SQL assessment

Case Study : Movie Streaming Platform

Context:

A movie app wants insights into viewer engagement, genre preferences, and content ratings.

Schema:

```
CREATE DATABASE movie_app;
USE movie_app;

CREATE TABLE users (
   user_id INT PRIMARY KEY,
   name VARCHAR(50),
   country VARCHAR(50)
);

CREATE TABLE movies (
   movie_id INT PRIMARY KEY,
   title VARCHAR(100),
   genre VARCHAR(50),
   release_year INT
);

CREATE TABLE ratings (
   user_id INT,
   movie_id INT,
   rating INT,
   rating_date DATE
);
```

```
CREATE TABLE watch_history (
   user_id INT,
   movie_id INT,
   watch_date DATE
);

INSERT INTO users VALUES
(1, 'Amit', 'India'),
(2, 'Sara', 'USA'),
(3, 'John', 'UK'),
(4, 'Priya', 'India'),
(5, 'Ali', 'UAE'),
(6, 'Maria', 'USA');

INSERT INTO movies VALUES
(101, 'Action Hero', 'Action', 2022),
(102, 'Love Story', 'Romance', 2019),
(103, 'Comedy Nights', 'Comedy', 2021),
(104, 'SciFi World', 'Sci-Fi', 2023),
(105, 'Horror House', 'Horror', 2024);

INSERT INTO ratings VALUES
(1,101,5,'2023-01-10'),
(2,101,4,'2023-02-11'),
(3,102,5,'2023-03-12'),
(4,103,3,'2023-04-13'),
(5,104,5,'2023-05-14'),
(6,105,4,'2023-06-15');

INSERT INTO watch_history VALUES
(1,101,'2023-01-05'),
```

```
(1,102,'2023-01-07'),
(2,101,'2023-02-01'),
(3,102,'2023-02-03'),
(4,103,'2023-03-04'),
(5,104,'2023-03-05'),
(6,105,'2023-03-06'),
(1,101,'2023-04-01');
```

Q1. Top 3 most-watched genres per country.

```
SELECT u.country, m.genre, COUNT(*)
FROM users u
JOIN watch_history w ON u.user_id = w.user_id
JOIN movies m ON w.movie_id = m.movie_id
GROUP BY u.country, m.genre
ORDER BY total_watch DESC
LIMIT 3;
```

Q2. Identify users who rated more than 20 movies.

```
SELECT user_id, COUNT(movie_id)
FROM ratings
GROUP BY user_id
HAVING COUNT(movie_id) > 20;
```

Q3. Find movies released after 2020 that have never been watched.

```
SELECT movie_id, title
FROM movies
WHERE release_year > 2020
AND movie_id NOT IN (SELECT movie_id FROM watch_history);
```

Q4. Compute the average rating per genre.

```
SELECT m.genre, AVG(r.rating)
```

```
FROM movies m
JOIN ratings r ON m.movie_id = r.movie_id
GROUP BY m.genre;
```

Q5. List users who gave 5-star ratings to all movies in a genre.

```
SELECT r.user_id, m.genre
FROM ratings r
JOIN movies m ON r.movie_id = m.movie_id
WHERE r.rating = 5
GROUP BY r.user_id, m.genre;
```

Q6. Identify movies watched by users from at least 5 different countries.

```
SELECT w.movie_id, COUNT(DISTINCT u.country)
FROM watch_history w
JOIN users u ON w.user_id = u.user_id
GROUP BY w.movie_id
HAVING COUNT(DISTINCT u.country) >= 5;
```

Q7. Find the average number of movies watched per user per month.

```
SELECT user_id,
COUNT(movie_id) / COUNT(DISTINCT MONTH(watch_date))
FROM watch_history
GROUP BY user_id;
```

Q8. List users who watched the same movie more than once.

```
SELECT user_id, movie_id, COUNT(*)
FROM watch_history
GROUP BY user_id, movie_id
HAVING COUNT(*) > 1;
```

Q9. Count how many movies have ratings but were never watched.

```
SELECT DISTINCT movie_id
FROM ratings
WHERE movie_id NOT IN (SELECT movie_id FROM watch_history);
```

Q10. Identify the genre with the highest average 5-star rating count.

```
SELECT genre
FROM movies m
JOIN ratings r ON m.movie_id = r.movie_id
WHERE r.rating = 5
GROUP BY genre
ORDER BY COUNT(*) DESC
LIMIT 1;
```