**AFTER Trigger**

EmployeeDetails

|  |  |  |
| --- | --- | --- |
| **ColumnName** | **Data Type** | **Constraints** |
| EmployeeID | Int | PrimaryKey |
| EmployeeName | Varchar(100) | NotNull |
| ContactNo | Varchar(100) | NotNull |
| Department | Varchar(100) | NotNull |
| Salary | Decimal(10,2) | NotNull |
| JoiningDate | DateTime | AllowNull |

CREATE TABLE EMPLOYEEDETAILS

(

EmployeeID Int Primary Key,

EmployeeNameVarchar(100) Not Null,

ContactNoVarchar(100) Not Null,

Department Varchar(100) Not Null,

Salary Decimal(10,2) Not Null,

JoiningDateDateTime Null

)

EmployeeLogs

|  |  |  |
| --- | --- | --- |
| ColumnName | Data Type | Constraints |
| LogID | Int | PrimaryKey,AutoIncrement |
| EmployeeID | Int | NotNull |
| EmployeeName | Varchar(100) | NotNull |
| ActionPerformed | Varchar(100) | NotNull |
| ActionDate | DateTime | NotNull |

**CREATE TABLE EmployeeLogs (**

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

**EmployeeID INT NOT NULL,**

**EmployeeNameVARCHAR(100) NOT NULL,**

**ActionPerformedVARCHAR(100) NOT NULL,**

**ActionDate DATETIME NOT NULL**

**);**

1. CreateatriggerthatfiresAFTERINSERT,UPDATE,andDELETEoperationsonthe EmployeeDetails table to display the message "Employee record inserted", "Employee record updated", "Employee record deleted"
2. CreateatriggerthatfiresAFTERINSERT,UPDATE,andDELETEoperationsonthe 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%ofthesalary)fornewemployeesandupdateabonuscolumnintheEmployeeDetailstable.
4. CreateatriggertoensurethattheJoiningDateisautomaticallysettothecurrentdateifit is NULL during an INSERT operation.
5. CreateatriggerthatensurethatContactNoisvalidduringinsertandupdate(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,**

**MovieTitleVARCHAR(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),**

**MovieIDINT NOT NULL,**

**MovieTitleVARCHAR(255) NOT NULL,**

**ActionPerformedVARCHAR(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.