```sql
-- 1 Создаём таблицу пользователей
DROP TABLE IF EXISTS users;
CREATE TABLE users (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY COMMENT
"Идентификатор строки",
  first_name VARCHAR(100) NOT NULL COMMENT "Имя пользователя",
  last_name VARCHAR(100) NOT NULL COMMENT "Фамилия пользователя",
  email VARCHAR(100) NOT NULL UNIQUE COMMENT "Почта",
  phone VARCHAR(100) NOT NULL UNIQUE COMMENT "Телефон",
  pswd_hash VARCHAR(100) NOT NULL UNIQUE COMMENT "pasword",
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP COMMENT "Время
создания строки"
) COMMENT "Пользователи";

-- 2 Таблица профилей
DROP TABLE IF EXISTS profiles;
CREATE TABLE profiles (
  user_id INT UNSIGNED NOT NULL PRIMARY KEY COMMENT "Ссылка на
пользователя",
  photo_id INT DEFAULT NULL UNIQUE COMMENT "",
  sex CHAR(1) DEFAULT NULL COMMENT "Пол",
  birthday DATE DEFAULT NULL COMMENT "Дата рождения",
  city_id INT UNSIGNED DEFAULT NULL COMMENT "Ссылка на город
проживания",
  country_id INT UNSIGNED DEFAULT NULL COMMENT "Ссылка на город
проживания",
  films_watched INT UNSIGNED DEFAULT NULL COMMENT "film",
  serials_watched INT UNSIGNED DEFAULT NULL COMMENT "serial",
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP COMMENT "Время
создания строки"
) COMMENT "Профили";

-- 3 Таблица городов
DROP TABLE IF EXISTS cities;
CREATE TABLE cities (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(150) NOT NULL UNIQUE,
  country_id INT UNSIGNED
);

-- 4 Таблица стран
DROP TABLE IF EXISTS countries;
CREATE TABLE countries (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(150) NOT NULL UNIQUE
);

-- 5 Таблица сообщений
DROP TABLE IF EXISTS messages;
CREATE TABLE messages (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY COMMENT
"Идентификатор строки",
```

```sql
  from_user_id INT UNSIGNED NOT NULL COMMENT "Ссылка на
отправителя сообщения",
  to_user_id INT UNSIGNED NOT NULL COMMENT "Ссылка на получателя
сообщения",
  body TEXT NOT NULL COMMENT "Текст сообщения",
  is_delivered BOOLEAN COMMENT "Признак доставки",
  is_read BOOLEAN,
  created_at DATETIME DEFAULT NOW() COMMENT "Время создания
строки"
) COMMENT "Сообщения";

-- 6 Таблица дружбы
DROP TABLE IF EXISTS friendship;
CREATE TABLE friendship (
  user_id INT UNSIGNED NOT NULL COMMENT "Ссылка на инициатора
дружеских отношений",
  friend_id INT UNSIGNED NOT NULL COMMENT "Ссылка на получателя
приглашения дружить",
  friendship_status ENUM('requested', 'approved', 'declined',
'unfriended') COMMENT "Ссылка на статус (текущее состояние)
отношений",
  confirmed_at DATETIME COMMENT "Время подтверждения приглашения",
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP COMMENT "Время
создания строки",
  PRIMARY KEY (user_id, friend_id) COMMENT "Составной первичный
ключ"
) COMMENT "Таблица дружбы";

-- 7
DROP TABLE IF EXISTS films;
CREATE TABLE films (
  id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY COMMENT
"",
  title VARCHAR(50) COMMENT "",
  genre_id INT UNSIGNED COMMENT "",
  years DATE COMMENT "",
  rating TINYINT DEFAULT NULL COMMENT "",
  is_in_cinema BOOLEAN COMMENT "",
  price DECIMAL(10,2) NOT NULL COMMENT ""
);

-- 8
DROP TABLE IF EXISTS serials;
CREATE TABLE serials (
  id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY COMMENT
"",
  title VARCHAR(50) COMMENT "",
  genre_id VARCHAR(50) COMMENT "",
  years SMALLINT COMMENT "",
  rating TINYINT DEFAULT NULL COMMENT "",
  season TINYINT DEFAULT NULL COMMENT ""
);
```

```sql
-- 9
DROP TABLE IF EXISTS genres;
CREATE TABLE genres (
     id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY COMMENT
"",
     genre VARCHAR(100) UNIQUE
);

-- 10
DROP TABLE IF EXISTS rated;
CREATE TABLE rated (
  id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY COMMENT
"",
  film_id BIGINT UNSIGNED NOT NULL COMMENT "",
  serials_id BIGINT UNSIGNED NOT NULL COMMENT "",
  user_id BIGINT UNSIGNED NOT NULL COMMENT "",
  rating TINYINT NOT NULL COMMENT "",
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP COMMENT ""
);

-- 11
DROP TABLE IF EXISTS news;
CREATE TABLE news (
  id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY COMMENT
"",
  title VARCHAR(50) COMMENT "",
  body TEXT COMMENT "",
  film_id BIGINT UNSIGNED DEFAULT NULL COMMENT "",
  serials_id BIGINT UNSIGNED DEFAULT NULL COMMENT "",
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP COMMENT ""
);

-- 12
DROP TABLE IF EXISTS cinemas;
CREATE TABLE cinemas (
  id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY COMMENT
"",
  name VARCHAR(50) COMMENT "",
  adress VARCHAR(50) COMMENT "",
  phone VARCHAR(20) DEFAULT NULL COMMENT ""
);

-- 13
DROP TABLE IF EXISTS afisha;
CREATE TABLE afisha (
  id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY COMMENT
"",
  film_id BIGINT UNSIGNED NOT NULL COMMENT "",
  in_cinema_untill DATE COMMENT "",
  cinema_id BIGINT UNSIGNED NOT NULL COMMENT ""
);
```

```sql
-- 14
DROP TABLE IF EXISTS reviews;
CREATE TABLE reviews (
  id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY COMMENT
"",
  user_id BIGINT UNSIGNED NOT NULL COMMENT "",
  film_id BIGINT UNSIGNED NOT NULL COMMENT "",
  serials_id BIGINT UNSIGNED NOT NULL COMMENT "",
  body TEXT COMMENT "",
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP COMMENT ""
);

-- 15
DROP TABLE IF EXISTS celebs;
CREATE TABLE celebs (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY COMMENT "",
  first_name VARCHAR(100) NOT NULL COMMENT "",
  last_name VARCHAR(100) NOT NULL COMMENT "",
  created_at DATETIME DEFAULT NOW() COMMENT "",
  updated_at DATETIME DEFAULT NOW() ON UPDATE NOW() COMMENT ""
);

-- 16
DROP TABLE IF EXISTS celebs_profiles;
CREATE TABLE celebs_profiles (
  celeb_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY
COMMENT "",
  sex CHAR(1) NOT NULL COMMENT "",
  birthday DATE NOT NULL COMMENT "",
  city_id VARCHAR(100) NOT NULL COMMENT "",
  country_id VARCHAR(100) NOT NULL COMMENT "",
  about TEXT NOT NULL COMMENT "",
  photo_id INT UNSIGNED COMMENT ""
);

-- 17
DROP TABLE IF EXISTS roles;
CREATE TABLE roles (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY COMMENT "",
  roles VARCHAR(100) UNIQUE COMMENT "",
  created_at DATETIME DEFAULT NOW() COMMENT ""
);

-- 18
DROP TABLE IF EXISTS works_of_celebs;
CREATE TABLE works_of_celebs (
  film_id INT UNSIGNED NOT NULL,
  serials_id INT UNSIGNED NOT NULL,
  celeb_id INT UNSIGNED NOT NULL,
  role_id INT UNSIGNED NOT NULL,
  PRIMARY KEY (film_id, serials_id, celeb_id, role_id)
```

```
);


SELECT * FROM users;
desc profiles;
SELECT * FROM profiles;
ALTER TABLE profiles MODIFY COLUMN photo_id INT DEFAULT NULL
UNIQUE;
UPDATE profiles SET photo_id = null WHERE photo_id BETWEEN 100 and
200;
UPDATE profiles SET photo_id = FLOOR(1 + RAND() * 500);
UPDATE profiles SET photo_id = 999 WHERE user_id = 10;
DROP TEMPORARY TABLE IF EXISTS gender;
CREATE TEMPORARY TABLE gender (gender CHAR(1));
INSERT INTO gender VALUES ('M'), ('F');
UPDATE profiles SET sex = ( SELECT gender FROM gender ORDER BY
RAND() LIMIT 1);
SELECT * FROM cities;
SELECT * FROM countries;
desc messages;
SELECT * FROM messages;
SELECT * FROM messages WHERE from_user_id =  to_user_id;
UPDATE messages SET
  from_user_id = FLOOR(1 + RAND() * 100),
  to_user_id = FLOOR(1 + RAND() * 100);
SELECT * FROM messages WHERE from_user_id =  to_user_id;
UPDATE messages SET to_user_id = 100 WHERE id = 40;

SELECT * FROM friendship;
SELECT * FROM friendship WHERE confirmed_at < created_at;
UPDATE friendship SET created_at = NOW() WHERE confirmed_at <
created_at;
SELECT * FROM friendship WHERE user_id = friend_id and friend_id =
user_id;
UPDATE friendship SET
  user_id = FLOOR(1 + RAND() * 100),
  friend_id = FLOOR(1 + RAND() * 100);
UPDATE friendship SET friend_id = 3 WHERE user_id = friend_id ;
desc films;
SELECT * FROM films;
ALTER TABLE films MODIFY COLUMN rating INT DEFAULT NULL;
ALTER TABLE films MODIFY COLUMN is_in_cinema VARCHAR(10) DEFAULT
NULL;
UPDATE films SET genre_id = FLOOR(1 + RAND() * 13);
UPDATE films SET rating = FLOOR(1 + RAND() * 1000);
UPDATE films SET is_in_cinema = "no" WHERE is_in_cinema != "yes";
UPDATE films SET price = FLOOR(350.00 + RAND() * 1200.00) WHERE
is_in_cinema = "yes";
UPDATE films SET price = 0 WHERE is_in_cinema = "no";

desc serials;
SELECT * FROM serials;
```

```sql
ALTER TABLE serials MODIFY COLUMN rating INT DEFAULT NULL;
ALTER TABLE serials MODIFY COLUMN genre_id INT UNSIGNED DEFAULT
NULL;
UPDATE serials SET genre_id = FLOOR(1 + RAND() * 13);
UPDATE serials SET rating = FLOOR(1 + RAND() * 1000);

SELECT * FROM genres;

desc genres;
UPDATE genres SET genre = 'Action' WHERE id = 1;
UPDATE genres SET genre = 'Adventure' WHERE id = 2;
UPDATE genres SET genre = 'Animation' WHERE id = 3;
UPDATE genres SET genre = 'Comedy' WHERE id = 4;
UPDATE genres SET genre = 'Crime' WHERE id = 5;
UPDATE genres SET genre = 'Drama' WHERE id = 6;
UPDATE genres SET genre = 'Fantasy' WHERE id = 7;
UPDATE genres SET genre = 'History' WHERE id = 8;
UPDATE genres SET genre = 'Horror' WHERE id = 9;
UPDATE genres SET genre = 'Sci-Fi' WHERE id = 10;
UPDATE genres SET genre = 'Thriller' WHERE id = 11;
UPDATE genres SET genre = 'War' WHERE id = 12;
UPDATE genres SET genre = 'Western' WHERE id = 13;
delete from genres where id >= 14;

SELECT * FROM rated;
desc rated ;
ALTER TABLE rated MODIFY COLUMN rating INT DEFAULT NULL;
ALTER TABLE rated MODIFY COLUMN film_id INT DEFAULT NULL;
ALTER TABLE rated MODIFY COLUMN serials_id INT DEFAULT NULL;

UPDATE rated SET
  film_id = FLOOR(1 + RAND() * 100),
  serials_id = FLOOR(1 + RAND() * 100),
  user_id = FLOOR(1 + RAND() * 100),
  rating = FLOOR(1 + RAND() * 1000);
UPDATE rated SET film_id = NULL WHERE id < 50;
UPDATE rated SET film_id = 5 WHERE id = 50;
UPDATE rated SET serials_id = NULL WHERE id >= 50;


desc news;
SELECT * FROM news;
ALTER TABLE news MODIFY COLUMN film_id BIGINT UNSIGNED DEFAULT
NULL;
ALTER TABLE news MODIFY COLUMN serials_id BIGINT UNSIGNED DEFAULT
NULL;
UPDATE news SET
  film_id = FLOOR(1 + RAND() * 100);
UPDATE news SET film_id = NULL WHERE serials_id IS NOT NULL;

SELECT * FROM cinemas;
```

```sql
DELETE n1 FROM cinemas n1, cinemas n2 WHERE n1.id > n2.id AND
n1.name = n2.name;
DELETE FROM cinemas WHERE CHAR_LENGTH(name) < 5;

SELECT * FROM afisha;
UPDATE afisha SET
  film_id = FLOOR(1 + RAND() * 100),
  cinema_id = FLOOR(1 + RAND() * 50);
desc reviews ;
SELECT * FROM reviews;
UPDATE reviews SET
  film_id = FLOOR(1 + RAND() * 100),
  serials_id = FLOOR(1 + RAND() * 100);
UPDATE reviews SET user_id = FLOOR(1 + RAND() * 100);
ALTER TABLE reviews MODIFY COLUMN film_id BIGINT UNSIGNED DEFAULT
NULL;
ALTER TABLE reviews MODIFY COLUMN serials_id BIGINT UNSIGNED
DEFAULT NULL;
UPDATE reviews SET film_id = NULL WHERE serials_id = FLOOR(1 +
RAND() * 100);
UPDATE reviews SET serials_id = NULL WHERE film_id IS NOT NULL;

SELECT * FROM celebs;
SELECT * FROM celebs WHERE updated_at < created_at;
UPDATE celebs SET updated_at = NOW() WHERE updated_at <
created_at;

desc celebs_profiles;
SELECT * FROM celebs_profiles;
SELECT * FROM celebs_profiles n1,  celebs_profiles n2 WHERE
n1.celeb_id > n2.celeb_id AND n1.photo_id = n2.photo_id;
DELETE FROM cinemas WHERE CHAR_LENGTH(name) < 5;;
ALTER TABLE celebs_profiles MODIFY COLUMN photo_id INT UNSIGNED
DEFAULT NULL UNIQUE;
UPDATE celebs_profiles SET photo_id = FLOOR(RAND() * 500);
UPDATE celebs_profiles SET sex = ( SELECT gender FROM gender ORDER
BY RAND() LIMIT 1);
SELECT * FROM roles;
DELETE FROM roles WHERE CHAR_LENGTH(roles) < 5;
SELECT * FROM works_of_celebs;
UPDATE works_of_celebs SET
  film_id = FLOOR(1 + RAND() * 100),
  serials_id = FLOOR(1 + RAND() * 100),
  celeb_id = FLOOR(1 + RAND() * 100),
  role_id = FLOOR(1 + RAND() * 98);
UPDATE works_of_celebs SET role_id = FLOOR(1 + RAND() * 47);

SELECT * FROM profiles;
ALTER TABLE films MODIFY COLUMN id INT UNSIGNED NOT NULL;
ALTER TABLE serials MODIFY COLUMN id INT UNSIGNED NOT NULL;
ALTER TABLE news MODIFY COLUMN film_id INT UNSIGNED DEFAULT NULL;
```

```
ALTER TABLE news MODIFY COLUMN serials_id INT UNSIGNED DEFAULT
NULL;
ALTER TABLE afisha MODIFY COLUMN film_id INT UNSIGNED DEFAULT
NULL;
ALTER TABLE reviews MODIFY COLUMN user_id INT UNSIGNED DEFAULT
NULL;
ALTER TABLE reviews MODIFY COLUMN film_id INT UNSIGNED DEFAULT
NULL;
ALTER TABLE reviews MODIFY COLUMN serials_id INT UNSIGNED DEFAULT
NULL;
ALTER TABLE celebs_profiles MODIFY COLUMN city_id INT UNSIGNED
DEFAULT NULL;
ALTER TABLE celebs_profiles MODIFY COLUMN country_id INT UNSIGNED
DEFAULT NULL;
ALTER TABLE rated MODIFY COLUMN user_id INT UNSIGNED DEFAULT NULL;
ALTER TABLE rated MODIFY COLUMN film_id INT UNSIGNED DEFAULT NULL;
ALTER TABLE rated MODIFY COLUMN serials_id INT UNSIGNED DEFAULT
NULL;



desc profiles;
 ALTER TABLE profiles
  ADD CONSTRAINT profiles_user_id_fk
    FOREIGN KEY (user_id) REFERENCES users(id)
    ON DELETE CASCADE,
  ADD CONSTRAINT profiles_city_id_fk
    FOREIGN KEY (city_id) REFERENCES cities(id)
    ON DELETE SET NULL,
  ADD CONSTRAINT profiles_country_id_fk
    FOREIGN KEY (country_id) REFERENCES countries(id)
    ON DELETE SET NULL,
  ADD CONSTRAINT profiles_films_id_fk
    FOREIGN KEY (films_watched) REFERENCES films(id)
    ON DELETE SET NULL,
  ADD CONSTRAINT profiles_serials_id_fk
    FOREIGN KEY (serials_watched) REFERENCES serials(id)
    ON DELETE SET NULL;

SELECT * FROM cities;
ALTER TABLE cities
  ADD CONSTRAINT cities_country_id_fk
    FOREIGN KEY (country_id) REFERENCES countries(id);

SELECT * FROM messages;
ALTER TABLE messages
  ADD CONSTRAINT messages_from_user_id_fk
    FOREIGN KEY (from_user_id) REFERENCES users(id),
  ADD CONSTRAINT messages_to_user_id_fk
    FOREIGN KEY (to_user_id) REFERENCES users(id);

SELECT * FROM friendship;
```

```
ALTER TABLE friendship
  ADD CONSTRAINT friendship_user_id_fk
    FOREIGN KEY (user_id) REFERENCES users(id),
  ADD CONSTRAINT friendship_friend_id_fk
    FOREIGN KEY (friend_id) REFERENCES users(id)
      ON DELETE CASCADE;


SELECT * FROM films;
ALTER TABLE films
  ADD CONSTRAINT films_genre_id_fk
    FOREIGN KEY (genre_id) REFERENCES genres(id);

desc films ;
SELECT * FROM serials;
ALTER TABLE serials
  ADD CONSTRAINT serials_genre_id_fk
    FOREIGN KEY (genre_id) REFERENCES genres(id);

   desc news;
SELECT * FROM news;
ALTER TABLE news
  ADD CONSTRAINT news_films_id_fk
    FOREIGN KEY (film_id) REFERENCES films(id),
  ADD CONSTRAINT news_serials_id_fk
    FOREIGN KEY (serials_id) REFERENCES serials(id);

   desc afisha ;
SELECT * FROM afisha;
ALTER TABLE afisha
  ADD CONSTRAINT afisha_films_id_fk
    FOREIGN KEY (film_id) REFERENCES films(id),
  ADD CONSTRAINT afisha_cinema_id_fk
    FOREIGN KEY (cinema_id) REFERENCES cinemas(id);

desc reviews ;
SELECT * FROM reviews;
ALTER TABLE reviews
  ADD CONSTRAINT reviews_user_id_fk
    FOREIGN KEY (user_id) REFERENCES users(id)
    ON DELETE CASCADE,
  ADD CONSTRAINT reviews_films_id_fk
    FOREIGN KEY (film_id) REFERENCES films(id),
  ADD CONSTRAINT reviews_serials_id_fk
    FOREIGN KEY (serials_id) REFERENCES serials(id);


desc celebs_profiles;
SELECT * FROM celebs_profiles;
 ALTER TABLE celebs_profiles
  ADD CONSTRAINT celebs_profiles_user_id_fk
    FOREIGN KEY (celeb_id) REFERENCES celebs(id)
```

```
    ON DELETE CASCADE,
  ADD CONSTRAINT celebs_profiles_city_id_fk
    FOREIGN KEY (city_id) REFERENCES cities(id),
  ADD CONSTRAINT celebs_profiles_country_id_fk
    FOREIGN KEY (country_id) REFERENCES countries(id);

   desc roles;
desc works_of_celebs;
SELECT * FROM works_of_celebs;
ALTER TABLE works_of_celebs
  ADD CONSTRAINT works_of_celebs_films_id_fk
    FOREIGN KEY (film_id) REFERENCES films(id),
  ADD CONSTRAINT works_of_celebs_serials_id_fk
    FOREIGN KEY (serials_id) REFERENCES serials(id),
  ADD CONSTRAINT works_of_celebs_celeb_id_fk
    FOREIGN KEY (celeb_id) REFERENCES celebs(id);
ALTER TABLE works_of_celebs
  ADD CONSTRAINT works_of_celebs_role_id_fk
    FOREIGN KEY (role_id) REFERENCES roles(id);


desc rated;
SELECT * FROM rated;
ALTER TABLE rated
  ADD CONSTRAINT rated_user_id_fk
    FOREIGN KEY (user_id) REFERENCES users(id)
    ON DELETE CASCADE,
  ADD CONSTRAINT rated_film_id_fk
    FOREIGN KEY (film_id) REFERENCES films(id),
  ADD CONSTRAINT rated_serials_id_fk
    FOREIGN KEY (serials_id) REFERENCES serials(id);

ALTER TABLE rated
  ADD CONSTRAINT rated_rating_f_fk
    FOREIGN KEY (rating) REFERENCES films(rating),
  ADD CONSTRAINT rated_rating_s_fk
    FOREIGN KEY (rating) REFERENCES serials(rating);
   tak nelzya delat ?

CREATE INDEX users_first_name_last_name_idx ON users(first_name,
last_name);
CREATE INDEX celebs_first_name_last_name_idx ON celebs(first_name,
last_name);
CREATE INDEX profiles_birthday_sex_idx ON profiles(birthday,sex);
CREATE INDEX celebs_profiles_birthday_sex_idx ON
celebs_profiles(birthday,sex);

CREATE INDEX films_years_film_idx ON films(years, title);
CREATE INDEX serials_years_film_idx ON serials(years, title);

CREATE INDEX films_genres_film_idx ON films(genre_id, title);
CREATE INDEX serials_genres_film_idx ON serials(genre_id, title);
```

```
SELECT * FROM news;
DROP TRIGGER IF EXISTS insert_null_news;

DELIMITER //

CREATE TRIGGER insert_null_news BEFORE INSERT ON news
    FOR EACH ROW
BEGIN
    IF (NEW.film_id IS NULL AND NEW.serials_id IS NULL)
     OR (NEW.film_id IS NOT NULL AND NEW.serials_id IS NOT NULL)
    THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'оба поля
принимают неопределенное значение NULL';
    END IF;
END//

DELIMITER ;

DROP TRIGGER IF EXISTS update_null_news;

DELIMITER //

CREATE TRIGGER update_null_news BEFORE UPDATE ON news
    FOR EACH ROW
BEGIN
    IF (NEW.film_id IS NULL AND NEW.serials_id IS NULL)
     OR (NEW.film_id IS NOT NULL AND NEW.serials_id IS NOT NULL)
        OR (OLD.film_id IS NULL AND NEW.serials_id IS NULL)
        OR (OLD.film_id IS NOT NULL AND NEW.serials_id IS NOT
NULL)
        OR (OLD.serials_id IS NULL AND NEW.film_id IS NULL)
        OR (OLD.serials_id IS NOT NULL AND NEW.film_id IS NOT
NULL)
    THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'оба поля
принимают неопределенное значение NULL';
    END IF;
END//

DELIMITER ;




SELECT * FROM rated;
DROP TRIGGER IF EXISTS insert_null_rated;

DELIMITER //

CREATE TRIGGER insert_null_rated BEFORE INSERT ON rated
```

```
    FOR EACH ROW
BEGIN
    IF (NEW.film_id IS NULL AND NEW.serials_id IS NULL)
     OR (NEW.film_id IS NOT NULL AND NEW.serials_id IS NOT NULL)
    THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'оба поля
принимают неопределенное значение NULL';
    END IF;
END//

DELIMITER ;

DROP TRIGGER IF EXISTS update_null_rated;

DELIMITER //

CREATE TRIGGER update_null_rated BEFORE UPDATE ON rated
    FOR EACH ROW
BEGIN
    IF (NEW.film_id IS NULL AND NEW.serials_id IS NULL)
     OR (NEW.film_id IS NOT NULL AND NEW.serials_id IS NOT NULL)
        OR (OLD.film_id IS NULL AND NEW.serials_id IS NULL)
        OR (OLD.film_id IS NOT NULL AND NEW.serials_id IS NOT
NULL)
        OR (OLD.serials_id IS NULL AND NEW.film_id IS NULL)
        OR (OLD.serials_id IS NOT NULL AND NEW.film_id IS NOT
NULL)
    THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'оба поля
принимают неопределенное значение NULL';
    END IF;
END//

DELIMITER ;

SELECT * FROM reviews;
DROP TRIGGER IF EXISTS insert_null_reviews;

DELIMITER //

CREATE TRIGGER insert_null_reviews BEFORE INSERT ON reviews
    FOR EACH ROW
BEGIN
    IF (NEW.film_id IS NULL AND NEW.serials_id IS NULL)
     OR (NEW.film_id IS NOT NULL AND NEW.serials_id IS NOT NULL)
    THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'оба поля
принимают неопределенное значение NULL';
    END IF;
END//

DELIMITER ;
```

```
DROP TRIGGER IF EXISTS update_null_reviews;

DELIMITER //

CREATE TRIGGER update_null_reviews BEFORE UPDATE ON reviews
    FOR EACH ROW
BEGIN
    IF (NEW.film_id IS NULL AND NEW.serials_id IS NULL)
     OR (NEW.film_id IS NOT NULL AND NEW.serials_id IS NOT NULL)
        OR (OLD.film_id IS NULL AND NEW.serials_id IS NULL)
        OR (OLD.film_id IS NOT NULL AND NEW.serials_id IS NOT
NULL)
        OR (OLD.serials_id IS NULL AND NEW.film_id IS NULL)
        OR (OLD.serials_id IS NOT NULL AND NEW.film_id IS NOT
NULL)
    THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'оба поля
принимают неопределенное значение NULL';
    END IF;
END//

DELIMITER ;
```