level}’,
    gpa   = ‘${data.gpa}’
where studentID = ${data.studentID};
```

Courses

- Select all existing courses' materials

```
select id, name, code
from courses;
```

- Collect all studentCourses' data of a student

```
select studentCourse.id       as courseID,
       courses.name           as courseName,
       courses.code           as courseCode,
       studentCourse.grade    as courseGrade,
       studentCourse.level    as courseLevel,
       studentCourse.semester as courseSemester
from students
        join studentCourse on students.id = studentCourse.studentID
        join courses on studentCourse.courseID = courses.id
where students.id = ${req.session.user.id}
order by courseLevel, courseSemester;
```

- Check a course belongs to a student

```
select studentID
from studentCourse
where studentID = ${studentId}
  and courseID = ${courseID};
```

- Inserting a new course into studentCourse

```
INSERT INTO studentCourse (studentID, courseID, grade, level, semester)
values (${studentID}, ${data.courseID}, ${data.courseGrade},
        ${data.courseLevel}, ${data.courseSemester});
```

- Collect studentCourse's information

```
select courses.id             as courseID,
       courses.code           as courseCode,
       courses.name           as courseName,
       studentCourse.grade    as courseGrade,
       studentCourse.level    as courseLevel,
       studentCourse.semester as courseSemester
from studentCourse
        join courses on studentCourse.courseID = courses.id
where studentCourse.id = ${studentCourseID};
```

- Update studentCourse information

```
update studentCourse
set courseID = '${data.courseID}',
    grade    = ${data.courseGrade},
    level    = ${data.courseLevel},
    semester = ${data.courseSemester}
where id = ${studentCourseID};
```

- Delete studentCourse

```
delete
from studentCourse
where id = ${studentCourseID};
```

Notes

- Collect all notes’ data of a studentCourse

```
select notes.id         as noteID,
       if(char_length(notes.title) > 45, concat(substr(notes.title, 1, 45), ‘...’),
          notes.title)   as noteTitle,
       if(char_length(notes.content) > 100,
          concat(substr(notes.content, 1, 100), ‘...’),
          notes.content) as noteContent
from notes
         join studentCourse on notes.studentCourseID = studentCourse.id
where studentCourse.id = ${studentCourseID};
```

- Search in studentCourse notes

```
select notes.id         as noteID,
       if(char_length(notes.title) > 45, concat(substr(notes.title, 1, 45), ‘...’),
          notes.title)   as noteTitle,
       if(char_length(notes.content) > 100,
          concat(substr(notes.content, 1, 100), ‘...’),
          notes.content) as noteContent
from notes
         join studentCourse on notes.studentCourseID = studentCourse.id
where studentCourse.id = ${studentCourseID}
  and (notes.title like ‘%${searchWord}%’
    or notes.content like ‘%${searchWord}%’);
```

- Add new note to a studentCourse

```
insert into notes (studentCourseID, title, content)
values (${req.params.studentCourseID}, ‘${req.body.title}’, ‘${req.body.content}’);
```

- Collect note’s information

```
select title, content
from notes
where notes.id = ${noteID};
```

- Update note’s data

```
update notes
set title   = ‘${req.body.title}’,
    content = ‘${req.body.content}’
where id = ${req.params.noteID};
```

- Delete note

```
delete
from notes
where id = ${noteID};
```

Admins

Login

- Collect admin ID and password to verify the login

```
select *
from admins
where id = '${adminID}';
```

Admin panel

- Collect admin information to show it in admin panel

```
select admin_db.personalData.firstName   as firstName,
       admin_db.personalData.lastName    as lastName,
       admin_db.personalData.phoneNumber as phoneNumber,
       admin_db.personalData.email       as email
from admins
        join personalData on admins.id = admin_db.personalData.adminID
where admins.id = '${req.session.user.id}';
```

- Check if a range of IDs exist in students table

```
select id
from students
where id between ${startID} and ${endID};
```

- Check if a range of IDs exist in unregisteredStudents table

```
select id
from unregisteredStudents
where id between ${startID} and ${endID};
```

- Insert a range of IDs to unregisteredStudents table

```
INSERT INTO unregisteredStudents (id)
VALUES ?;
```

Show students’ data

- Show students that their IDs are in a range of IDs

```
select students.id          as studentID,
       personalData.firstName as firstName,
       personalData.lastName  as lastName,
       academicData.level     as level,
       academicData.GPA       as gpa
from students
        join personalData on students.id = personalData.studentID
        join contactData on students.id = contactData.studentID
        join academicData on students.id = academicData.studentID
where students.id between ${startID} and ${endID};
```

- Show students that their level is equal to a specific level

```
select students.id          as studentID,
       personalData.firstName as firstName,
       personalData.lastName  as lastName,
       academicData.level     as level,
       academicData.GPA       as gpa
from students
        join personalData on students.id = personalData.studentID
        join contactData on students.id = contactData.studentID
        join academicData on students.id = academicData.studentID
where academicData.level = ${level};
```

- Show students that their GPAs are in a range of GPAs

```
select students.id          as studentID,
       personalData.firstName as firstName,
       personalData.lastName  as lastName,
       academicData.level     as level,
       academicData.GPA       as gpa
from students
        join personalData on students.id = personalData.studentID
        join contactData on students.id = contactData.studentID
        join academicData on students.id = academicData.studentID
where academicData.gpa between ${startGPA} and ${endGPA};
```

Edit students’ data

- Get the column data type

```
SELECT data_type
FROM information_schema.columns
WHERE table_schema = ‘student_db’
  AND table_name = ‘${tableName}’
  AND column_name = ‘${columnName}’;
```

- Update a table with a column of an integer data type

```
update ${data.tableName}
set ${data.columnName} = ${data.updateValue}
where studentID between ${data.startID} and ${data.endID};
```

- Update a table with a column of a non integer data type

```
update ${data.tableName}
set ${data.columnName} = ‘${data.updateValue}’
where studentID between ${data.startID} and ${data.endID};
```

Delete students

```
delete
from students
where id between ${startID} and ${endID};
```

Add new students

```
INSERT INTO unregisteredStudents (id)
VALUES ?;
```