Advanced SQL Project

Examination System

Supervised By

Eng/Sara

Team Members

N	Name	Work Description
1	Ahmed Mohamed Reda Salah	DDL-DML-Stored Procedures-Triggers-View
2	Aya Fathy Mohamed Ibrahim	ERD-Mapping, -**Stored Procedures:**SP_AssignInstructorToBranchIntake SP_ViewStudentGradesByInstructor &SP_CreateExamByInstructor **-Triggers:** TRG_EnforceExamTimeWindow **-Views:**v_EligibleStudentsForExam **-Users and logins**
3	Manar Abdelmnam Elmeslmany	Mapping-DDL-Stored Procedures-Triggers-View
4	Mohamed Fayez Abdelatif Mabrouk	ERD - Mapping - SP (SP_GetStudentExamResult)
5	Mahmoud Mohamed Ahmed Salama	ERD- Mapping- DDL, **- Stored Procedure** _CreateExamWithManualQuestions & AssignExamToStudent -**Triggers**-ValidateExamQuestions & One Correct Answer per Question. - **Views**_v_QuestionPool —> Instructor/Admin v_ExamQuestions —> Instructor/Admin

Abstract:

The Examination System Database project aims to design and implement a comprehensive database solution that manages examinations within a training environment using Microsoft SQL Server. The system provides a centralized question pool with multiple question types (multiple-choice, true/false, and text-based) that instructors can use to generate exams either manually or randomly. It ensures flexibility in exam creation while maintaining data integrity and consistency through constraints, triggers, and stored procedures.

The database manages courses, instructors, students, branches, tracks, and intakes, with proper relationships to support real-world training scenarios. Instructors can create and schedule exams for their courses, assign degrees to questions, and monitor student performance. Students can only access their assigned exams during the specified time, and the system automatically evaluates objective questions while offering review options for subjective answers.

From a technical perspective, the system employs optimized indexes, file groups, and backup strategies to enhance performance and reliability. SQL Server roles and permissions ensure secure access control, with distinct accounts for administrators, training managers, instructors, and students. Daily automated backups safeguard data, while views, functions, and stored procedures provide a user-friendly interface, eliminating the need for direct queries.

Introduction:

In modern educational and training environments, examinations play a vital role in assessing student knowledge, skills, and overall performance. Traditional examination methods, which rely heavily on manual question preparation, paper-based exams, and manual grading, are often time-consuming, prone to errors, and inefficient in managing large numbers of students and courses. Moreover, ensuring exam integrity, fair evaluation, and secure storage of results becomes increasingly challenging without a structured and automated system.

The Examination System Database project addresses these issues by providing a centralized, secure, and automated examination management solution built using Microsoft SQL Server. The system enables training managers, instructors, and students to interact seamlessly through well-defined roles and permissions. A question pool is maintained to allow instructors to generate exams either manually or randomly, with support for different question types such as multiple-choice, true/false, and text-based questions. The system automatically grades objective questions while offering instructors tools to review and grade subjective answers.

Problem Statement:

Current examination processes face multiple challenges:

1-Manual exam creation and grading lead to delays, inefficiency, and errors.

2-Lack of a centralized system for managing courses, instructors, students, and exams results in poor data organization.

3-Ensuring exam fairness and preventing unauthorized access to exam data is difficult without role-based security.

4-Backup and recovery processes are often neglected, leading to risks of data loss.

5-Scalability issues arise when handling multiple courses, branches, intakes, and large student groups.

The proposed Examination System Database seeks to solve these problems by offering an integrated, automated, and reliable platform that supports the entire examination lifecycle—from question creation to student result reporting ,while maintaining high performance, security, and data integrity.

System Requirements:

Functional Requirements:

1-User Management

2-Course & Training Management

3-Question Pool Management

4-Exam Creation & Scheduling

5-Exam Execution and Result Management

6-Security & Access Control

7-Backup & Recovery

Non-Functional Requirements:

Performance:

1-Use indexes for faster query execution and high performance.

2-Support efficient random question selection and result calculation.

Scalability:

1-Database design supports multiple courses, exams, instructors, and students without performance degradation.

Reliability & Availability:

1-We do daily automated backups to ensure data safety and quick recovery.

2-We ensure that exams and student answers are never lost during execution.

Security:

1-we use SQL Server authentication and permissions to enforce role-based access.

2-We protect student exam answers and grades from unauthorized access.

3-Implement constraints and triggers to preserve data integrity.

Maintainability:

1-Use naming conventions for all objects (tables, procedures, triggers, functions).

2-Centralize business logic in stored procedures/functions for easier updates.

Usability:

1-Provide views and stored procedures to simplify user interaction.

2-Ensure that instructors, managers, and students can perform tasks without writing SQL queries.

Data Integrity:

1-Use constraints, foreign keys, and triggers to maintain valid relationships between entities.

2-Prevent invalid data entry (e.g., exam marks exceeding course max degree).

Entity RelationShip Diagram(ERD):

These are the following entities:

1-Instructor : Ins_ID (PK) ,Name ,Email ,UserName.

2-Student: Student_ID (PK) ,Name ,Email ,UserName.

3-Exam : Exam_ID (PK) , Date,start time, end time, total time , Status , Corrective .

5-Exam_Corrective:Corrective_ID(PK) , Is_Completed , Start Time ,End Time , Corrective Date

6-Question pool :ID (PK) , Body , Degree , type.

7-Choices :Ch_ID (PK) , Body ,Is_Correct.

8-Track:Track_ID (PK), Track_Name.

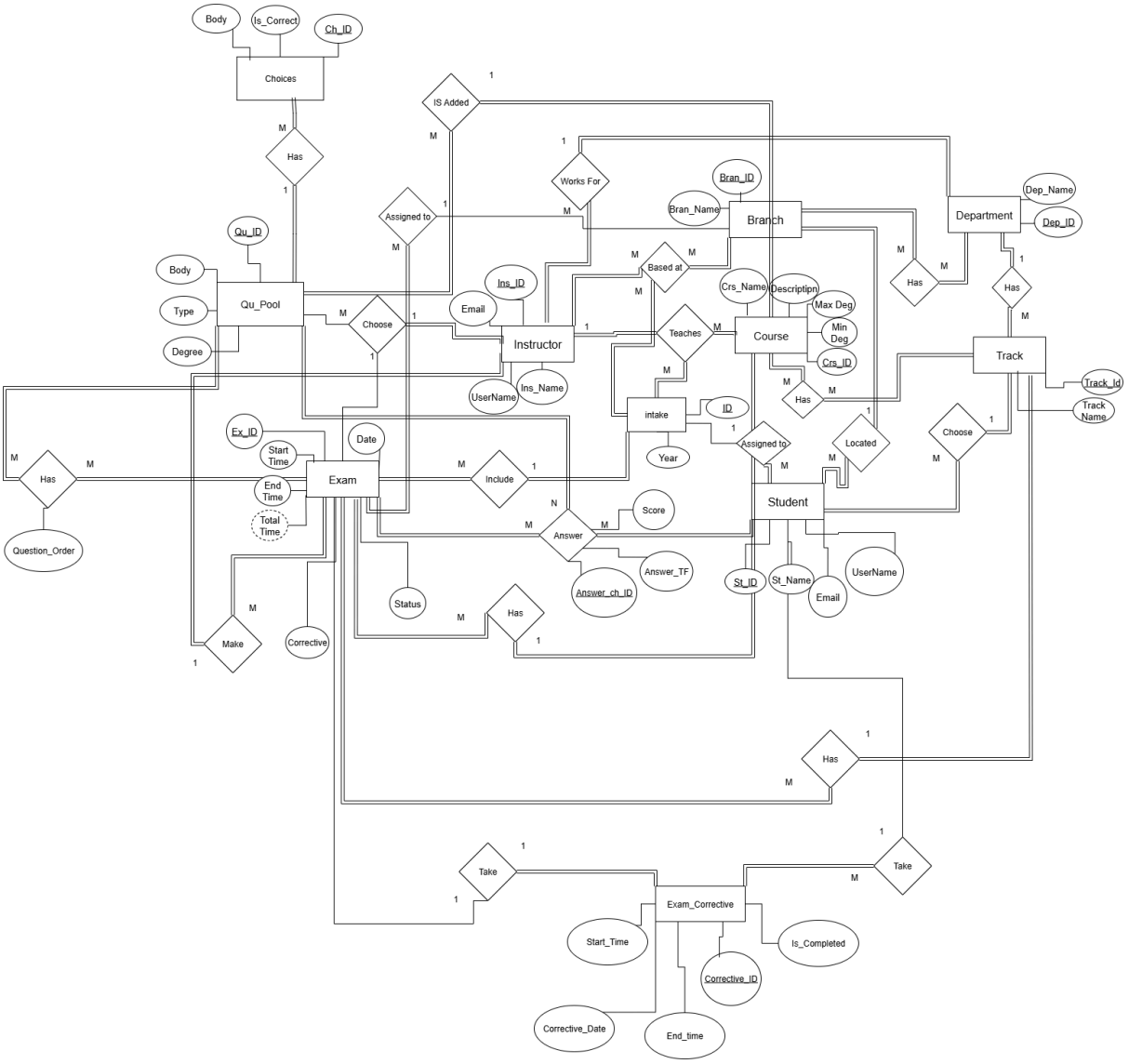
9-Course :Crs_Id (PK) ,crs_Name , Description , Max Degree , Min Degree.

10-Department :Dept_ID (PK) , Dept_Name.

11-Branch :Branch_ID (PK) ,Branch Name.

12-Intake : ID (PK) ,Year.

ERD:


Body
Is_Correct
Ch_ID
Choices
IS Added
Has
Works For
Bran_ID
Bran_Name
Branch
Dep_Name
Department
Dep_ID
Assigned to
Qu_ID
Based at
Body
Ins_ID
Crs_Name
Description
Max Deg
Min Deg
Crs_ID
Type
Qu_Pool
Email
Choose
Instructor
Teaches
Course
Has
Track
Degree
UserName
Ins_Name
intake
ID
Track_Id
Track Name
Ex_ID
Start Time
Date
Year
Assigned to
Located
Choose
Has
Exam
Include
End Time
Total Time
Score
Student
Question_Order
Answer
Answer_TF
Answer_ch_ID
St_ID
St_Name
Email
UserName
Status
Has
Make
Corrective
Has
Take
Take
Exam_Corrective
Start_Time
Is_Completed
Corrective_ID
Corrective_Date
End_time

Mapping:

Mapping and changing every entity to table , their attributes to column and breaking the relations into tables:

1-Department ↔ Track

- Dep_ID as a foreign key in Track table
- **Relationship:** 1 Department → many Tracks (1:N)

2-Instructor ↔ Department

- Dept_ID as a foreign key in Instructor table
- **Relationship:** 1 Department → many instructors (1:N)

3-Choices ↔ Question_Pool

- Qu_ID as a foreign key in choices table
- **Relationship:** 1 question → many choices(1:N)

4-Course ↔ Instructor ↔ Intake

- New table for Crs_ID, Ins_ID, Intake_ID
- **Relationship:** Ternary

5- Department ↔ Branch

- New table for Dept_ID, Bran_ID
- **Relationship:** Many Departments ↔ Many Branches (M:N)

6- Exam ↔ Branch, Course, Intake, Instructor, Track

- Bran_ID, Crs_ID, Intake_ID, Inst_ID, Track_ID as foreign keys in Exam table
- **Relationship:** 1 Branch, Course, Intake, Instructor, Track ↔ Many exams (1:N)

7- Exam ↔ Corrective

- Ex_ID as a foreign key in Exam_Corrective table
- **Relationship:** 1 Exam ↔ 1 corrective (May-Must)

8- Student↔ Corrective

- St_ID as a foreign key in Exam_Corrective table
- **Relationship:** 1 students → many correctives (1:N)

9-Exam ↔ Questions

- New table for Ex_ID QuP_ID
- **Relationship:** Many Exams ↔ Many Questions (M:N)

10- Instructor ↔ Exam ↔ Question Pool

- New table for Inst_ID, Ex_ID, QuP_Id
- **Relationship:** Ternary Relation

11- Intake ↔ Instructor ↔ Branch

- New table for intake_ID, Inst_ID, Bran_ID
- **Relationship:** Ternary

12- Question Pool ↔ Course

- Crs_ID as a foreign key in Qu_Pool table
- **Relationship:** 1 Course ↔ Many question

13- Student ↔ Exam ↔ Question_Pool

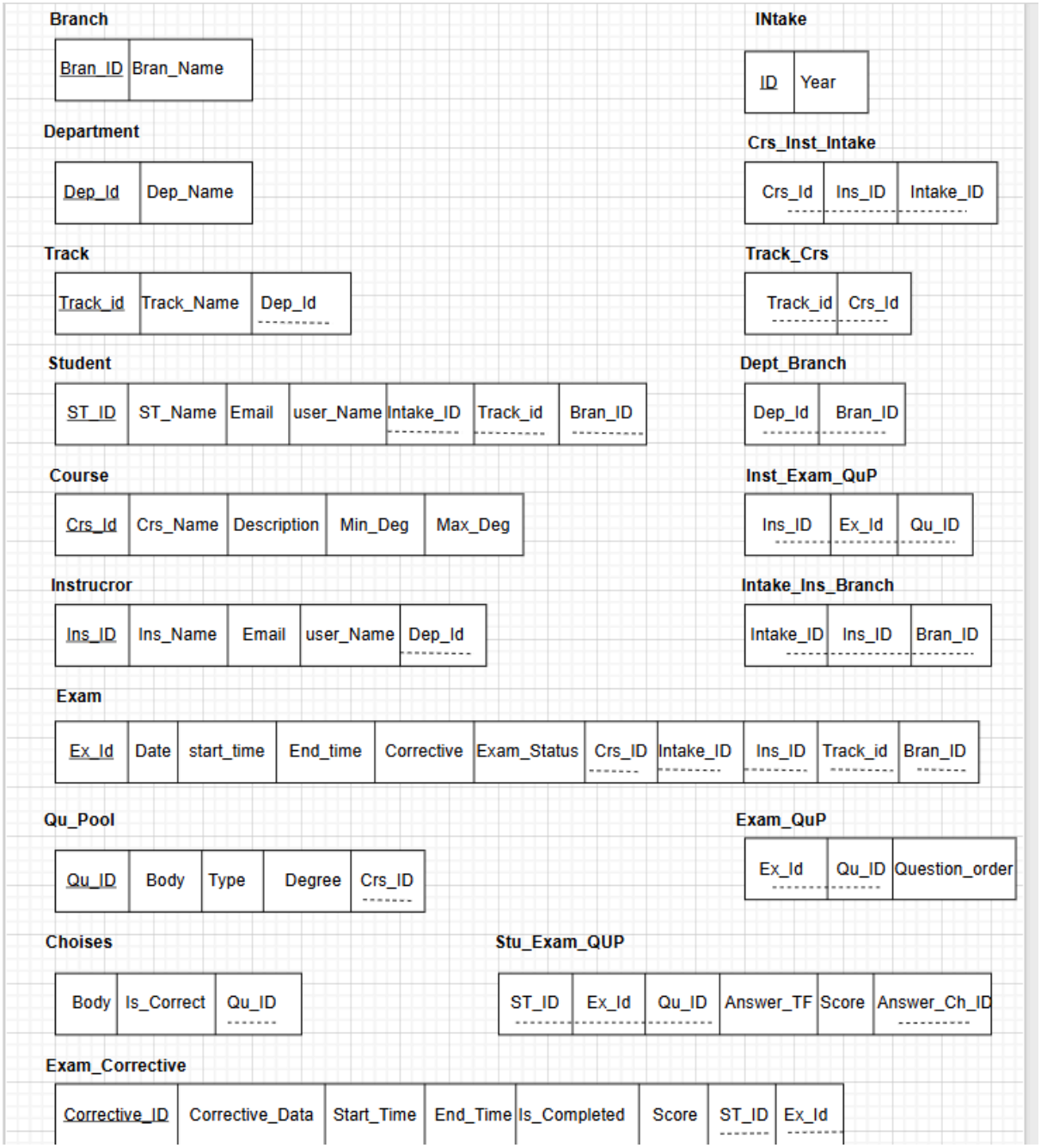
- New table for St_ID, Ex_ID, QuP_ID
- **Relationship:** Ternary

14- Student ↔ Intake, Track, Branch

- Intake_ID, Track_ID, Bran_ID as a foreign key in Student table
- **Relationship:** 1 intake, Track, Branch ↔ Many Students

15- Track ↔ Course

- New table for Track_ID, Crs_ID
- **Relationship:** Many tracks ↔ Many Courses


Branch
Bran_ID
Bran_Name
INtake
ID
Year
Department
Dep_Id
Dep_Name
Crs_Inst_Intake
Crs_Id
Ins_ID
Intake_ID
Track
Track_id
Track_Name
Dep_Id
Track_Crs
Track_id
Crs_Id
Student
ST_ID
ST_Name
Email
user_Name
Intake_ID
Track_id
Bran_ID
Dept_Branch
Dep_Id
Bran_ID
Course
Crs_Id
Crs_Name
Description
Min_Deg
Max_Deg
Inst_Exam_QuP
Ins_ID
Ex_Id
Qu_ID
Instrucror
Ins_ID
Ins_Name
Email
user_Name
Dep_Id
Intake_Ins_Branch
Intake_ID
Ins_ID
Bran_ID
Exam
Ex_Id
Date
start_time
End_time
Corrective
Exam_Status
Crs_ID
Intake_ID
Ins_ID
Track_id
Bran_ID
Qu_Pool
Qu_ID
Body
Type
Degree
Crs_ID
Exam_QuP
Ex_Id
Qu_ID
Question_order
Choises
Body
Is_Correct
Qu_ID
Stu_Exam_QUP
ST_ID
Ex_Id
Qu_ID
Answer_TF
Score
Answer_Ch_ID
Exam_Corrective
Corrective_ID
Corrective_Data
Start_Time
End_Time
Is_Completed
Score
ST_ID
Ex_Id

Data Definition Language(DDL):

1. Database Creation

- **Database Name:** `Examination_System_Data`
- **Filegroups:**
  - **Primary:** Holds main objects.
  - **EXAM_DATA_FG:** Dedicated for large exam-related data (Exam, Questions, Answers).
- **Log File:** Separate transaction log file.
- **File Settings:** Initial size 500MB, with automatic growth configured.

---

2. Core Entities and Tables

Branch

- Stores branch information (e.g., Cairo, Alexandria).

Department

- Stores academic departments (e.g., CS, IT).

Dept_Branch

- Junction table mapping departments to branches (many-to-many).

Track

- Defines tracks/programs (e.g., AI, Software Development).
- Linked to a department.

Intake

- Represents academic year/intake.

Student

- Stores student data (name, username, email).
- Linked to Intake, Track, and Branch.

Course

- Stores course details (name, description, min/max degrees).
- Validation: `Max_Deg > Min_Deg`.

Instructor

- Stores instructor data (name, username, email).

- Linked to a department.

---

3. Exam and Assessment

Exam

- Stores exam details (date, start/end time, corrective flag, status).
- Linked to Branch, Course, Intake, Instructor, and Track.
- Constraints:
  - End time > Start time.
  - Duration between 30–240 minutes.
  - Status limited to `Scheduled`, `In Progress`, `Completed`, `Cancelled`.

Question Pool (Qu_Pool)

- Repository of exam questions.
- Supports MCQ and True/False types.
- Default degree = 5.

Choices

- Stores choices for MCQs.
- Marks correct answer(s).

Exam_Questions

- Maps exams to selected questions with a defined order.

Student Exam Answers (Stu_Exam_QUP)

- Stores students' answers per exam question.
- Supports either MCQ choice or True/False (mutually exclusive).
- Stores score per question.

Exam_Corrective

- Stores corrective (make-up) exam details.
- Linked to original exam and student.
- Duration 30–240 minutes.
- Tracks completion status and score.

---

4. Mapping / Relationship Tables

Crs_Inst_Intake

- **Links courses, instructors, and intakes.**

Track_Crs

- **Links tracks to courses.**

Inst_Exam_QuP

- **Links instructors to exams and the questions they provided.**

Intake_Ins_Branch

- **Maps intakes, instructors, and branches.**

---

5. Key Features

- **Normalization: Proper many-to-many mappings using junction tables.**
- **Constraints: Data validation through CHECK constraints (exam duration, degrees, statuses).**
- **Scalability: Large exam/question/answer data placed in separate filegroup (`EXAM_DATA_FG`).**

- **Integrity: Extensive use of foreign keys and composite primary keys to enforce consistency.**

Stored Procedures:

Stored Procedure	Description
SP_AssignInstructorToBranchIntake	**Purpose**: Purpose of this stored procedure is to **manually assign instructor to specific branch in specific intake** , but it does it carefully by checking if the instructor is already assigned before to the specific branch and intack both. ——————————————————— 🔹 **Step-by-step Explanation:** **1) Input Parameters** ● `@InstructorID`: **The ID of the instructor you want to assign.** ● `@BranchID`: **The ID of the branch you want to assign an instructor to it.** ● `@IntakeID`: **The ID of the intake you want to assign an instructor to.** **2) Check if the instructor not assigned before** ● **It looks up the** `Intake_Ins_Branch` **table to make sure the instructor with** `@InstructorID` **not assigned before at a specific branch with** `@BranchID` **in specific intake with** `@IntakeID` **.**

	• **If yes, it prints message : *"Instructor is already assigned to this Branch and Intake."*** **3) Assign the instructor to the table:** **Adds the instructor , branch and the intake to `Intake_Ins_Branch` table**
SP_ViewStudentGradesByInstructor	**Purpose**: This stored procedure allows an **instructor** to view the **grades of all students** in a specific exam they created. It retrieves each student's total score, compares it to the **maximum course degree**, and determines whether the student has **passed** or **needs a corrective exam**. --- 🔹 **Step-by-step Explanation** **1) Input Parameters** • **`@InstructorID`: The ID of the instructor you will write.** • **`@ExamID`: The ID of the exam you want to see the grades of the student who took that exam.** **2) It ensures Only grades for exams assigned to the given instructor are displayed.** **3) Students' scores are calculated using aggregate function**

	(Sum)across `Stu_Exam_QUP`Table. **4) It checks If a student's total grade is less than 60% of the course maximum degree, message will be printed"Needs Corrective Exam", otherwise "Passed".** **5) Prevents null scores from affecting results by using `ISNULL` if there is null value it will replace it by Zero**
SP_CreateExamByInstructor	**Purpose**: This stored procedure allows an **instructor** to create an exam for a specific **course, branch, intake, and track**. It automatically validates exam conditions, assigns random questions from the course's question pool, and links them to both the **exam** and the **instructor**. The procedure also ensures that only **authorized instructors** can create exams and that exam rules (duration, corrective/normal) are respected. --- 🔹 **Step-by-step Explanation** **1) Input Parameters** • **`@InstructorID`: The ID of the instructor who wants to create the exam.** • **`@Crs_ID`: The ID of the course for which the exam is being created.** • **`@ExamDate`: The date on which the exam will be held.**

	• **`@StartTime`:The exam start time.** • **`@EndTime`The exam end time (must make exam exactly 60 or 120 minutes).** • **`@Branch_ID`The branch where the exam is assigned.** • **`@Intake_ID`The intake (batch/semester) for which the exam is created.** • **`@Track_ID`The track within the intake where this exam applies.** • **`@NumQuestions`The number of random questions to include in the exam.** • **`@IsCorrective`Flag to determine if the exam is normal (0) or corrective (1).**

	1)Instructor authorization: **Only instructors officially assigned to teach the course in the given intake can create exams.** **2)Track validation:** **The specified track must be linked to the chosen course.** **3)Exam duration validation:** **The exam duration must be exactly 1 hour (60 minutes) or 2 hours (120 minutes).** **4)One Normal exam per Intake:** **A course can have only one normal exam per intake. If another is needed, it must be created as a corrective exam.** **5)Randomized question selection:** **Questions for the exam are chosen randomly from the question pool of the course.** **6)Question–Instructor linkage:** **Ensures exam questions are tied back to the instructor who created the exam.** **7)Error handling:** **Any errors during exam creation are caught, and an error message is printed instead of failing silently.**

SP_Assign_Student_To_Course	**Purpose:** This procedure is designed to **manually assign a student to a specific course**, but it does so carefully by checking whether the student and course exist, and whether the course already belongs to the student's track. --- 🔹 **Step-by-step Explanation** **1) Input Parameters** • `@StudentID`: **The ID of the student you want to assign.** • `@CourseID`: **The ID of the course you want to assign.** **2) Check if the student exists** • **It looks up the `Student` table to make sure the student with `@StudentID` exists.** • **If not, it raises an error: *"Student does not exist."*** **3) Check if the course exists** • **It looks up the `Course` table to ensure the course with `@CourseID` exists.** • **If not, it raises an error: *"Course does not exist."*** **4) Get the student's track** • **Every student belongs to a `Track`.** • **The procedure retrieves the**

	`Track_ID` of the student. **5) Check if the course already belongs to that track** • **It searches the `Track_Crs` table (relationship between tracks and courses).** • **If the course is not in the student's track, it inserts a new row into `Track_Crs`, effectively adding that course to the track.** • **If the course is already there, it just prints a message: *"Course is already in student's track."*** **6) Error handling** • **The procedure is wrapped in a `TRY…CATCH` block.** • **If something unexpected goes wrong (e.g., constraint violation), it prints the error message instead of crashing.**

`SP_Assign_Students_To_Track`	**Purpose:** This procedure is used to **assign multiple students at once** to a specific track. Instead of assigning one by one, you can pass a comma-separated list of student IDs, and it updates all of them to the given track. This procedure is a **bulk assign tool that lets you quickly move one or multiple students into a specific track**, while ensuring the track and all students exist before updating.
`SP_AutoAssignCorrectiveExam`	**Purpose:** The purpose of this stored procedure is **to automatically assign corrective (make-up) exams to students who failed** their original exams. **It ensures that:** ● **Only students who scored below the minimum required degree get corrective exams.** ● **Corrective exams are scheduled automatically (7 days later, from 9 AM to 11 AM).** ● **No duplicate corrective exams are created for the same student and exam.**

`SP_assign_ins_crs_intake`	**Purpose:** **In specific intake, each instructor must be assigned to a specific course** **Steps:** ● **3 variables needed (@CourseID , @InstructorID, @Intake_id** ● **First: checking for existence of Courceid, instructorid and intake_id entered** ● **After that check that all of them have not entered together before** ● **Finally: Insertion commands**

`SP_examdetails_St`	**Purpose:** **To enter the students id and reach to their exam details (id, course name, Date, start time, End time and its status** **Steps:** ● **One variable needed StudentID** ● **First: Select columns needed from many tables (joins used)** ● **Ordering them by date** ● **Finally: check by enter any student ID**

Triggers:

`Triggercountdegrees`	**Purpose:** **To make sure that the count of degrees that instructor choose=100** **Action:** **Raise error that count of degrees <> 100**
`TRG_EnforceExamTimeWindow`	**Purpose:** **This trigger ensures that student exam answers inserted into the `Stu_Exam_QUP` table follow strict exam rules.** **It prevents invalid or unauthorized answer submissions, enforces the exam time window, and guarantees that students can only submit valid answers once per question.** **Action:** **Raise error "Cannot submit answers outside the exam time frame."**
`TR_Student_PreventDelete`	**Purpose:** **To prevent deletion of students who have exam records in the system, thereby maintaining referential integrity and preventing orphaned data.**

`TR_Stu_Exam_QUP_ValidateDegree`	**Purpose:** **To enforce a business rule that every question must be worth exactly 5 points in the student exam scoring system, ensuring consistent grading across all exams.**
`TR_Qu_Pool_PreventDelete`	**Purpose:** **The purpose of the trigger `TR_Qu_Pool_PreventDelete` is to control and protect deletion of questions from the `Qu_Pool` table.** **This trigger prevents accidental deletion of questions that are already used in exams, while allowing safe deletion of unused questions and their choices.**
`TR_Exam_Corrective_ValidateDate`	**Purpose:** **The purpose of the trigger `TR_Exam_Corrective_ValidateDate` is to enforce a business rule on corrective exams so that their scheduled date is always after the original exam date** **This trigger ensures data integrity by making sure that a corrective exam can only be scheduled after its corresponding original exam date.**

Views:

`v_EligibleStudentsForExam`	**Purpose**: This view is designed to provide a **list of students eligible to take an exam**, along with details about the exam, course, intake, track, and branch. It also checks whether a student has **already taken the exam** and marks their eligibility status accordingly.
`V_StudentAnswers`	**Purpose:** **To views the name of the students, question body and their answers**
v_ExamSchedule	**Purpose:** **To display detailed information about scheduled exams.** **Includes exam date, time, duration (calculated in minutes), branch, track, instructor, and whether it's corrective.** **Useful as an exam timetable/report for students, instructors, and admins.**

v_ExamChoices	**Purpose:** **To show all choices for each question in an exam.** • **Includes choice text, whether it's correct, and course name.** • **Useful for validating exams (e.g., checking correct/incorrect choices).**
v_TeachingAssignments	**Purpose:** **To show which courses are assigned to each instructor.** • **Includes instructor details, course, department, branch, track, intake, and total courses assigned.** • **Useful for workload management and academic planning.**
v_CorrectiveExams	**Purpose:** **To monitor corrective (make-up) exams.** • **Shows corrective exam details, student info, original exam, original score, corrective score, result (passed/failed), and how many days after the original exam it was scheduled.**

	• **Useful for tracking student improvements and compliance with exam policies.**
v_ExamResultsByStudent	**Purpose:** **To calculate the final result of each student per exam.** • **Summarizes total score, pass/fail status, number of questions, average score per question.** • **Compares student's score with Min_Degree (passing threshold).** • **Useful for exam reports, certificates, and analytics.**
v_ExamQuestions	**Purpose:** **To list the questions assigned to each exam.** • **Displays exam ID, course, question details (text, type, degree, order), and instructor who created it.** • **Helps in reviewing exam contents and auditing question assignments.**

v_QuestionPool	**Purpose:** **To display detailed information about scheduled exams.** ● **Includes exam date, time, duration (calculated in minutes), branch, track, instructor, and whether it's corrective.** ● **Useful as an exam timetable/report for students, instructors, and admins.**