

Databases I, Winter 2023
Milestone 2
Submission: 01/12/2023 (11:59 pm)

You are required to submit a zip file named after your **Team Number** containing the following documents:

- a) SQL file
- b) A PDF file containing:
 - Team name
 - Team leader name, ID and tutorial.
 - Team members' names.
 - Team members' IDs.
 - Team members' tutorial groups.

1. Important Guidelines:

- All alpha-numeric attributes should be defined with type varchar (40).
- All numeric values should be defined with data type int or decimal.
- You must follow the names of the required functions, stored procedures and views. **Not matching the names will result in grade loss.**
- You must follow the order and data types of the parameters required for a stored procedure or function with a specific order and certain data type. Failing to follow this guideline will result in grade loss of that function/stored procedure/view.

- You must follow the order of the needed columns in case a table is required to be returned from a function or fetched from a view.
- You should do the appropriate join if you are required to implement a function/view that fetches data from different tables in your database.
- You should insert the data in the appropriate tables in your database if the required stored procedure inserts data in multiple tables in your database.
- Always think about all **consequences** and required actions to be done in every requirement.
- The status of any sent request can either be 'pending' (default), 'accepted' or 'rejected'.
- The system needs to keep track of the selected courses' grade per exam type and semester.
 - The type of any Exam can either be 'Normal' (default), 'First_makeup' or 'Second_makeup'.
 - The grade will be 'Null' if the student is still taking the course or did not attend the Exam. If he/she took the course, a value should be assigned to the grade.
- The status of the student (blocked or unblocked) should be represented using bit data type. if the student does not pay the installment by the deadline, the student will be blocked and unable to log in to the system.
 - For students, 0 means blocked and 1 means unblocked.
- The status of any Payment can either be 'notPaid' (default), or 'Paid'.
- The Offered attribute of the course should be represented using bit data type

- For Course, '0' means course is not offered in the current semester while '1' means course is offered in the current semester.
- If the student has only one payment with no installments this means that number of installment attribute value should be '0' not Null.
- If the student has any number of installments, please consider the below:
 - Number of installments should be equals to the number of months between the payment's start date and the payment's deadline.
 - The first installment's start date equals to the payment start date.
 - The start date of each upcoming installment is equals to the previous installment's deadline.
 - The deadline of each installment is a month from the installment's start date.
 - Installment's amount is equal to (payment amount / total number of installments).
- Required courses: A course is considered required (according to a certain student) if it is unattended or failed.
 - Failed course: Student failed in this course before and is not eligible for the 2nd makeup.
 - Unattended Course: A mandatory course from previous semesters that the student did not take before (Must be a course from a semester less than student's current semester).
- Optional Courses: courses from the current semester or upcoming semesters (Student is allowed to take the optional course if he/she satisfied their prerequisites).

- Missing courses: The courses remaining to be taken at any semester in order for the student to graduate.
- **Any compilation/syntax error in any of the requirements will result in grade loss**

2. Requirements:

You are required to implement the following:

2.1 Basic Structure of the Database:

1. Write an SQL query to create database called “Advising_Team_Your Team Number”
2. Put the queries that create all the tables of your database with their definition inside the below procedure.
 - i. Type: stored procedure
 - ii. Name: CreateAllTables
 - iii. Input: Nothing
 - iv. Output: Nothing
3. Create a procedure that Drop all tables inside your database.
 - i. Type: stored procedure
 - ii. Name: DropAllTables
 - iii. Input: Nothing
 - iv. Output: Nothing
4. Create a Stored procedure to clear all records in all tables existing in your database.

- i. Type: stored procedure
- ii. Name: clearAllTables
- iii. Input: Nothing
- iv. Output: Nothing

2.2 Basic Data Retrieval

A) Fetch details for all Active students.

- i. Type: view
- ii. Name: view_Students
- iii. Input: Nothing
- iv. Output: Student Table

B) Fetch details for all courses with their prerequisites

- i. Type: view
- ii. Name: view_Course_prerequisites
- iii. Input: Nothing
- iv. Output: Table

C) Fetch details for all Instructors along with their assigned courses

- i. Type: view
- ii. Name: Instructors_AssignedCourses
- iii. Input: Nothing
- iv. Output: Table

D) Fetch details for all payments along with their corresponding student

- i. Type: view
- ii. Name: Student_Payment
- iii. Input: Nothing

iv. Output: Table

E) Fetch details for all courses along with their corresponding slots' details and instructors

- i. Type: view
- ii. Name: Courses_Slots_Instructor
- iii. Input: Nothing
- iv. Output: Table (CourseID, Course.name, Slot ID, Slot Day, Slot Time, Slot Location, Slot's Instructor name)

F) Fetch details for all courses along with their exams' details.

- i. Type: view
- ii. Name: Courses_MakeupExams
- iii. Input: Nothing
- iv. Output: Table (Course's name, Course's semester, MakeUp_Exam details)

G) Fetch students along with their taken courses' details

- i. Type: view
- ii. Name: Students_Courses_transcript
- iii. Input: Nothing
- iv. Output: table (Student id, student name, course id, course name, exam type, course grade, semester (the student took the course in), and Instructor's name)

H) Fetch all semesters along with their offered courses

- i. Type: view
- ii. Name: Semster_offered_Courses
- iii. Input: Nothing

iv. Output: Table (Course id, Course name, Semester code)

I) Fetch all graduation plans along with their initiated advisors

- i. Type: view
- ii. Name: Advisors_Graduation_Plan
- iii. Input: Nothing
- iv. Output: Table (Graduation_Plan details, Advisor id, Advisor name)

2.3 All Other Requirements

A) Student Registration

- i. Type: Stored Procedure
- ii. Name: Procedures_StudentRegistration
- iii. Input: first name varchar (40), last name varchar (40), password varchar (40), faculty varchar (40), email varchar (40), major varchar (40), Semester int.
- iv. Output: Student id int

B) Advisor Registration

- i. Type: Stored Procedure
- ii. Name: Procedures_AdvisorRegistration
- iii. Input: advisor name varchar (40), password varchar (40), email varchar (40), office varchar (40),
- iv. Output: Advisor id int

C) List all advising students

- i. Type: Stored Procedure

- ii. Name: Procedures_AdminListStudents
- iii. Input: Nothing
- iv. Output: Table

D) List all Advisors

- i. Type: Stored Procedure
- ii. Name: Procedures_AdminListAdvisors
- iii. Input: Nothing
- iv. Output: Table

E) List all Students with their Advisors

- i. Type: Stored Procedure
- ii. Name: AdminListStudentsWithAdvisors
- iii. Input: Nothing
- iv. Output: Table

F) Add new Semester

- i. Type: Stored Procedure
- ii. Name: AdminAddingSemester
- iii. Input: start_date date, end_date date, and semester code
- iv. Output: Nothing

G) Add new course

- i. Type: Stored Procedure
- ii. Name: Procedures_AdminAddingCourse
- iii. Input: major varchar (40), semester int, credit hours int, course name varchar (40), and offered bit

- iv. Output: Nothing

H) Link instructor to course on specific slot

- i. Type: Stored Procedure
- ii. Name: Procedures_AdminLinkInstructor
- iii. Input: InstructorId int, courseId int, and slotID int
- iv. Output: Nothing

I) Link student to course with Specific instructor

- i. Type: Stored Procedure
- ii. Name: Procedures_AdminLinkStudent
- iii. Input: Instructor Id int, student ID int, course ID, and semester code varchar (40)
- iv. Output: Nothing

J) Link student to advisor

- i. Type: Stored Procedure
- ii. Name: Procedures_AdminLinkStudentToAdvisor
- iii. Input: studentID int, advisorID int
- iv. Output: Nothing

K) Admin add exam

- i. Type: Stored Procedure
- ii. Name: Procedures_AdminAddExam
- iii. Input: Type varchar (40), date datetime, courseID int
- iv. Output: Nothing



J) Issue installments as per the number of installments for a certain payment

- i. Type: Stored Procedure
- ii. Name: Procedures_AdminIssueInstallment
- iii. Input: payment ID int
- iv. Output: Nothing

M) Delete courses along with its related slots

- i. Type: Stored Procedure
- ii. Name: Procedures_AdminDeleteCourse
- iii. Input: courseID int
- iv. Output: Nothing

N) Update student's Status based on his/her financial status, if the students' installment status was "NotPaid" and the installment's deadline was passed then this student should be blocked.

(Student status should be 0 in case the student is blocked and 1 in case the student is Active).

- i. Type: Stored Procedure
- ii. Name: Procedure_AdminUpdateStudentStatus
- iii. Input: StudentID int
- iv. Output: Nothing

O) List all pending requests

- i. Type: View
- ii. Name: all_Pending_Requests
- iii. Input: Nothing

- iv. Output: Table (Pending requests details, initiated student name and Related advisor name)

P) Delete slots of certain courses if the course isn't offered in the current semester

- i. Type: Stored Procedure
- ii. Name: Procedures_AdminDeleteSlots
- iii. Input: current_semester varchar (40)
- iv. Output: Nothing

Q) Advisor login using username and password.

- i. Type: Scalar Function
- ii. Name: FN_AdvisorLogin
- iii. Input: ID int, password varchar (40)
- iv. Output: Success bit

R) Insert graduation Plan

- i. Type: Stored Procedure
- ii. Name: Procedures_AdvisorCreateGP
- iii. Input: Semester code varchar (40),
expected_graduation_date date, sem_credit_hours int,
advisor id int, student id int
- iv. Output: Nothing

S) Add course inside certain plan of specific student

- i. Type: Stored Procedure
- ii. Name: Procedures_AdvisorAddCourseGP

- iii. Input: student id int, Semester_code varchar (40), course name varchar (40).
- iv. Output: Nothing

T) Update expected graduation date in a certain graduation plan

- i. Type: Stored Procedure
- ii. Name: Procedures_AdvisorUpdateGP
- iii. Input: expected_grad_semster varchar (40) and studentID int
- iv. Output: nothing

U) Delete course from certain graduation plan in certain semester

- i. Type: stored procedure
- ii. Name: Procedures_AdvisorDeleteFromGP
- iii. Input: studentID int, semester code varchar (40) and course ID (the course he/she wants to delete)
- iv. Output: nothing

V) Retrieve requests for certain advisor

- i. Type: TVFunction
- ii. Name: FN_Advisors_Requests
- iii. Input: advisorID int
- iv. Output: Table (Requests details related to this advisor)

W) Approve/Reject extra credit hours' request

- i. Type: Stored Procedure
- ii. Name: Procedures_AdvisorApproveRejectCHRequest
- iii. Input: RequestID int, Current semester code varchar (40)

iv. Output: nothing

X) View all students assigned to specific advisor from a certain major along with their taken courses

- i. Type: Stored Procedure
- ii. Name: Procedures_AdvisorViewAssignedStudents
- iii. Input: AdvisorID int and major varchar (40)
- iv. Output: Table (Student id, Student name, Student major, Course name)

Y) Approve/Reject courses request

After approving/rejecting the request, the status of the request should be updated and all consequences should be handled. The approving/rejecting is based on the below conditions:

- All the Requested course's prerequisites are taken.
 - Student has enough assigned hours for the requested course.
- i. Type: Stored Procedure
 - ii. Name:Procedures_AdvisorApproveRejectCourseRequest
 - iii. Input: RequestID int, studentID int, and advisorID int {this advisor should be the one advising the student}
 - iv. Output: nothing

Z) View pending requests of specific advisor students

- i. Type: Stored Procedure
- ii. Name: Procedures_AdvisorViewPendingRequests
- iii. Input: Advisor ID int {this advisor should be the one advising the student}

iv. Output: Table of pending requests' details.

AA) Student Login using username and password.

- i. Type: Scalar Function
- ii. Name: FN_StudentLogin
- iii. Input: Student ID int, password varchar (40)
- iv. Output: Success bit

BB) Add Student mobile number(s)

- i. Type: Stored Procedure
- ii. Name: Procedures_StudentaddMobile
- iii. Input: StudentID int, mobile_number varchar (40)
- iv. Output: Nothing

CC) View available courses in the current semester

- i. Type: TVFunction
- ii. Name: FN_SemsterAvailableCourses
- iii. Input: semster_code varchar (40)
- iv. Output: Table of available courses within the semester

DD) Sending course's request

- i. Type: Stored Procedure
- ii. Name: Procedures_StudentSendingCourseRequest
- iii. Input: Student ID int, course ID int, type varchar (40), and comment varchar (40)

iv. Output: Nothing

EE) Sending extra credit hours' request

- i. Type: Stored Procedure
- ii. Name: Procedures_StudentSendingCHRequest
- iii. Input: Student ID int, credit hours int, type varchar (40), and comment varchar (40)
- iv. Output: Nothing

FF) View graduation plan along with the assigned courses

- i. Type: TVFunction
- ii. Name: FN_StudentViewGP
- iii. Input: student_ID int
- iv. Output: Table (Student Id, Student_name, graduation Plan Id, Course id, Course name, Semester code, expected graduation date, Semester credit hours, advisor id)

GG) Student view his first not paid installment deadline

- i. Type: Scalar Function
- ii. Name: FN_StudentUpcoming_installment
- iii. Input: StudentID int
- iv. Output: deadline date of first not paid installment

HH) View slots of certain course that is taught by a certain instructor

- i. Type: TVFunction
- ii. Name: FN_StudentViewSlot
- iii. Input: CourseID int, InstructorID int

- iv. Output: table of slots' details (Slot ID, location, time, day) with course name and Instructor name

II) Register for first makeup exam {refer to eligibility section (2.4.1) in Milestone 1}

- i. Type: Stored Procedure
- ii. Name: Procedures_StudentRegisterFirstMakeup
- iii. Input: StudentID int, courseID int, studentCurrent semester varchar (40)
- iv. Output: Nothing

JJ) Second makeup Eligibility Check {refer to eligibility section (2.4.1) in the description}

- i. Type: Scalar Function
- ii. Name: FN_StudentCheckSMEligibility
- iii. Input: CourseID int, Student ID int
- iv. Output: Eligible bit {0 → not eligible, 1 → eligible}

KK) Register for 2nd makeup exam {refer to eligibility section (2.4.1) in the description}

- i. Type: Stored Procedure
- ii. Name: Procedures_StudentRegisterSecondMakeup
- iii. Input: StudentID int, courseID int, Student Current Semester Varchar (40)
- iv. Output: Nothing

LL) View required courses

- i. Type: Stored Procedure
- ii. Name: Procedures_ViewRequiredCourses

- iii. Input: StudentID int, Current semester code Varchar (40)
- iv. Output: Table of the required courses' details.

MM) View optional courses

- i. Type: Stored Procedure
- ii. Name: Procedures_ViewOptionalCourse
- iii. Input: StudentID int, Current semester code Varchar (40)
- iv. Output: Table of the optional courses' details.

NN) View missing/remaining courses to specific student.

- i. Type: Stored Procedure
- ii. Name: Procedures_ViewMS
- iii. Input: StudentID int
- iv. Output: Table of missing courses' details

OO) Choose instructor for a certain selected course.

- i. Type: Stored Procedure
- ii. Name: Procedures_ChooseInstructor
- iii. Input: Student ID int, Instructor ID int, Course ID int
- iv. Output: nothing