**AFTER Trigger**

EmployeeDetails

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| EmployeeID | Int | Primary Key |
| EmployeeName | Varchar(100) | Not Null |
| ContactNo | Varchar(100) | Not Null |
| Department | Varchar(100) | Not Null |
| Salary | Decimal(10,2) | Not Null |
| Joining Date | DateTime | Allow Null |

CREATE TABLE EMPLOYEEDETAILS

(

EmployeeID Int Primary Key

EmployeeName Varchar(100) Not Null,

ContactNo Varchar(100) Not Null,

Department Varchar(100) Not Null,

Salary Decimal(10,2) Not Null,

JoiningDate DateTime Null

)

EmployeeLogs

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Constraints |
| LogID | Int | Primary Key, AutoIncrement |
| EmployeeID | Int | Not Null |
| EmployeeName | Varchar(100) | Not Null |
| ActionPerformed | Varchar(100) | Not Null |
| ActionDate | DateTime | Not Null |

**CREATE TABLE EmployeeLogs (**

**LogID INT PRIMARY KEY IDENTITY(1,1),**

**EmployeeID INT NOT NULL,**

**EmployeeName VARCHAR(100) NOT NULL,**

**ActionPerformed VARCHAR(100) NOT NULL,**

**ActionDate DATETIME NOT NULL**

**);**

1. Create a trigger that fires AFTER INSERT, UPDATE, and DELETE operations on the EmployeeDetails table to display the message "Employee record inserted", "Employee record updated", "Employee record deleted"
2. Create a trigger that fires AFTER INSERT, UPDATE, and DELETE operations on the EmployeeDetails table to log all operations into the EmployeeLog table.
3. Create a trigger that fires AFTER INSERT to automatically calculate the joining bonus (10% of the salary) for new employees and update a bonus column in the EmployeeDetails table.
4. Create a trigger to ensure that the JoiningDate is automatically set to the current date if it is NULL during an INSERT operation.
5. Create a trigger that ensure that ContactNo is valid during insert and update (Like ContactNo length is 10)

**Instead of Trigger**

**Movies**

|  |  |  |
| --- | --- | --- |
| **ColumnName** | **DataType** | **Constraint** |
| MovieID | INT | PRIMARY KEY |
| MovieTitle | VARCHAR(255) | NOT NULL |
| ReleaseYear | INT | NOT NULL |
| Genre | VARCHAR(100) | NOT NULL |
| Rating | DECIMAL(3,1) | NOT NULL |
| Duration | INT | NOT NULL – (In minutes) |

**CREATE TABLE Movies (**

**MovieID INT PRIMARY KEY,**

**MovieTitle VARCHAR(255) NOT NULL,**

**ReleaseYear INT NOT NULL,**

**Genre VARCHAR(100) NOT NULL,**

**Rating DECIMAL(3, 1) NOT NULL,**

**Duration INT NOT NULL**

**);**

**MoviesLog**

|  |  |  |
| --- | --- | --- |
| **ColumnName** | **DataType** | **Constraint** |
| LogID | INT | PRIMARY KEY,AUTO INCREMENT |
| MovieID | INT | NOT NULL |
| MovieTitle | VARCHAR(255) | NOT NULL |
| ActionPerformed | VARCHAR(100) | NOT NULL |
| ActionDate | DATETIME | NOT NULL |

**CREATE TABLE MoviesLog**

**(**

**LogID INT PRIMARY KEY IDENTITY(1,1),**

**MovieID INT NOT NULL,**

**MovieTitle VARCHAR(255) NOT NULL,**

**ActionPerformed VARCHAR(100) NOT NULL,**

**ActionDate DATETIME NOT NULL**

**);**

1. Create an INSTEAD OF trigger that fires on INSERT, UPDATE and DELETE operation on the Movies table. For that, log all operations performed on the Movies table into MoviesLog.
2. Create a trigger that only allows to insert movies for which Rating is greater than 5.5 .
3. Create trigger that prevent duplicate 'MovieTitle' of Movies table and log details of it in MoviesLog table.
4. Create trigger that prevents to insert pre-release movies.
5. Develop a trigger to ensure that the Duration of a movie cannot be updated to a value greater than 120 minutes (2 hours) to prevent unrealistic entries.