**Database: LearningManagementSystem**

**Description:** The core database for managing educational programs, user roles, courses, student performance, instructor assignments, branch operations, and post-graduation tracking.

**Entities and Their Attributes:**

**1. Entity: Branch**

* **Description:** Represents a physical location or campus where educational activities take place.
* **Attributes:**
  + BranchID (INT, Primary Key, IDENTITY(1,1)): A unique, auto-incrementing identifier for each branch.
    - *Domain:* Positive integers, auto-generated.
  + BranchName (VARCHAR(100), Not Null): The official name of the branch.
    - *Domain:* Text string, e.g., 'Cairo Branch', 'Alexandria Campus'.
  + Location (VARCHAR(100)): The geographical location of the branch, possibly derived from a governorate name.
    - *Domain:* Text string, e.g., 'Port Said', 'Giza'.
  + ContactEmail (VARCHAR(100)): The primary contact email address for the branch.
    - *Domain:* Valid email format, single value.
  + ContactPhone (VARCHAR(20)): The primary contact phone number for the branch.
    - *Domain:* Text string representing a phone number, e.g., '+201234567890'.

**2. Entity: Program**

* **Description:** Represents a high-level educational offering that encompasses multiple tracks, e.g., "Nine Months Program" or "Professional Training Program".
* **Attributes:**
  + ProgramID (INT, Primary Key, IDENTITY(1,1)): A unique, auto-incrementing identifier for each program.
    - *Domain:* Positive integers, auto-generated.
  + ProgramName (VARCHAR(100), Not Null, Unique): The official name of the program. Must be unique to avoid ambiguity.
    - *Domain:* Text string, e.g., 'Intensive Code Camp', 'Early Career Build Up'.
  + Description (TEXT): A detailed explanation of the program's objectives and content.
    - *Domain:* Long text.
  + DurationInMonths (INT): The typical duration of the program in months.
    - *Domain:* Positive integers, e.g., 1, 4, 9.
  + AdminID (INT, Not Null, Foreign Key to Admin): The administrator responsible for overseeing this program.
    - *Domain:* Positive integers corresponding to Admin.AdminID.

**3. Entity: Intake**

* **Description:** Represents a specific enrollment period or cohort for a particular program.
* **Attributes:**
  + IntakeID (VARCHAR(20), Primary Key): A unique identifier for the intake. It can be semantic (e.g., '2025Q1-FULLTIME').
    - *Domain:* Alphanumeric string, e.g., '2025Q1-FS', 'F2024-PT'.
  + IntakeType (VARCHAR(50)): Describes the nature or mode of the intake within its program.
    - *Domain:* Text string, e.g., 'Full-time Day', 'Part-time Evening', 'Weekend Bootcamp'.
  + StartDate (DATE): The official start date of the intake.
    - *Domain:* Valid date, e.g., '2025-01-15'.
  + EndDate (DATE): The official end date of the intake.
    - *Domain:* Valid date, e.g., '2025-10-15'.
  + ProgramID (INT, Not Null, Foreign Key to Program): The program to which this intake belongs.
    - *Domain:* Positive integers corresponding to Program.ProgramID.

**4. Entity: Users**

* **Description:** A super-type entity representing all individuals who interact with the system (Admins, Instructors, Students). It holds common attributes shared by all user types.
* **Attributes:**
  + UserID (INT, Primary Key, IDENTITY(1,1)): A unique, auto-incrementing identifier for each user.
    - *Domain:* Positive integers, auto-generated.
  + Username (VARCHAR(30), Not Null, Unique): A unique login username for the user.
    - *Domain:* Alphanumeric string, e.g., 'jsmith', 'mohamed\_a'.
  + FirstName (VARCHAR(30), Not Null): The user's first name.
    - *Domain:* Text string, e.g., 'Mohamed', 'Amr'.
  + LastName (VARCHAR(30), Not Null): The user's last name.
    - *Domain:* Text string, e.g., 'Ahmed', 'Ali'.
  + Salary (DECIMAL(10,2)): **(Ambiguity - See "Missed/Ambiguous Logic" below)** Intended to store the user's salary.
    - *Domain:* Decimal number with 10 total digits and 2 decimal places, e.g., 5000.00, 75000.50.
  + Email (VARCHAR(100), Not Null, Unique): The user's email address. Must be unique.
    - *Domain:* Valid email format, e.g., 'user@example.com'.
  + DateOfBirth (DATE): The user's date of birth.
    - *Domain:* Valid date.
  + Gender (CHAR(1), Check Constraint ('M', 'F')): The user's gender.
    - *Domain:* 'M' (Male) or 'F' (Female).
  + PasswordHash (VARCHAR(255), Not Null): A hashed version of the user's password for security.
    - *Domain:* Alphanumeric string representing a hash.
  + BranchID (INT, Not Null, Foreign Key to Branch): The branch the user is primarily associated with.
    - *Domain:* Positive integers corresponding to Branch.BranchID.
  + Address (VARCHAR(200)): The user's physical address.
    - *Domain:* Text string.

**5. Entity: UserRoles**

* **Description:** A junction table defining the specific roles (Admin, Instructor, Student) assigned to a user. A user can have multiple roles.
* **Attributes:**
  + UserID (INT, Primary Key, Foreign Key to Users): The ID of the user.
    - *Domain:* Positive integers corresponding to Users.UserID.
  + Role (VARCHAR(30), Primary Key, Check Constraint ('Admin', 'Instructor', 'Student')): The role assigned to the user.
    - *Domain:* 'Admin', 'Instructor', 'Student'.

**6. Entity: Admin**

* **Description:** Represents a user who has administrative privileges in the system. It specializes the Users entity.
* **Attributes:**
  + AdminID (INT, Primary Key, Foreign Key to Users): The ID of the administrator, which is also their UserID.
    - *Domain:* Positive integers corresponding to Users.UserID.
  + Position (VARCHAR(50)): The administrative position held.
    - *Domain:* Text string, e.g., 'System Administrator', 'Program Manager'.
  + HiringDate (DATE): The date the administrator was hired.
    - *Domain:* Valid date.

**7. Entity: Track**

* **Description:** Represents a specialized learning path within a program, focusing on specific skills or technologies.
* **Attributes:**
  + TrackID (INT, Primary Key): A unique identifier for the track.
    - *Domain:* Positive integers.
  + TrackName (VARCHAR(100), Not Null): The official name of the track.
    - *Domain:* Text string, e.g., 'Fullstack with Python', 'PowerBI Development'.
  + Description (TEXT): A detailed description of the track's content and objectives.
    - *Domain:* Long text.
  + TrackCategory (VARCHAR(100)): A broader category for the track.
    - *Domain:* Text string, e.g., 'Web Development', 'Data Science'.
  + AdminID (INT, Not Null, Foreign Key to Admin): The administrator responsible for this specific track.
    - *Domain:* Positive integers corresponding to Admin.AdminID.
  + ProgramID (INT, Not Null, Foreign Key to Program): The program to which this track belongs.
    - *Domain:* Positive integers corresponding to Program.ProgramID.

**8. Entity: Student**

* **Description:** Represents a user who is currently enrolled in a learning program and track. It specializes the Users entity.
* **Attributes:**
  + StudentID (INT, Primary Key, Foreign Key to Users): The ID of the student, which is also their UserID.
    - *Domain:* Positive integers corresponding to Users.UserID.
  + Major (VARCHAR(100)): The student's academic major (e.g., from their college degree).
    - *Domain:* Text string.
  + CollageGPA (DECIMAL(3,2)): The student's GPA from their college.
    - *Domain:* Decimal number, e.g., 3.50, 2.75.
  + IntakeID (VARCHAR(20), Not Null, Foreign Key to Intake): The specific intake period the student is enrolled in.
    - *Domain:* Alphanumeric string corresponding to Intake.IntakeID.
  + TrackID (INT, Not Null, Foreign Key to Track): The specific track the student is pursuing.
    - *Domain:* Positive integers corresponding to Track.TrackID.

**9. Entity: Instructor**

* **Description:** Represents a user who teaches courses and is associated with tracks. It specializes the Users entity.
* **Attributes:**
  + InstructorID (INT, Primary Key, Foreign Key to Users): The ID of the instructor, which is also their UserID.
    - *Domain:* Positive integers corresponding to Users.UserID.
  + ExpertiseArea (VARCHAR(100)): The instructor's primary area of expertise.
    - *Domain:* Text string, e.g., 'Machine Learning', 'Frontend Development'.
  + HiringDate (DATE): The date the instructor was hired.
    - *Domain:* Valid date.
  + ContractType (VARCHAR(20), Check Constraint ('External', 'Internal')): The type of employment contract for the instructor.
    - *Domain:* 'External' or 'Internal'.
  + InstructorDegrea (VARCHAR(20), Check Constraint ('PHD', 'Master', 'Bechlor')): The highest academic degree obtained by the instructor.
    - *Domain:* 'PHD', 'Master', 'Bechlor'.

**10. Entity: Course**

* **Description:** Represents a specific subject or module within a track.
* **Attributes:**
  + CourseCode (VARCHAR(10), Primary Key): A unique code for the course.
    - *Domain:* Alphanumeric string, e.g., 'PY101', 'DB202'.
  + CourseName (VARCHAR(100), Not Null): The full name of the course.
    - *Domain:* Text string, e.g., 'Introduction to Python', 'Database Fundamentals'.
  + Hours (INT): The total number of instructional hours for the course.
    - *Domain:* Positive integers.
  + Credits (INT): The number of academic credits awarded for the course.
    - *Domain:* Positive integers.
  + Prerequisites (VARCHAR(10), Foreign Key to Course.CourseCode): **(Ambiguity - See "Missed/Ambiguous Logic" below)** A course code representing a single prerequisite course.
    - *Domain:* Alphanumeric string corresponding to Course.CourseCode, or NULL.

**11. Entity: Exam**

* **Description:** Represents a specific assessment or test given within a course.
* **Attributes:**
  + ExamID (INT, Primary Key, IDENTITY(1,1)): A unique, auto-incrementing identifier for the exam.
    - *Domain:* Positive integers, auto-generated.
  + ExamType (VARCHAR(20), Check Constraint ('Midterm', 'Final', 'Quiz')): The type of exam.
    - *Domain:* 'Midterm', 'Final', 'Quiz'.
  + TotalMarks (INT, Not Null): The maximum possible score for the exam.
    - *Domain:* Positive integers.
  + PassingScore (INT, Not Null): The minimum score required to pass the exam.
    - *Domain:* Positive integers, less than or equal to TotalMarks.
  + Duration (INT): The duration of the exam in minutes.
    - *Domain:* Positive integers.
  + ScheduleDate (DATETIME): The date and time the exam is scheduled.
    - *Domain:* Valid date and time.
  + CourseCode (VARCHAR(10), Not Null, Foreign Key to Course): The course this exam belongs to.
    - *Domain:* Alphanumeric string corresponding to Course.CourseCode.

**12. Entity: Question**

* **Description:** Represents a specific question used in exams or question banks.
* **Attributes:**
  + QuestionID (INT, Primary Key, IDENTITY(1,1)): A unique, auto-incrementing identifier for the question.
    - *Domain:* Positive integers, auto-generated.
  + QuestionText (TEXT, Not Null): The full text of the question.
    - *Domain:* Long text.
  + DifficultyLevel (VARCHAR(20)): The perceived difficulty of the question.
    - *Domain:* Text string, e.g., 'Easy', 'Medium', 'Hard'.
  + QuestionType (VARCHAR(20), Check Constraint ('T/F', 'MCQ')): The format of the question.
    - *Domain:* 'T/F' (True/False) or 'MCQ' (Multiple Choice Question).
  + CourseCode (VARCHAR(10), Not Null, Foreign Key to Course): The course this question is related to.
    - *Domain:* Alphanumeric string corresponding to Course.CourseCode.

**13. Entity: Options**

* **Description:** Represents the possible answer choices for a question, especially for MCQ type questions.
* **Attributes:**
  + OptionID (INT, Primary Key, IDENTITY(1,1)): A unique, auto-incrementing identifier for each option.
    - *Domain:* Positive integers, auto-generated.
  + OptionText (VARCHAR(50)): The text of the answer option.
    - *Domain:* Text string.
  + QuestionID (INT, Not Null, Foreign Key to Question): The question this option belongs to.
    - *Domain:* Positive integers corresponding to Question.QuestionID.
  + IsCorrect (BIT): Indicates if this option is the correct answer for the question.
    - *Domain:* 0 (False) or 1 (True).

**14. Entity: JobProfile**

* **Description:** Defines a potential career role or type of job that tracks or programs aim to prepare students for.
* **Attributes:**
  + JobID (INT, Primary Key, IDENTITY(1,1)): A unique, auto-incrementing identifier for each job profile.
    - *Domain:* Positive integers, auto-generated.
  + JobTitle (VARCHAR(100), Not Null, Unique): The title of the job profile.
    - *Domain:* Text string, e.g., 'Junior Python Developer', 'Data Analyst'.
  + JobDescription (TEXT): A detailed description of the job role and responsibilities.
    - *Domain:* Long text.

**15. Entity: GraduatedStudent**

* **Description:** Contains additional details specific to students who have successfully graduated from a program. It's a specialization of the Student entity.
* **Attributes:**
  + StudentID (INT, Primary Key, Foreign Key to Student): The ID of the graduated student, which is also their StudentID. This correctly models the 1:1 relationship with Student.
    - *Domain:* Positive integers corresponding to Student.StudentID.
  + PositionTitle (VARCHAR(100)): The job title of the first position obtained after graduation.
    - *Domain:* Text string.
  + HiringStatus (VARCHAR(30)): The employment status after graduation.
    - *Domain:* Text string, e.g., 'Employed', 'Seeking Employment'.
  + CompanyName (VARCHAR(100)): The name of the company where the student was first employed.
    - *Domain:* Text string.
  + CompanyLevel (VARCHAR(30)): The type or size of the company.
    - *Domain:* Text string, e.g., 'Multinational', 'National', 'Startup'.
  + FinalEvaluation (INT): The student's final evaluation score.
    - *Domain:* Integers, e.g., 70, 85, 95.
  + GraduationDate (DATE): The date the student officially graduated.
    - *Domain:* Valid date.

**16. Entity: CertificateKPIs**

* **Description:** Stores information about external certifications obtained by students, which contribute to their Key Performance Indicators (KPIs).
* **Attributes:**
  + CertID (INT, Primary Key): A unique identifier for each certificate record.
    - *Domain:* Positive integers.
  + CertificateName (VARCHAR(100)): The name of the certification obtained.
    - *Domain:* Text string, e.g., 'AWS Certified Cloud Practitioner'.
  + CertificateSource (VARCHAR(100)): The issuing body or platform for the certificate.
    - *Domain:* Text string, e.g., 'Coursera', 'Microsoft', 'AWS'.
  + Cost (DECIMAL(10,2)): The cost incurred to obtain the certificate.
    - *Domain:* Decimal number, e.g., 100.00, 49.99.
  + CrsHours (INT): The number of hours associated with the certification course.
    - *Domain:* Positive integers.
  + StudentID (INT, Not Null, Foreign Key to Student): The student who obtained this certificate.
    - *Domain:* Positive integers corresponding to Student.StudentID.

**17. Entity: Feedback**

* **Description:** Stores feedback provided by students about various aspects of the learning system, such as instructors, courses, tracks, or general system experience.
* **Attributes:**
  + FeedbackID (INT, Primary Key, IDENTITY(1,1)): A unique, auto-incrementing identifier for each feedback record.
    - *Domain:* Positive integers, auto-generated.
  + FeedbackText (TEXT, Not Null): The actual text content of the feedback.
    - *Domain:* Long text.
  + FeedbackRate (INT, Check Constraint BETWEEN 1 AND 5): A numerical rating associated with the feedback.
    - *Domain:* Integers from 1 to 5.
  + FeedbackDate (DATE, Not Null, Default GETDATE()): The date the feedback was submitted.
    - *Domain:* Valid date, defaults to current date.
  + StudentID (INT, Not Null, Foreign Key to Student): The student who submitted the feedback.
    - *Domain:* Positive integers corresponding to Student.StudentID.
  + FeedbackType (VARCHAR(50), Not Null, Check Constraint ('Instructor', 'Course', 'Track', 'General System')): Categorizes what the feedback is about.
    - *Domain:* 'Instructor', 'Course', 'Track', 'General System'.
  + InstructorID (INT, Nullable, Foreign Key to Instructor): The instructor the feedback is about (if FeedbackType is 'Instructor').
    - *Domain:* Positive integers corresponding to Instructor.InstructorID, or NULL.
  + CourseCode (VARCHAR(10), Nullable, Foreign Key to Course): The course the feedback is about (if FeedbackType is 'Course').
    - *Domain:* Alphanumeric string corresponding to Course.CourseCode, or NULL.
  + TrackID (INT, Nullable, Foreign Key to Track): The track the feedback is about (if FeedbackType is 'Track').
    - *Domain:* Positive integers corresponding to Track.TrackID, or NULL.
  + CK\_Feedback\_Target\_Logic (CHECK Constraint): Ensures that based on FeedbackType, only the relevant target foreign key (InstructorID, CourseCode, or TrackID) is populated, and others are NULL. If FeedbackType is 'General System', all target FKs must be NULL.

**18. Entity: JobProfile**

* **Description:** Defines a potential career role or type of job that tracks or programs aim to prepare students for.
* **Attributes:**
  + JobID (INT, Primary Key, IDENTITY(1,1)): A unique, auto-incrementing identifier for each job profile.
    - *Domain:* Positive integers, auto-generated.
  + JobTitle (VARCHAR(100), Not Null, Unique): The title of the job profile.
    - *Domain:* Text string, e.g., 'Junior Python Developer', 'Data Analyst'.
  + JobDescription (TEXT): A detailed description of the job role and responsibilities.
    - *Domain:* Long text.

**19. Entity: FreelanceJob**

* **Description:** Records details of freelancing jobs undertaken by students. This is used to track work experience that contributes to graduation requirements.
* **Attributes:**
  + FreelanceJobID (INT, Primary Key, IDENTITY(1,1)): A unique, auto-incrementing identifier for each freelancing job.
    - *Domain:* Positive integers, auto-generated.
  + StudentID (INT, Not Null, Foreign Key to Student): The student who performed this freelancing job.
    - *Domain:* Positive integers corresponding to Student.StudentID.
  + JobTitle (VARCHAR(200), Not Null): The title or a brief description of the freelancing project.
    - *Domain:* Text string.
  + Description (TEXT): A more detailed description of the freelancing work.
    - *Domain:* Long text.
  + ClientNationality (VARCHAR(20)): The nationality of the client for whom the freelancing job was done.
    - *Domain:* Text string, e.g., 'Egyptian', 'American'. (Note: Typo in original code ClinetNationality).
  + Platform (VARCHAR(50)): The platform used for the freelancing job.
    - *Domain:* Text string, e.g., 'Upwork', 'Fiverr', 'Direct Client'.
  + CompensationAmountInDollar (DECIMAL(10,2), Nullable): The compensation received for the job, specified in USD.
    - *Domain:* Decimal number, e.g., 50.00, 150.75. (Implicitly USD, good for explicit naming).
  + JobStatus (VARCHAR(30), Not Null, Check Constraint): The current status of the freelancing job.
    - *Domain:* 'Completed', 'In Progress', 'Cancelled', 'Delivered', 'On Hold'. (Note: StartDate and EndDate are missing from the final code provided, which were in the previous version. This might be an oversight.)

**Relationships: Cardinality, Participation, and Logic**

1. **Branch - Users**
   * **Cardinality:** One-to-Many (1:M)
   * **Participation:**
     + Branch: Optional (A branch may exist without any users yet).
     + Users: Mandatory (Every user must be associated with a branch, BranchID INT NOT NULL).
   * **Logic:** A user is primarily affiliated with one specific branch.
2. **Users - UserRoles**
   * **Cardinality:** One-to-Many (1:M) (specifically, Users to the composite PK of UserRoles)
   * **Participation:**
     + Users: Optional (A user may or may not have a specific role defined in UserRoles if UserRoles only covers Admin/Instructor/Student and other user types could exist, though your current CHECK constraint makes it mandatory to have one of these three roles for any entry in UserRoles).
     + UserRoles: Mandatory (Every role assignment must belong to an existing user, UserID INT NOT NULL).
   * **Logic:** Defines which roles (Admin, Instructor, Student) a user possesses. A user can have multiple roles.
3. **Users - Admin**
   * **Cardinality:** One-to-One (1:1) (Super-type/Sub-type specialization)
   * **Participation:**
     + Users: Optional (A user does not have to be an admin).
     + Admin: Mandatory (An admin *must* be a user, enforced by AdminID being PK and FK to UserID).
   * **Logic:** An Admin is a specialized type of User, inheriting the UserID.
4. **Users - Student**
   * **Cardinality:** One-to-One (1:1) (Super-type/Sub-type specialization)
   * **Participation:**
     + Users: Optional (A user does not have to be a student).
     + Student: Mandatory (A student *must* be a user, enforced by StudentID being PK and FK to UserID).
   * **Logic:** A Student is a specialized type of User, inheriting the UserID.
5. **Users - Instructor**
   * **Cardinality:** One-to-One (1:1) (Super-type/Sub-type specialization)
   * **Participation:**
     + Users: Optional (A user does not have to be an instructor).
     + Instructor: Mandatory (An instructor *must* be a user, enforced by InstructorID being PK and FK to UserID).
   * **Logic:** An Instructor is a specialized type of User, inheriting the UserID.
6. **Program - Intake**
   * **Cardinality:** One-to-Many (1:M)
   * **Participation:**
     + Program: Optional (A program may not have any intakes yet).
     + Intake: Mandatory (Every intake must belong to a program, ProgramID INT NOT NULL).
   * **Logic:** An intake is a specific cohort or period of enrollment for a particular program.
7. **Program - Track**
   * **Cardinality:** One-to-Many (1:M)
   * **Participation:**
     + Program: Optional (A program may not have any tracks defined yet).
     + Track: Mandatory (Every track must belong to a program, ProgramID INT NOT NULL).
   * **Logic:** A track is a specific learning path that exists within a larger program.
8. **Admin - Program**
   * **Cardinality:** One-to-Many (1:M)
   * **Participation:**
     + Admin: Optional (An admin may not be assigned to manage any programs yet).
     + Program: Mandatory (Every program must have an assigned admin, AdminID INT NOT NULL).
   * **Logic:** An admin is responsible for overseeing one or more programs.
9. **Admin - Track**
   * **Cardinality:** One-to-Many (1:M)
   * **Participation:**
     + Admin: Optional (An admin may not be assigned to manage any tracks yet).
     + Track: Mandatory (Every track must have an assigned admin, AdminID INT NOT NULL).
   * **Logic:** An admin is responsible for overseeing one or more tracks.
10. **Branch - Branch\_Intake\_Track**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Branch: Optional (A branch may not have specific Branch\_Intake\_Track combinations defined yet).
      + Branch\_Intake\_Track: Mandatory (Every entry in this junction table must be associated with a branch, BranchID INT NOT NULL).
    * **Logic:** Part of defining a specific program offering instance at a branch for a given intake.
11. **Intake - Branch\_Intake\_Track**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Intake: Optional (An intake may not be offered at any branch for any track yet).
      + Branch\_Intake\_Track: Mandatory (Every entry must be associated with an intake, IntakeID VARCHAR(20) NOT NULL).
    * **Logic:** Part of defining a specific program offering instance at a branch for a given intake.
12. **Track - Branch\_Intake\_Track**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Track: Optional (A track may not be offered at any branch for any intake yet).
      + Branch\_Intake\_Track: Mandatory (Every entry must be associated with a track, TrackID INT NOT NULL).
    * **Logic:** Part of defining a specific program offering instance at a branch for a given intake.
13. **Student - Intake** (via Student.IntakeID)
    * **Cardinality:** Many-to-One (M:1)
    * **Participation:**
      + Student: Mandatory (Every student must be enrolled in an intake, IntakeID VARCHAR(20) NOT NULL).
      + Intake: Optional (An intake may have no students enrolled yet).
    * **Logic:** A student enrolls in a specific intake period.
14. **Student - Track** (via Student.TrackID)
    * **Cardinality:** Many-to-One (M:1)
    * **Participation:**
      + Student: Mandatory (Every student must be pursuing a specific track, TrackID INT NOT NULL).
      + Track: Optional (A track may have no students enrolled yet).
    * **Logic:** A student pursues a specific learning track.
15. **Course - Course\_Track**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Course: Optional (A course may not be part of any track).
      + Course\_Track: Mandatory (Every entry must be associated with a course, CourseCode VARCHAR(10) NOT NULL).
    * **Logic:** Part of defining which courses belong to which tracks.
16. **Track - Course\_Track**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Track: Optional (A track may not have any courses defined yet).
      + Course\_Track: Mandatory (Every entry must be associated with a track, TrackID INT NOT NULL).
    * **Logic:** Part of defining which courses belong to which tracks. (Resulting in a Many-to-Many between Course and Track).
17. **Instructor - Instructor\_Course**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Instructor: Optional (An instructor may not be assigned to teach any courses yet).
      + Instructor\_Course: Mandatory (Every teaching assignment must be associated with an instructor, InstructorID INT NOT NULL).
    * **Logic:** Part of defining which instructors teach which courses.
18. **Course - Instructor\_Course**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Course: Optional (A course may not have an instructor assigned yet).
      + Instructor\_Course: Mandatory (Every teaching assignment must be associated with a course, CourseCode VARCHAR(10) NOT NULL).
    * **Logic:** Part of defining which instructors teach which courses. (Resulting in a Many-to-Many between Instructor and Course).
19. **Instructor - Instructor\_Track**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Instructor: Optional (An instructor may not be explicitly associated with any tracks yet).
      + Instructor\_Track: Mandatory (Every track association must be for a valid instructor, InstructorID INT NOT NULL).
    * **Logic:** Part of defining an instructor's general affiliation or expertise with a track.
20. **Track - Instructor\_Track**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Track: Optional (A track may not have any instructors explicitly associated yet).
      + Instructor\_Track: Mandatory (Every instructor association must be for a valid track, TrackID INT NOT NULL).
    * **Logic:** Part of defining an instructor's general affiliation or expertise with a track. (Resulting in a Many-to-Many between Instructor and Track).
21. **Student - Student\_Course**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Student: Optional (A student may not be enrolled in any courses yet).
      + Student\_Course: Mandatory (Every course enrollment must be for a valid student, StudentID INT NOT NULL).
    * **Logic:** Part of tracking student enrollment and performance in courses.
22. **Course - Student\_Course**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Course: Optional (A course may have no students enrolled yet).
      + Student\_Course: Mandatory (Every course enrollment must be for a valid course, CourseCode VARCHAR(10) NOT NULL).
    * **Logic:** Part of tracking student enrollment and performance in courses. (Resulting in a Many-to-Many between Student and Course).
23. **Course - Exam**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Course: Optional (A course may not have any exams yet).
      + Exam: Mandatory (Every exam must belong to a course, CourseCode VARCHAR(10) NOT NULL).
    * **Logic:** An exam is an assessment specifically designed for a course.
24. **Course - Question**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Course: Optional (A course may not have any questions in the question bank yet).
      + Question: Mandatory (Every question must be associated with a course, CourseCode VARCHAR(10) NOT NULL).
    * **Logic:** Questions are categorized and associated with specific courses.
25. **Question - Options**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Question: Optional (A question may not have any options defined yet, e.g., for T/F questions).
      + Options: Mandatory (Every option must belong to a question, QuestionID INT NOT NULL).
    * **Logic:** Options provide choices for specific questions.
26. **Exam - Exam\_Question**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Exam: Optional (An exam may not have questions assigned yet).
      + Exam\_Question: Mandatory (Every question assignment to an exam must be for a valid exam, ExamID INT NOT NULL).
    * **Logic:** Part of defining which questions are included in a particular exam.
27. **Question - Exam\_Question**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Question: Optional (A question may not be used in any exams yet).
      + Exam\_Question: Mandatory (Every question assignment to an exam must be for a valid question, QuestionID INT NOT NULL).
    * **Logic:** Part of defining which questions are included in a particular exam. (Resulting in a Many-to-Many between Exam and Question).
28. **Exam - Student\_Exam**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Exam: Optional (An exam may not have been taken by any students yet).
      + Student\_Exam: Mandatory (Every exam attempt must be for a valid exam, ExamID INT NOT NULL).
    * **Logic:** Part of recording student attempts and scores on exams.
29. **Student - Student\_Exam**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Student: Optional (A student may not have taken any exams yet).
      + Student\_Exam: Mandatory (Every exam attempt must be for a valid student, StudentID INT NOT NULL).
    * **Logic:** Part of recording student attempts and scores on exams. (Resulting in a Many-to-Many between Student and Exam).
30. **Student - GraduatedStudent**
    * **Cardinality:** One-to-One (1:1) (Specialization)
    * **Participation:**
      + Student: Optional (A student does not have to be graduated yet).
      + GraduatedStudent: Mandatory (A graduated student *must* be an existing student, enforced by StudentID being PK and FK).
    * **Logic:** A GraduatedStudent is a specialized type of Student who has completed their program.
31. **Student - CertificateKPIs**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Student: Optional (A student may not have any external certificates yet).
      + CertificateKPIs: Mandatory (Every certificate must belong to a student, StudentID INT NOT NULL).
    * **Logic:** Tracks external certifications obtained by students.
32. **Student - Feedback**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Student: Optional (A student may not have submitted any feedback yet).
      + Feedback: Mandatory (Every feedback submission must be from a student, StudentID INT NOT NULL).
    * **Logic:** Tracks feedback provided by students.
33. **Instructor - Feedback** (Conditional via FeedbackType)
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Instructor: Optional (An instructor may not have feedback about them).
      + Feedback: Optional (A feedback record is not always about an instructor, InstructorID NULL). Mandatory if FeedbackType is 'Instructor'.
    * **Logic:** Allows students to provide feedback specific to an instructor.
34. **Course - Feedback** (Conditional via FeedbackType)
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Course: Optional (A course may not have feedback about it).
      + Feedback: Optional (A feedback record is not always about a course, CourseCode NULL). Mandatory if FeedbackType is 'Course'.
    * **Logic:** Allows students to provide feedback specific to a course.
35. **Track - Feedback** (Conditional via FeedbackType)
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Track: Optional (A track may not have feedback about it).
      + Feedback: Optional (A feedback record is not always about a track, TrackID NULL). Mandatory if FeedbackType is 'Track'.
    * **Logic:** Allows students to provide feedback specific to a track.
36. **Track - Track\_JobProfile**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Track: Optional (A track may not be mapped to any job profiles yet).
      + Track\_JobProfile: Mandatory (Every mapping must be for a valid track, TrackID INT NOT NULL).
    * **Logic:** Part of defining which job profiles a track can lead to.
37. **JobProfile - Track\_JobProfile**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + JobProfile: Optional (A job profile may not be mapped to any tracks yet).
      + Track\_JobProfile: Mandatory (Every mapping must be for a valid job profile, JobID INT NOT NULL).
    * **Logic:** Part of defining which job profiles a track can lead to. (Resulting in a Many-to-Many between Track and JobProfile).
38. **Student - FreelanceJob**
    * **Cardinality:** One-to-Many (1:M)
    * **Participation:**
      + Student: Optional (A student may not have completed any freelancing jobs yet).
      + FreelanceJob: Mandatory (Every freelancing job must be associated with a student, StudentID INT NOT NULL).
    * **Logic:** Tracks freelancing work undertaken by students, potentially for graduation requirements.