```
--Tables
Create table Users(
User_ID number primary KEY ,
User_name varchar2(100) ,
Pass  varchar2(255),
Email varchar2(100) ,
Role_type varchar2(50) Check (Role_type in ('USER','ADMIN')),
created_at date DEFAULT sysdate ,
ban    CHAR(1)    DEFAULT 'N'   NOT NULL
);
ALTER TABLE Users
ADD reported_count number DEFAULT 0;
CREATE table Movies(
movie_id number primary key ,
title varchar2(255),
genre varchar2(100),
released_date date,
description varchar2(2000)
);


CREATE TABLE Actors (
   actor_id   NUMBER PRIMARY KEY,
   actor_name VARCHAR2(255) NOT NULL
);
CREATE TABLE Movie_Actors (
```

```
  movie_id NUMBER,
  actor_id NUMBER,
  PRIMARY KEY (movie_id, actor_id),
  FOREIGN KEY (movie_id) REFERENCES Movies(movie_id),
  FOREIGN KEY (actor_id) REFERENCES Actors(actor_id)
);
CREATE TABLE Reviews (
  review_id NUMBER,
  movie_id NUMBER NOT NULL,
  user_id NUMBER NOT NULL,
  review_text VARCHAR2(2000),
  review_date DATE DEFAULT SYSDATE,
  is_bad CHAR(1) DEFAULT 'N' NOT NULL,
  PRIMARY KEY (user_id, movie_id),
  FOREIGN KEY (movie_id) REFERENCES Movies(movie_id),
  FOREIGN KEY (user_id) REFERENCES Users(User_ID),
  CONSTRAINT chk_reviews_is_bad CHECK (is_bad IN ('N', 'Y'))
);

ALTER TABLE Reviews ADD CONSTRAINT unique_REVIEW_ID UNIQUE (REVIEW_ID);
ALTER TABLE Reviews
ADD v_count number DEFAULT 0;

CREATE TABLE Ratings (
  rating_id NUMBER,
  movie_id NUMBER,
```

```
  user_id NUMBER,
  rating NUMBER(3,1) CHECK (rating BETWEEN 0.0 AND 10.0),
  rating_date DATE DEFAULT SYSDATE,
  PRIMARY KEY (user_id, movie_id),
  FOREIGN KEY (movie_id) REFERENCES Movies(movie_id),
  FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
ALTER TABLE Ratings ADD CONSTRAINT unique_rating_id UNIQUE (RATING_ID);
CREATE TABLE Watchlist (
  user_id number  ,
  movie_id number,
  added_date date DEFAULT sysdate,
  PRIMARY KEY (user_id, movie_id),
  FOREIGN KEY (user_id) REFERENCES Users(user_id),
  FOREIGN KEY (movie_id) REFERENCES Movies(movie_id)
);
ALTER TABLE WATCHLIST
ADD TITLE varchar2(255);
```

```
--SEQUENCES



CREATE SEQUENCE User_seq
START WITH 1
INCREMENT BY 1;
CREATE OR REPLACE TRIGGER User_trigger
BEFORE INSERT ON Users
FOR EACH ROW
BEGIN
  SELECT User_seq.nextval INTO :new.User_ID FROM dual;
END;



CREATE SEQUENCE movie_seq
START WITH 1
INCREMENT BY 1;
CREATE OR REPLACE TRIGGER movie_trigger
BEFORE INSERT ON Movies
FOR EACH ROW
BEGIN
  SELECT movie_seq.nextval INTO :new.movie_id FROM dual;
END;
CREATE SEQUENCE act_seq
```

```
START WITH 1

INCREMENT BY 1;

CREATE OR REPLACE TRIGGER act_trigger

BEFORE INSERT ON Actors

FOR EACH ROW

BEGIN

  SELECT act_seq.nextval INTO :new.ACTOR_ID FROM dual;

END;
```

```
--PROCEDURES


create or replace PROCEDURE search_movies_by_title (
   p_search_text IN VARCHAR2,
   p_resultset   OUT SYS_REFCURSOR
)
IS
BEGIN
   OPEN p_resultset FOR
      SELECT DISTINCT m.movie_id, m.title, m.genre, m.released_date, m.description
      FROM Movies m
      WHERE LOWER(m.title) LIKE '%' || LOWER(p_search_text) || '%';
END search_movies_by_title;



create or replace PROCEDURE ACT_NAMES (
   id_in IN NUMBER,
   P_RESULTSET OUT SYS_REFCURSOR
)
IS
BEGIN
   OPEN P_RESULTSET FOR
```

```
    SELECT DISTINCT a.actor_name
    FROM Actors a
    JOIN Movie_Actors ma ON a.actor_id = ma.actor_id
    WHERE ma.movie_id = id_in;
END ACT_NAMES;

create or replace PROCEDURE add_to_watchlist (
  p_movie_id IN NUMBER,
  p_user_id IN NUMBER,
  P_user_title in VARCHAR2,
  p_message OUT VARCHAR2
)
AS
  v_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_count
  FROM WATCHLIST
  WHERE MOVIE_ID = p_movie_id AND USER_ID = p_user_id;

  IF v_count > 0 THEN
    p_message := 'Movie already in watchlist.';
  ELSE
    INSERT INTO WATCHLIST (MOVIE_ID, USER_ID, title)
    VALUES (p_movie_id, p_user_id,P_user_title);
    p_message := 'Movie added to watchlist.';
  END IF;
```

```
END;



create or replace PROCEDURE submit_rating (
   p_movie_id IN NUMBER,
   p_user_id IN NUMBER,
   p_rating   IN NUMBER
) AS
   v_count INTEGER;
BEGIN

   IF p_rating < 0.0 OR p_rating > 10.0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Rating must be between 0.0 and 10.0');
   END IF;

   SELECT COUNT(*) INTO v_count
   FROM Ratings
   WHERE movie_id = p_movie_id AND user_id = p_user_id;

   IF v_count > 0 THEN

      UPDATE Ratings
      SET rating = p_rating,
         rating_date = SYSDATE
      WHERE movie_id = p_movie_id AND user_id = p_user_id;
   ELSE
```

```
      INSERT INTO Ratings (movie_id, user_id, rating)
      VALUES (p_movie_id, p_user_id, p_rating);
   END IF;

   COMMIT;
END;



create or replace PROCEDURE submit_Report (
   p_user_id IN NUMBER,
   p_review_id IN NUMBER
) AS
   v1_count INTEGER;
   v2_count INTEGER;
BEGIN
   SELECT v_count INTO v1_count
   FROM Reviews
   WHERE review_id = p_review_id;

   SELECT REPORTED_COUNT INTO v2_count
   FROM users
   WHERE user_id = p_user_id;

   UPDATE reviews
   SET v_count = v1_count + 1,
```

```
      is_bad = 'Y'
    WHERE review_id = p_review_id;

    UPDATE users
    SET REPORTED_COUNT = v2_count + 1
    WHERE user_id = p_user_id;

    COMMIT;
END;

create or replace PROCEDURE submit_review(
    p_user_id     IN NUMBER,
    p_movie_id    IN NUMBER,
    p_review_text  IN VARCHAR2
)
IS
    v_count NUMBER;
BEGIN

    SELECT COUNT(*) INTO v_count
    FROM Reviews
    WHERE user_id = p_user_id AND movie_id = p_movie_id;

    IF v_count = 0 THEN

      INSERT INTO Reviews (user_id, movie_id, review_text)
```

```
    VALUES (p_user_id, p_movie_id, p_review_text);
  ELSE

    UPDATE Reviews
    SET review_text = p_review_text,
      review_date = SYSDATE
    WHERE user_id = p_user_id AND movie_id = p_movie_id;
  END IF;

  COMMIT;
END;



create or replace PROCEDURE get_user(
  id_in IN NUMBER,
  USER_out OUT VARCHAR2,
  PASS_out OUT VARCHAR2,
  EMAIL_out OUT VARCHAR2,
  ROLE_TYPE_out OUT VARCHAR2,
  CREATED_AT_out OUT DATE,
  BAN_out OUT CHAR
) AS
BEGIN
  SELECT User_name, Pass, Email, Role_type, Created_at, Ban
  INTO USER_out, PASS_out, EMAIL_out, ROLE_TYPE_out, CREATED_AT_out, BAN_out
  FROM Users
```

```
   WHERE User_ID = id_in;
END get_user;


create or replace PROCEDURE sign_in(
   email_in IN VARCHAR2,
   password_in IN VARCHAR2,
   id_out OUT NUMBER
) AS
   temp NUMBER;
BEGIN
   SELECT user_id INTO temp
   FROM USERS
   WHERE email = email_in
   AND pass = password_in;

   id_out := temp;
END sign_in;
```