VIEWS

- **v_display_questions_Mcq**

  Shows all MCQ questions with choices and correct answer.

- **View: v_display_questions_TF**

  Shows all True/False questions with correct answer.

- **View: v_display_questions_text**
  Shows all Text-based questions with correct answer.
- **View: v_display_exam**
  Displays all exams.
- **View: v_display_exam_questions_mcq**
  Shows all MCQ questions linked to exams with their degree.
- **View: v_display_exam_questions_tf**
  Shows all True/False questions linked to exams with their degree.
- **View: v_display_exam_questions_text**

  Shows all Text questions linked to exams with their degree.

- **vw_AllInstructor**
  Purpose: Displays all instructors along with the courses they are assigned to.
  Details: Joins the Instructor_Course table with Course and Instructor to show:
  Instructor ID ,Instructor full name ,Course name ,Course description
- **vw_AllStudent**
  Purpose: Displays all students along with their intake, branch, and track information.
  Details: Joins the Student table with Branch and Track to show:
  Student ID ,Full name (first + last),Intake number,Phone number,Branch name ,Track name

- **w_CoursePerformance**
  is designed to analyze and summarize the performance of students in each course. It joins the Course, Exam, Student, and Student_Answer tables to calculate key performance metrics. For every course, it computes the average score, as well as the minimum and maximum scores, using a custom function (fn_GetTotalGrade) to determine each student's total exam grade. Additionally, it counts the number of distinct students who participated in the exams of that course. This view provides instructors and administrators with a clear performance overview, helping in evaluation, monitoring, and decision-making
- **The view vw_PendingTextAnswers**
  is created to monitor and manage text-based exam answers that are still awaiting instructor evaluation. It retrieves all student responses for questions where the question type is "Text" and the mark has not yet been assigned. The view shows key details including the student's ID, full name, exam ID, question ID, question text, and the submitted answer. This provides instructors with a centralized list of pending answers that require manual review and grading. By using this view, the evaluation process becomes more efficient, ensuring no student's text answers are overlooked or left ungraded.
- **The view vw_PassRatePerCourse**
  is designed to provide an analytical report of student performance across all courses. It calculates the total number of students enrolled in each course, the number of students who passed (with a total grade of 50 or higher), and the pass rate percentage. The pass rate is computed by dividing the number of passed students by the total number of students, then multiplying by 100 to express it as a percentage. This view helps administrators and instructors evaluate course outcomes, compare success rates between courses, and identify areas where improvements may be needed in teaching or assessment.

V_Student_Exam

This view V_Student_Exam is created to show the upcoming exam schedule for students.

Data Returned:

Student ID, first name, last name

Course ID and Exam ID

Exam date, start time, and end time

Instructor ID

Logic:

Joins Student, Exam, and Exam_Student tables to connect students with their exams.

Filters exams to only show those with a date on or after today (future or current exams).

In short: It lists all future exams for each student, including course and instructor details.

- **V_Student_Answers**

This view V_Student_Answers is designed to display the answers students submitted for their exams.

Data Returned:

Student ID, first name, last name
Exam ID and exam date
Question ID and question text
The student's submitted answer

Logic:

Joins Student, Exam, Student_Answer, and Question tables.
Links each student with their exam and the specific questions they answered.

In short: It provides a detailed record of each student's submitted answers for every exam question.

- **V_Student_Results**

This view V_Student_Results is designed to show students' results for each exam along with their pass/fail status.

Data Returned:

Exam ID and total exam degree (calculated from all questions).

Student ID and the student's total earned marks in that exam.

Pass/Fail status based on whether the student's marks are at least 50% of the exam total.

Logic:

Aggregates exam total marks from Exam_Question.

Aggregates each student's marks from Student_Answer.

Joins both sets to compare and determine the result.

In short: It calculates each student's exam score and classifies them as Pass or Fail depending on whether they reach half of the exam's total marks.

Stored procedures

- **procedure add_question_sp :**
  This procedure adds a new question into the question pool.
  MCQ : Inserts the question along with four choices and the correct answer into the MCQ table.
  True/False: inserts "True" and "False" as choices in the Q_T_F table and correct answer .
  Text: Stores the correct answer in the Text_Question table.

- **procedure add_exam_sp :**
  This procedure allows instructors to create an exam. It saves exam details such as type, start time, end time, date, instructor, intake number, and course.
  Ensures the exam is created only for courses taught by the instructor.

  Checks that the end time is greater than the start time.

- **add questions into exam :**
  This procedure links a question to a specific exam with a given degree/weight.
  Ensures the exam exists.
  Prevents adding the same question more than once to the exam.
  Ensures the question belongs to the same course as the exam.
  Validates that the question degree is greater than zero.
  Ensures that the total assigned degrees for the exam do not exceed the maximum course degree.

- **SP_Grade_Exam_Auto :**
  This stored procedure automatically grades student answers for a specific exam (@Exam_Id) by updating the st_Mark column in the Student_Answer table.
  It performs grading in two phases:

  MCQ Questions

  Compares each student's answer (st_Answer) to the correct choice in the MCQ table.
  If the answer is correct → assigns the question's weight (quest_degree) from Exam_Question.
  If incorrect → assigns a score of **0**.

  True/False Questions

  Compares each student's answer (st_Answer) to the correct choice in the Q_T_F table.
  If the answer is correct → assigns the question's weight (quest_degree).
  If incorrect → assigns a score of **0**.
- **SP_Record_Student_Answer:**
  This procedure records or updates a student's answer for a given exam, ensuring validity of answer and exam time restrictions.

Exam Validation

Retrieves exam start time, end time, and date from the Exam table.
Gets current system time (GETDATE()).

Question Type Validation

If MCQ: Ensures the submitted answer matches one of the available choices (Choice1–Choice4).
If TrueFalse: Ensures the answer is either True or False.
If Text Question: No restriction.

Time Restriction

Checks if the current time (@Now) is within the exam date (E_date) and between exam StartTime and EndTime.
If the time is valid, the answer can be recorded; otherwise, raises an error.

Upsert Logic (Insert or Update Answer)

If a record for the student's answer to this question already exists in Student_Answer, it updates the answer.
If not, it inserts a new record with:
A new ans_id (calculated as MAX(ans_id)+1).
The given Exam_Id, Student_Id, Question_Id, and st_Answer.
st_Mark = NULL (to be graded later by another procedure such as SP_Grade_Exam_Auto).

- **SP_Get_Exam_Question_Random :**
  This procedure automatically selects and assigns random questions from a course's question bank into a specific exam, while enforcing exam rules.
  Retrieve Exam Information
  - Gets Course_Id and INS_Id (instructor) from the Exam table.
  - If no matching exam is found → raises an error and exits.

  Calculate Current Exam Marks
  - Sums up the total marks (quest_degree) already assigned to the exam in Exam_Question.
  - Stores result in @CurrentTotal.

  Select Random Questions
  - Uses a CTE (RandomQuestions) to randomly choose TOP(@NumQuestions) questions from the Question table for the exam's course.

- Ensures no duplicate questions are inserted by excluding those already present in Exam_Question.
- Assigns default marks (quest_degree) based on question type:
  - MCQ → 2 marks
  - Q_T_F (True/False) → 2 marks
  - Text → 5 marks

Insert into Exam_Question

- Inserts the randomly chosen questions into Exam_Question for the given exam.
- Uses CROSS APPLY to calculate the total new marks being inserted.
- Only inserts if (@CurrentTotal + TotalNewDegree) ≤ 100 to ensure exam total marks don't exceed 100.

- **sp_AddInstructor**
  is designed to add a new instructor along with their corresponding user account in a secure and consistent way. It accepts details such as the instructor's ID, name, email, phone, username, and password, and ensures both records are saved inside a single transaction. First, it inserts login credentials into the Users table, using the provided User_Id. Then, it inserts the instructor's personal and professional information into the Instructor table, linking it to the same User_Id. By wrapping these steps in a transaction, the procedure guarantees data integrity, rolling back changes if any error occurs.

- **sp_UpdateStudent i**

  s designed to update the details of an existing student in the database. It takes the student's ID along with updated values for first name, last name, email, phone number, and user ID as parameters. Before applying changes, it checks if the student exists using a validation function (f*n_CheckStudentExists*). If the student does not exist, the procedure stops and prints a message. If the student exists, it updates the record with the new information. This approach ensures data accuracy and prevents invalid updates, while also providing clear feedback on the operation's outcome.
- **sp_CalculateFinalResult**

  is created to compute and display the final result for a specific student. It accepts the student's ID as input and first verifies whether the student exists in the system. If the student is found, it sums up the marks from the Student_Answer table to calculate the total grade. In cases where no grades are available, the final result is set to zero. The procedure then outputs the student's ID, first name, last name, and calculated final result. This ensures accurate performance tracking and provides clear feedback while handling errors gracefully.
- **sp_AssignExamToStudents**

  is used to assign a specific exam to exactly three students at once. It takes the exam ID and three student IDs as input parameters. Inside the procedure, each student is inserted into the Exam_Student table with the given exam ID. Once all three assignments are completed, a confirmation message is displayed. Finally, the procedure retrieves and shows the basic details (student ID, first name, and last name) of the three assigned students. This ensures that exam distribution is quick, reliable, and easy to verify, while handling errors if any issue occurs.

- **AddCourse**
  Adds a new course to the Course table.
  Validates that the course name is not empty or duplicated, and that degree values are valid (Max > 0, Min ≥ 0, Max ≥ Min).
  If validations pass, inserts the course and returns the new Course_Id.
- **UpdateCourse**
  Updates an existing course in the Course table.
  Validates that the Course_ID is a positive number and exists, the course name is not empty, and degree values are valid.
  If all checks pass, it updates the course details and confirms with a success message.
- **AssignInstructorToCourse**
  Assigns an instructor to a specific course in a specific intake.
  Validates that all IDs (Course_ID, Instructor_ID, Intake_ID) are positive and exist in their respective tables.
  Ensures the instructor is not already assigned to that course in that intake, then inserts the assignment.
- **SearchExams**
  Searches exams in the Exam table based on optional filters: Exam ID, type, course, instructor, intake, date, and allowed option.
  Validates IDs to be positive numbers and ensures the exam date is in a valid format before filtering.
- **Addbranch**
  It's a stored procedure : Adds a new branch while validating the ID, branch name, and assigned training manager.
- **Addtrack**
  It's a stored procedure : Adds a new track within a department, ensuring data integrity and valid relationships.
- **Addintake**
  It's a stored procedure : Registers a new intake (batch/year) with correct start and end dates.

- **proc_add_student**

This stored procedure `proc_add_student` is designed to add a new student into the system.

Validations:

Checks that the given Branch, Track, and Intake exist.

Ensures the Student ID doesn't already exist.

Transaction Handling:

Starts a transaction.

If the provided User ID already exists in the `users` table → it prevents adding the student.

Otherwise, it inserts a new record into `users` and then inserts the student's details into `Student`.

In short: It safely inserts a new student along with a linked user account, only if all references are valid and IDs are unique.

- **proc_update_student**

This stored procedure proc_update_student is built to edit an existing student's details.

Validations:

Ensures the provided Intake, Branch, and Track exist in their parent tables.

Transaction Handling:

Uses a transaction with TRY…CATCH to guarantee safe execution.

Updates the student's personal and academic details (name, email, phone, intake, branch, track).

Feedback:
If no record matches the given Student ID, it prints a message saying no update occurred.
otherwise, it prints a success message.

In short: It updates a student's information after validating references, with safe transaction control and error handling

- **proc_delete_student_record**

This stored procedure proc_delete_student_record is designed to remove a student from the system.

Validation:

Checks if the given Student ID exists; if not, it throws an error.

Transaction Handling:

Begins a transaction to ensure safe deletion.

Deletes the student record from the Student table.

Prints a confirmation message if deletion succeeds.

Error Handling:

Uses TRY...CATCH with ROLLBACK to undo changes if an error occurs.

In short: It safely deletes a student record after confirming the ID exists, with transaction control and rollback on errors

- **proc_search_student**

This stored procedure proc_search_student is used to find students based on different optional criteria.

Functionality:

Retrieves student details along with related intake, branch, and track information.

Search Filters (all optional):

First name

Last name

Intake number

Branch ID

Track ID

Flexible Query:

If a parameter is NULL, that filter is ignored.

Allows partial searches (e.g., by only intake, or by name + branch).

In short: It searches and displays student details with their intake, branch, and track, based on any combination of the provided search criteria.

Triggers

- **trigger trg_check_exam_time on Student_Answe：**
  This trigger validates exam timing before allowing a student to insert an answer.
  Checks if the exam date matches today.
  Ensures the current time is greater than or equal to the exam's start time.
  Ensures the current time is less than or equal to the exam's end time.
  Behavior:
  If any condition fails , raises an error and rolls back the insert.
  If all conditions pass ， allows the insert into Student_Answer

- **Trg_PrvDelete_Course_WithExams**

This trigger Trg_PrvDelete_Course_WithExams is created to stop deleting courses that already have exams assigned to them.

When it runs:

Fires after a DELETE on the Course table.

Logic:

Checks if the deleted course(s) are linked to any exams in the Exam table.

If yes → it rolls back the deletion and shows a message "You can not delete a course exams".

In short: It ensures that a course cannot be removed from the system if there are exams related to it.

- **The trigger trg_BackupOnTrainingManagerLogon**
  is a server-level logon trigger designed to automate database backups based on a specific user's activity. It activates whenever a user logs into the SQL Server. If the user logging in is TrainingManager and the login occurs exactly at 9 AM, the trigger initiates a database backup operation. The backup is stored in a specified path with a clear name, making it easy to track. This ensures that critical data is backed up daily without requiring manual intervention, enhancing reliability and security. It also ties accountability to the Training Manager's login schedule