ITI Examination System

# System Overview:

**ITI Examination system is SQL server-based database designed to manage educational entities, including:**

**Student Instructor Course**

**Exam Question**

## Key Features:

**. Performance Optimization: Uses user definition filegroups for better logical data separation**

**. Comprehensive Functionality: Includes stored procedures, functions, views, rules to handle exam lifecycle from scratch to ending.**

**. This Entity-Relationship Diagram (ERD) outlines the core structure of an e-learning or educational management system, defining the key entities and their relationships. The system is designed to support user management, course administration, student interactions, and assessment handling in an online learning environment.**

**A computer screen shot of a diagram

AI-generated content may be incorrect.**

## Project Outcome:

1. **User Management:**

**. Different permission levels required for each user type**

**. Profile management for all user types**

1. **Student Management:**

**. Tracking of student progress and activities**

**. Student performance monitoring capabilities**

1. **Course Management:**

**. Course enrollment system, content organization**

**. Prerequisite and dependency management between courses**

**. Course scheduling functionality**

1. **Question/Assessment Requirements:**

**. Question bank management**

**. Question creation and categorization**

**. Assessment/exam creation**

**. Grading and evaluation system**

**. Answer submissions and tracking**

1. **Manager Requirements:**

**. User management interface**

**. reporting and analytics**

**. system configuration**

**. content approval workflow**

# System Requirements:

**. System must support multiple user types [ manager – student – instructor], User authentication and authorization system needed**

**. Each user has first name, last name, phone, address, role, email, gender, password**

**. when Student register and has profile management, Student must have courses and enrolled in track.**

**. Student must take course which has knowledge about course name and description of course content, maximum and minimum degree of this course.**

**. Student must belong to track to begin his lifecycle in intake and choose track by knowing his name**

**. student information must added into system in first time by branch manager.**

**. Instructor must have one or many courses to teach which can pick exam for students.**

**. Every Track must have own courses which is under supervised by department.**

**. After Course is finished student must have exam that includes questions for each course.**

**. All Exams have same shape of questions type which are [ MCQ – True/False – Text ], Of course question answers is stored and text question answers have different nature which is defined by keywords in student answers and at least question mark.**

**. Exam Cycle depend on type of exam that student should take main exam and if he/she fail in it student transferred directly into corrective exam [ Last Chance ], Time is one of the most important terms in exam that calculate start and end of exam to determine duration time to decide student submission.**

**. Exam degree stored to compare it with student degree to determine if he/she passed or failed.**

**. After student complete exam it take student degree to define his status and take him/her to next process.**

**. Manager must manage [ Branch – Department ] and add new Intake, Manager control on student Information.**

# Filegroups And Files:

**The database uses four custom filegroups, each targeting a functional area of the system:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **File group** | **File name** | **Path** | **Size** | **Max Size** | **Growth** |
| **FG\_Users** | **UsersDataFile** | **E:\Power pi iti\Advanced sql\Final\_project\ExamProject\ExamProject\UsersData.ndf** | **10MB** | **100MB** | **10MB** |
| **FG\_Courses** | **CoursesDataFile** | **E:\Power pi iti\Advanced sql\Final\_project\ExamProject\ExamProject\CoursesData.ndf** | **10MB** | **100MB** | **10MB** |
| **FG\_Admin** | **AdminDataFile** | **E:\Power pi iti\Advanced sql\Final\_project\ExamProject\ExamProject\AdminData.ndf** | **10MB** | **100MB** | **10MB** |

# Tables:

|  |  |  |
| --- | --- | --- |
| **No.** | **Column name** | **Data Type** |
| **1.User Table** | | |
| **1** | **User\_id** | **Int (PK)** |
| **2** | **First\_name** | **Varchar (40)** |
| **3** | **Last\_name** | **Varchar (40)** |
| **4** | **Email** | **Varchar (40)** |
| **5** | **Age** | **Int** |
| **6** | **Gender** | **Varchar (6)** |
| **7** | **Role** | **Varchar (20)** |
| **9** | **Address** | **Varchar (50)** |
| **10** | **phone** | **Varchar (15)** |
| **2.Manager Table** | | |
| **1** | **Manager\_id** | **Int (PK)** |
| **2** | **User\_id** | **Int (FK)** |
| **3.Student Table** | | |
| **1** | **Student\_id** | **Int (PK)** |
| **2** | **User\_id** | **Int (FK)** |
| **4.Instructor Table** | | |
| **1** | **Instructor\_id** | **Int (PK)** |
| **2** | **User\_id** | **Int (FK)** |
| **5.Course Table** | | |
| **1** | **Course\_id** | **Int (PK)** |
| **2** | **Course\_name** | **Varchar (50)** |
| **3** | **Description** | **Varchar (300)** |
| **4** | **Max\_deg** | **Int** |
| **5** | **Min\_deg** | **Int** |
| **6** | **Text\_q** | **Varchar (500)** |
| **6.Question Table** | | |
| **1** | **Question\_id** | **Int (PK)** |
| **2** | **Exam\_id** | **Int (FK)** |
| **3** | **Keywords\_Text\_Q** | **Nvarchar (200)** |
| **4** | **Marks\_Q** | **Nvarchar (300)** |
| **5** | **Correct\_Answer\_Q** | **Nvarchar (300)** |
| **6** | **MCQ\_Q** | **Nvarchar (200)** |
| **7** | **Ture\_false\_q** | **Nvarchar (200)** |
| **8** | **Text\_Q** | **Nvarchar (200)** |
| **7.Exam Table** | | |
| **1** | **Exam\_id** | **Int (PK)** |
| **2** | **Instructor\_id** | **Int (FK)** |
| **3** | **Course\_id** | **Int (FK)** |
| **4** | **Student\_id** | **Int (FK)** |
| **5** | **Main\_ex** | **Varchar (500)** |
| **6** | **Startime\_Ex** | **Datetime** |
| **7** | **Endtime\_Ex** | **Datetime** |
| **8** | **Corrective\_Ex** | **Varchar (300)** |
| **9** | **t\_marks\_Ex** | **Varchar (300)** |
| **10** | **Min\_deg\_Ex** | **Int** |
| **11** | **Max\_deg\_ex** | **Int** |
| **12** | **Degree** | **Float** |
| **8.Exam\_Result Table** | | |
| **1** | **St\_result** | **Int (PK)** |
| **2** | **Exam\_id** | **Int (FK)** |
| **3** | **Score** | **Varchar (300)** |
| **4** | **completed** | **Varchar (300)** |
| **5** | **Pending** | **Varchar (100)** |
| **9.Branch Table** | | |
| **1** | **Branch\_id** | **Int (PK)** |
| **2** | **Manager\_id** | **Int (FK)** |
| **3** | **Branch\_name** | **Varchar (30)** |
| **10.Intake Table** | | |
| **1** | **Intake\_id** | **Int (PK)** |
| **2** | **Manager\_id** | **Int (FK)** |
| **3** | **Branch\_id** | **Int (FK)** |
| **4** | **Intake\_name** | **Varchar (500** |
| **11.Departement Table** | | |
| **1** | **Dept\_id** | **Int (PK)** |
| **2** | **Manager\_id** | **Int (FK)** |
| **3** | **Dept\_name** | **Varchar (30)** |
| **12. Track** | | |
| **1** | **Track\_id** | **Int (PK)** |
| **2** | **Intake\_id** | **Int (FK)** |
| **3** | **Track\_name** | **Varchar (40)** |

# Views:

|  |  |  |
| --- | --- | --- |
| **num** | **name** | **Description** |
| **Course Table** | | |
| **1** | **CourseDetails\_v** | **Number of Exams in each course** |
| **Exam Table** | | |
| **1** | **Exam\_Students** | **Exam ID, Exam result, Student Full name** |
| **2** | **ExamDetails** | **Details of exam join with track, student, exam result, course** |
| **3** | **vw\_StudentExamDegrees** | **Student Exam score** |
| **Question Table** | | |
| **1** | **vw\_Question** | **Question for Instructors** |
| **Student Table** | | |
| **1** | **View\_Student\_UpcomingExams** | **Course Exam with Exam duration** |
| **2** | **Vw\_studentExams** | **Student exams result related to course name and exam time** |
| **3** | **Vw\_studentinfo** | **Student full name and student track name** |
| **4** | **Student\_course\_details** | **Student full name, course name with student score** |
| **Manager Table** | | |
| **1** | **TrainingManager\_details** | **Manager authority on branch and intake** |
| **Instructor Table** | | |
| **1** | **instructor\_course\_students** | **view lists all students in courses taught by each instructor.**  **It includes instructor information, student information, and course details.** |

# Functions:

|  |  |  |
| --- | --- | --- |
| **num** | **name** | **Description** |
| **Branch Table** | | |
| **1** | **fn\_GetManagerInfoByBranch** | **This function returns the name and email of the manager responsible for a specific branch**  **by joining the Branch, Manager, and User tables, filtered by the given branch name.** |
| **Course Table** | | |
| **1** | **fn\_GetCoursesByInstructor** | **This function returns a distinct list of course names assigned to a specific instructor**  **by joining the Exam and Course tables using the instructor's ID.** |
| **Department Table** | | |
| **1** | **fn\_DepartmentsWithManagers** | **This function returns to a list of all departments along with the full name of their assigned**  **managers by joining the Department, Manager, and User tables.** |
| **Exam Table** | | |
| **1** | **fn\_TotalExamsByStudent** | **This function returns the total number of exams taken by a specific student**  **by counting the rows in the Exam table where the student\_id matches the input.** |
| **Instructor Table** | | |
| **1** | **CountInstructorExams** | **Function to Count Exams Created by an Instructor** |
| **2** | **GetInstructorCourses** | **Returns a distinct list of courses assigned to a specific instructor**  **by joining the Exam and Course tables.** |
| **3** | **GetInstructorStudents** | **Retrieves a distinct list of students who took exams created by a specific instructor**  **by joining Exam, Student, and User tables.** |
| **4** | **GetInstructorPassRate** | **Calculates the percentage of exams passed (degree ≥ min\_deg\_Ex)**  **by students for a specific instructor.** |
| **5** | **GetInstructorSummary** | **Returns a summary per course taught by the instructor including**  **course ID, name, total exams, and average score.** |
| **Student Table** | | |
| **1** | **fn\_GetStudentFinalGrade** | **Calculates the final grade of a student in a specific course by summing**  **all exam scores for that student and course.** |
| **2** | **fn\_IsStudentPassed** | **Determines if a student passed a specific course by comparing the total**  **earned grade to the course's minimum passing degree.** |
| **3** | **fn\_GetStudentTrackName** | **Retrieves the track name of a student by joining the Student and**  **Track tables using the student’s track\_id.** |
| **Track Table** | | |
| **1** | **fn\_CountStudentsPerTrack** | **Returns the number of students in each track by performing a**  **LEFT JOIN between Track and Student tables and grouping by track name.** |

# Stored Procedures:

|  |  |  |
| --- | --- | --- |
| **num** | **name** | **Description** |
| **student Table** | | |
| **1** | **UpdateStudentInfo** | **Update student info to new track** |
| **2** | **GetTopStudents** | **Return top student by number** |
| **3** | **getstudentcountbytrack** | **Return students by track** |
| **4** | **GetStudentFullReport** | **Return full report about student Grade , courses, track** |
| **5** | **add\_student** | **Add student by manager** |
| **Instructor table** | | |
| **1** | **AddNewInstructor** | **Add new instructor** |
| **Courses table** | | |
| **1** | **GetCoursesByTrack** | **Get all courses by track number** |
| **2** | **GetCourseGrades** | **Get Coures Grade** |
| **Exam Table** | | |
| **1** | **AddExam** | **Add new exam by instructor** |

# Trigger:

|  |  |  |
| --- | --- | --- |
| **num** | **name** | **Description** |
| **1** | **CheckMaxDegree\_TR** | **Rules enforce exam’s total marks don’t exceed the maximum allowed for its associated course.** |
| **2** | **trg\_LogExamCreation** | **Provide notification when a new exam record is created.** |
| **3** | **TRG\_Prevent\_Answer\_After\_Time** | **Prevent submitting exam answers after exam has ended.** |
| **4** | **trg\_PreventStudentDelete** | **Prevent students’ deletion who have existing exam answers which has several logical issues.** |
| **5** | **trg\_AfterStudentAnswerInsert** | **Designed to execute stored procedure after student answer is inserted.** |

# Index:

|  |  |  |  |
| --- | --- | --- | --- |
| **num** | **name** | **Type** | **Description** |
| **Exam Table** | | | |
| **1** | **IX\_StudentExam\_ExamID\_StdID** | **clustered** | **Index Exam\_ID, Student\_ID** |
| **Question Table** | | | |
| **1** | **IX\_QuestionPick\_Exam\_Inst** | **clustered** | **Index Exam\_ID, Question\_ID** |
| **2** | **IX\_QuestionPool\_ID\_Answer** | **Non clustered** | **Index Correct\_Answer\_Q** |
| **Course Student Table** | | | |
| **1** | **IX\_StudentCourse\_CourseID** | **clustered** | **Index Course\_ID** |
| **Student Table** | | | |
| **1** | **IX\_Student\_UserID** | **clustered** | **Index User\_ID** |
| **Exam Table** | | | |
| **1** | **IX\_Exam\_CourseID** | **clustered** | **Index Course\_ID** |
| **2** | **IX\_StudentExam\_StdID** | **clustered** | **Index Student\_ID** |
| **3** | **IX\_StudentExam\_ExamID** | **clustered** | **Index Exam\_ID** |
| **Course Table** | | | |
| **1** | **IX\_Course\_Name** | **Non clustered** | **Index Course\_name** |
| **User Table** | | | |
| **1** | **IX\_Student\_Name** | **Non clustered** | **Index First\_name, Last\_name** |

# Rule:

|  |  |  |
| --- | --- | --- |
| **num** | **name** | **Description** |
| **1** | **role\_rule** | **Check user position not out of role array** |
| **2** | **email\_rule** | **Basic email format** |
| **3** | **keywords\_rule** | **Rule for keyword text length** |
| **4** | **exam\_reference\_rule** | **validate Exam\_id exists in Exam table** |

# Backups:

|  |  |
| --- | --- |
| **name** | **Images** |
| **Auto Backup** | **A screenshot of a computer program  AI-generated content may be incorrect.** |

# Server name:

|  |  |
| --- | --- |
| **Server name** | **db19793.public.databaseasp.net** |
| **Username** | **db19793** |
| **Password** | **123456789** |
| **Username** | **SuperAdmin** |
| **Password** | **Sp123456** |
| **Username** | **Student** |
| **Password** | **St123456** |
| **Username** | **Manager** |
| **Password** | **M123456** |
| **Username** | **Instructor** |
| **Password** | **I123456** |

# Team members:

|  |  |  |
| --- | --- | --- |
| **num** | **name** | **Responsibility** |
| **1** | **Abdalla Atef** | **Mapping, Trigger** |
| **2** | **Ahmed Sayed** | **Table Creation, Stored Procedures** |
| **3** | **Mahmoud Hamed** | **View, Index** |
| **4** | **Shimaa Mohamed** | **Function, Backup** |
| **5** | **Youssef Abo khaled** | **ERD schema, Rules, Documentation** |