SET A

Movie – Actor Database

Consider the following database

Movie (m_name varchar (25), release_year integer, budget money)

Actor (a_namevarchar(30), role varchar(30), charges money, a_address varchar(30))

Movie and Actor are related with many to many relationship.

Create the above database in PostGreSQL and insert sufficient records.

a. Write a trigger which will be executed whenever an actor is deleted from the actor table,

display appropriate message.

b. Write a trigger which will be executed whenever a movie is deleted from the movie table,

display appropriate message.

c. Write a trigger which will be executed whenever insertion is made to the movie table. If

the budget is less than 1,00,000 do not allow the insertion. Give appropriate message.

-- Create the Movie Table

```
CREATE TABLE Movie (
   m_name VARCHAR(25),
   release_year INTEGER,
   budget MONEY,
   PRIMARY KEY (m_name)
);
```

-- Create the Actor Table

```
CREATE TABLE Actor (
```

```sql
    a_name VARCHAR(30),
    role VARCHAR(30),
    charges MONEY,
    a_address VARCHAR(30),
    PRIMARY KEY (a_name)
);
```

-- Create a Many-to-Many Relationship Table (Movie_Actor)

```sql
CREATE TABLE Movie_Actor (
    m_name VARCHAR(25),
    a_name VARCHAR(30),
    FOREIGN KEY (m_name) REFERENCES Movie (m_name),
    FOREIGN KEY (a_name) REFERENCES Actor (a_name),
    PRIMARY KEY (m_name, a_name)
);
```

-- Insert data into the Movie table

```sql
INSERT INTO Movie VALUES ('MovieA', 2023, 150000);
INSERT INTO Movie VALUES('MovieB', 2024, 50000);
INSERT INTO Movie VALUES('MovieC', 2022, 200000);
```

-- Insert data into the Actor table

```sql
INSERT INTO Actor VALUES ('Actor1', 'Lead', 10000, 'Address1');
INSERT INTO Actor VALUES('Actor2', 'Supporting', 5000, 'Address2');
INSERT INTO Actor VALUES('Actor3', 'Lead', 12000, 'Address3');
```

-- Insert into the Movie_Actor relationship table

```sql
INSERT INTO Movie_Actor VALUES ('MovieA', 'Actor1');
INSERT INTO Movie_Actor VALUES('MovieB', 'Actor3');

INSERT INTO Movie_Actor VALUES('MovieC', 'Actor3');
```

--Q1. Create a function to handle deletion of an actor

```
CREATE OR REPLACE FUNCTION actor_deleted() RETURNS TRIGGER
AS '

BEGIN

   RAISE NOTICE " Actor % has been deleted from the Actor table.",
OLD.a_name;

   RETURN OLD;

END;

' LANGUAGE plpgsql;
```

-- Create a trigger to execute the function when an actor is deleted

```
CREATE TRIGGER actor_delete_trigger

AFTER DELETE ON Actor

FOR EACH ROW

EXECUTE FUNCTION actor_deleted();
```

--Test the trigger

```
DELETE  FROM Actor WHERE a_name ='Actor2';


select * from Actor;
```

--Q.2Write a trigger which will be executed whenever a movie is deleted from the movie table, display appropriate message.

```
CREATE OR REPLACE FUNCTION movie_deleted() RETURNS TRIGGER AS '
BEGIN
    RAISE NOTICE "Movie % has been deleted from the Movie table.", OLD.m_name;
    RETURN OLD;
END;
' LANGUAGE plpgsql;
```

-- Create a trigger to execute the function when a movie is deleted

```
CREATE TRIGGER movie_delete_trigger
AFTER DELETE ON Movie
FOR EACH ROW
EXECUTE FUNCTION movie_deleted();
```

--Test the trigger

```
DELETE FROM Movie WHERE m_name = 'MovieB';
```

Q 3. Write a trigger which will be executed whenever insertion is made to the movie table. If the budget is less than 1,00,000 do not allow the insertion. Give appropriate message.

```
CREATE OR REPLACE FUNCTION check_movie_budget()  RETURNS TRIGGER AS '
BEGIN
    IF NEW.budget < 100000 THEN
```

```
        RAISE EXCEPTION "Budget for movie % is too low. It must be at least
100,000.", NEW.m_name;
    END IF;
    RETURN NEW;
END;
' LANGUAGE plpgsql;
```

-- Create a trigger to execute the function when a movie is inserted

```
CREATE TRIGGER movie_insert_trigger
BEFORE INSERT ON Movie
FOR EACH ROW
EXECUTE FUNCTION check_movie_budget();
```

--Test the trigger

```
INSERT INTO Movie VALUES ('MovieD', 2025, 20000);
```