Database Schema Documentation

Database Schema Overview

The database for the online education platform consists of several interconnected tables designed to manage various entities such as instructors, categories, courses, students, enrollments, and feedback. Each table is linked using primary and foreign key relationships to maintain data integrity and ensure efficient querying.

Entity-Relationship Diagram (ERD)

The ERD illustrates the relationships between different entities in the database. Key relationships include:

1. **Instructor** can teach many **Course** instances.
2. **Category** can have many **Course** instances.
3. **Course** can be enrolled by many **Student** instances.
4. **Student** can enroll in many **Course** instances.
5. **Feedback** is linked to both **Student** and **Course** entities.

Tables and Relationships

1. **Instructor Table**

```
CREATE TABLE IF NOT EXISTS Instructor (
    ID BIGSERIAL PRIMARY KEY,
    Name VARCHAR(100) UNIQUE NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    Bio TEXT,
    Registration_Date DATE DEFAULT CURRENT_DATE
);
```

Description:
- **ID**: Unique identifier for each instructor.
- **Name**: Name of the instructor. It is required and unique.
- **Email**: Email address of the instructor. It is required and unique.
- **Bio**: Short biography of the instructor.
- **Registration_Date**: The date when the instructor registered on the platform. Defaults to the current date.

2. **Category Table**

```
CREATE TABLE IF NOT EXISTS Category (
    ID BIGSERIAL PRIMARY KEY,
    Name VARCHAR(100) UNIQUE NOT NULL,
    Description TEXT
);
```

Description:

- **ID**: Unique identifier for each category.
- **Name**: Name of the category. It is required and unique.
- **Description**: Detailed description of the category.

3. **Course Table**

```
CREATE TABLE IF NOT EXISTS Course (
    ID BIGSERIAL PRIMARY KEY,
    Title VARCHAR(200) NOT NULL,
    Description TEXT,
    Duration INT NOT NULL,
    Price INT NOT NULL,
    Instructor_ID BIGINT REFERENCES Instructor(ID) ON DELETE SET NULL,
    Category_ID BIGINT REFERENCES Category(ID) ON DELETE SET NULL,
    Total_Enrollments INT DEFAULT 0 CHECK (Total_Enrollments >= 0),
    Average_Rating FLOAT DEFAULT 0.0 CHECK (Average_Rating BETWEEN 0
AND 5),
    Creation_Date DATE DEFAULT CURRENT_DATE
);
```

Description:

- **ID**: Unique identifier for each course.
- **Title**: Title of the course. It is required.
- **Description**: Detailed description of the course.
- **Duration**: Duration of the course in hours. It is required.
- **Price**: Price of the course. It is required.
- **Instructor_ID**: ID of the instructor teaching the course. It references the `Instructor` table.
- **Category_ID**: ID of the category to which the course belongs. It references the `Category` table.
- **Total_Enrollments**: The total number of students enrolled in the course. Defaults to 0 and should be a non-negative number.
- **Average_Rating**: The average rating given by students. It ranges between 0 and 5.
- **Creation_Date**: The date when the course was created. Defaults to the current date.

4. **Student Table**

```
CREATE TABLE IF NOT EXISTS Student (
    ID BIGSERIAL PRIMARY KEY,
    Name VARCHAR(100) UNIQUE NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    Date_Of_Birth DATE NOT NULL,
    Registration_Date DATE DEFAULT CURRENT_DATE
);
```

Description:

- **ID**: Unique identifier for each student.
- **Name**: Name of the student. It is required and unique.
- **Email**: Email address of the student. It is required and unique.
- **Date_Of_Birth**: Date of birth of the student. It is required.
- **Registration_Date**: The date when the student registered on the platform. Defaults to the current date.

5. **Enrollment Table**

```
CREATE TABLE IF NOT EXISTS Enrollment (
    ID BIGSERIAL PRIMARY KEY,
    Student_ID BIGINT REFERENCES Student(ID) ON DELETE CASCADE,
    Course_ID BIGINT REFERENCES Course(ID) ON DELETE CASCADE,
    Enrollment_Date DATE DEFAULT CURRENT_DATE,
    Completion_Status BOOLEAN DEFAULT FALSE,
    UNIQUE (Student_ID, Course_ID)
);
```

Description:

- **ID**: Unique identifier for each enrollment.
- **Student_ID**: ID of the student who enrolled. It references the `Student` table.
- **Course_ID**: ID of the course in which the student enrolled. It references the `Course` table.
- **Enrollment_Date**: The date when the student enrolled in the course. Defaults to the current date.
- **Completion_Status**: Indicates whether the student has completed the course. Defaults to `FALSE`.
- **UNIQUE** constraint on `Student_ID, Course_ID`: Ensures a student cannot enroll in the same course more than once.

6. **Feedback Table**

```
CREATE TABLE IF NOT EXISTS Feedback (
    ID BIGSERIAL PRIMARY KEY,
    Student_ID BIGINT REFERENCES Student(ID) ON DELETE CASCADE,
    Course_ID BIGINT REFERENCES Course(ID) ON DELETE CASCADE,
    Rating FLOAT NOT NULL CHECK (Rating BETWEEN 0 AND 5),
    Comment TEXT,
    Feedback_Date DATE DEFAULT CURRENT_DATE
);
```

Description:

- **ID**: Unique identifier for each feedback entry.
- **Student_ID**: ID of the student giving the feedback. It references the `Student` table.
- **Course_ID**: ID of the course for which feedback is provided. It references the `Course` table.
- **Rating**: The rating given by the student. It ranges between 0 and 5.
- **Comment**: Optional comment provided by the student.
- **Feedback_Date**: The date when the feedback was submitted. Defaults to the current date.

Relationships

- **One-to-Many Relationship**:
  - **Instructor to Course**: An instructor can teach multiple courses. The `Instructor_ID` in the `Course` table references the `ID` in the `Instructor` table. If an instructor is deleted, the `Instructor_ID` in all related courses will be set to NULL.
  - **Category to Course**: A course belongs to a specific category (e.g., Programming, Data Science). The `Category_ID` in the `Course` table references the `ID` in the `Category` table. If a category is deleted, the `Category_ID` in all related courses will be set to NULL.
- **Many-to-One Relationship**:
  - **Student to Enrollment**: A student can be enrolled in multiple courses. The `Student_ID` in the `Enrollment` table references the `ID` in the `Student` table. If a student is deleted, all related enrollments will be deleted due to the cascade delete constraint.
  - **Course to Enrollment**: A course can have multiple students enrolled. The `Course_ID` in the `Enrollment` table references the `ID` in the `Course` table. If a course is deleted, all related enrollments will be deleted due to the cascade delete constraint.
  - **Student to Feedback**: A student can give feedback for multiple courses. The `Student_ID` in the `Feedback` table references the `ID` in the `Student` table. If a student is deleted, all related feedback entries will be deleted due to the cascade delete constraint.
  - **Course to Feedback**: A course can receive feedback from multiple students. The `Course_ID` in the `Feedback` table references the `ID` in the `Course` table. If a course is deleted, all related feedback entries will be deleted due to the cascade delete constraint.

Database Constraints

- **Primary Keys**:
  - `ID` in `Instructor`, `Category`, `Course`, `Student`, `Enrollment`, `Feedback` tables.
- **Foreign Key Constraints**:
  - `Instructor_ID` in `Course` table references `ID` in `Instructor` table.
  - `Category_ID` in `Course` table references `ID` in `Category` table.
  - `Student_ID` in `Enrollment` table references `ID` in `Student` table (Cascade delete).
  - `Course_ID` in `Enrollment` table references `ID` in `Course` table (Cascade delete).
  - `Student_ID` in `Feedback` table references `ID` in `Student` table (Cascade delete).
  - `Course_ID` in `Feedback` table references `ID` in `Course` table (Cascade delete).
- **Unique Constraints**:
  - `Name` in `Instructor` table.
  - `Email` in `Instructor` and `Student` tables.
  - `Name` in `Category` table.
- **Check Constraints**:
  - `Total_Enrollments` in `Course` table must be non-negative.
  - `Average_Rating` in `Course` table must be between 0 and 5.
  - `Rating` in `Feedback` table must be between 0 and 5.

This documentation outlines the structure, relationships, and constraints of the database schema, ensuring data integrity and consistency across all tables.