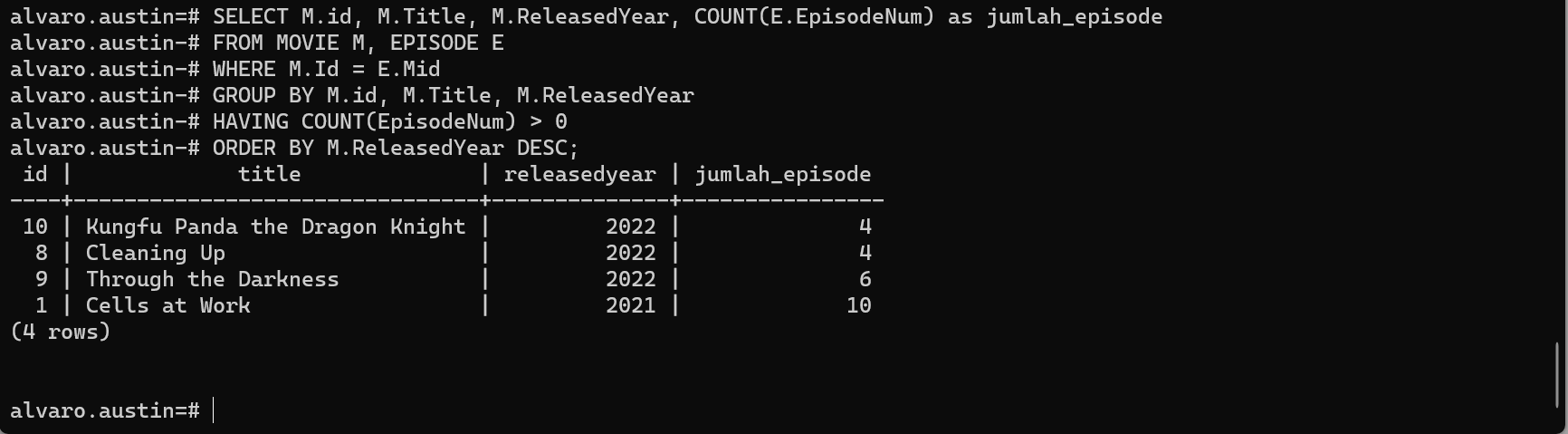
1. Untuk setiap movie, tampilkan id movie, judul movie, tahun rilis, dan jumlah episode nya, lalu urutkan dari tahun rilis terbaru hingga terlama. Untuk movie yang tidak memiliki episode tidak perlu ditampilkan



Code:

SELECT M.id, M.Title, M.ReleasedYear, COUNT(E.EpisodeNum) as jumlah\_episode

FROM MOVIE M, EPISODE E

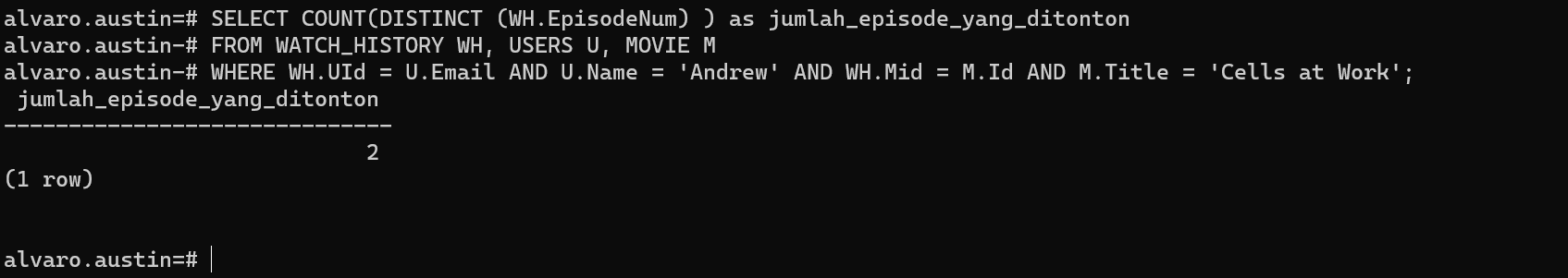
WHERE M.Id = E.Mid

GROUP BY M.id, M.Title, M.ReleasedYear

HAVING COUNT(EpisodeNum) > 0

ORDER BY M.ReleasedYear DESC;

1. Tampilkan jumlah episode dari movie “Cells at Work” yang telah ditonton oleh user bernama “Andrew”. Jika Andrew menonton episode yang sama lebih dari sekali maka tetap dihitung



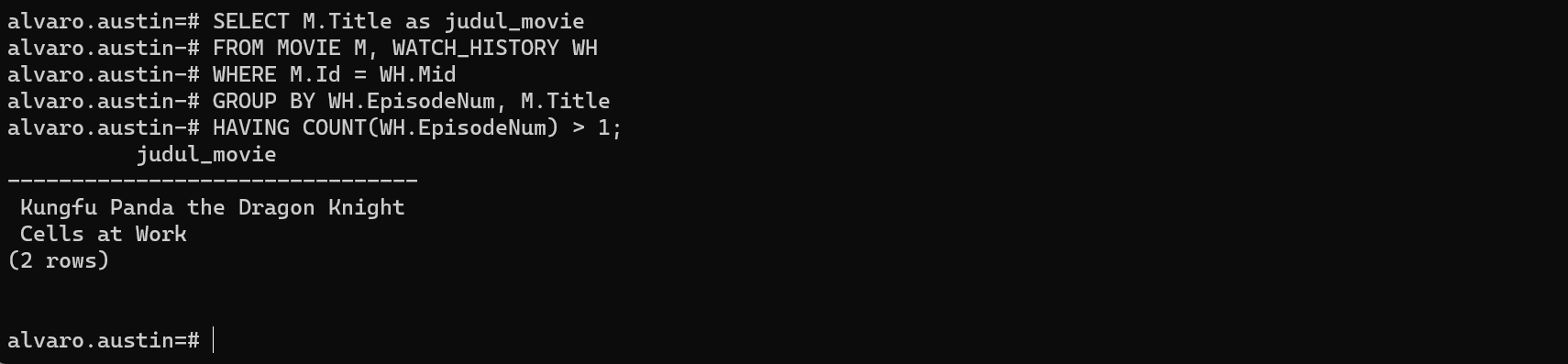
Code:

SELECT COUNT(DISTINCT (WH.EpisodeNum) ) as jumlah\_episode\_yang\_ditonton

FROM WATCH\_HISTORY WH, USERS U, MOVIE M

WHERE WH.UId = U.Email AND U.Name = 'Andrew' AND WH.Mid = M.Id AND M.Title = 'Cells at Work';

1. Tampilkan semua judul movie, dimana memiliki episode yang telah ditonton lebih dari 1 kali. Contoh: movie “Cells at Work” memiliki episode yang telah ditonton lebih dari 1 kali (pada season 2 episode 1).



Code:

SELECT M.Title as judul\_movie

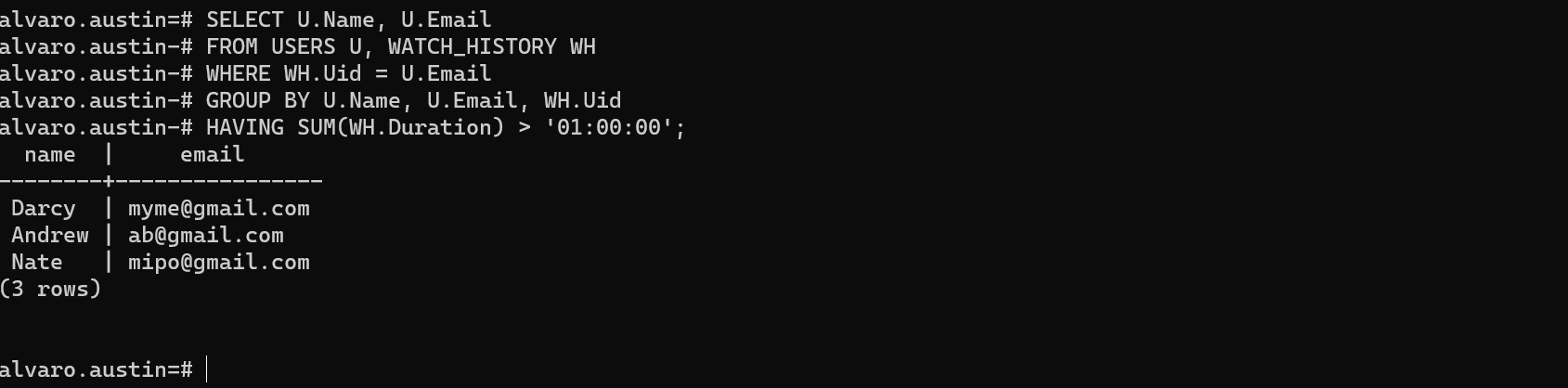
FROM MOVIE M, WATCH\_HISTORY WH

WHERE M.Id = WH.Mid

GROUP BY WH.EpisodeNum, M.Title

HAVING COUNT(WH.EpisodeNum) > 1;

1. Tampilkan nama dan email dari user yang yang durasi menontonnya selalu lebih dari 1 jam.



Code:

SELECT U.Name, U.Email

FROM USERS U, WATCH\_HISTORY WH

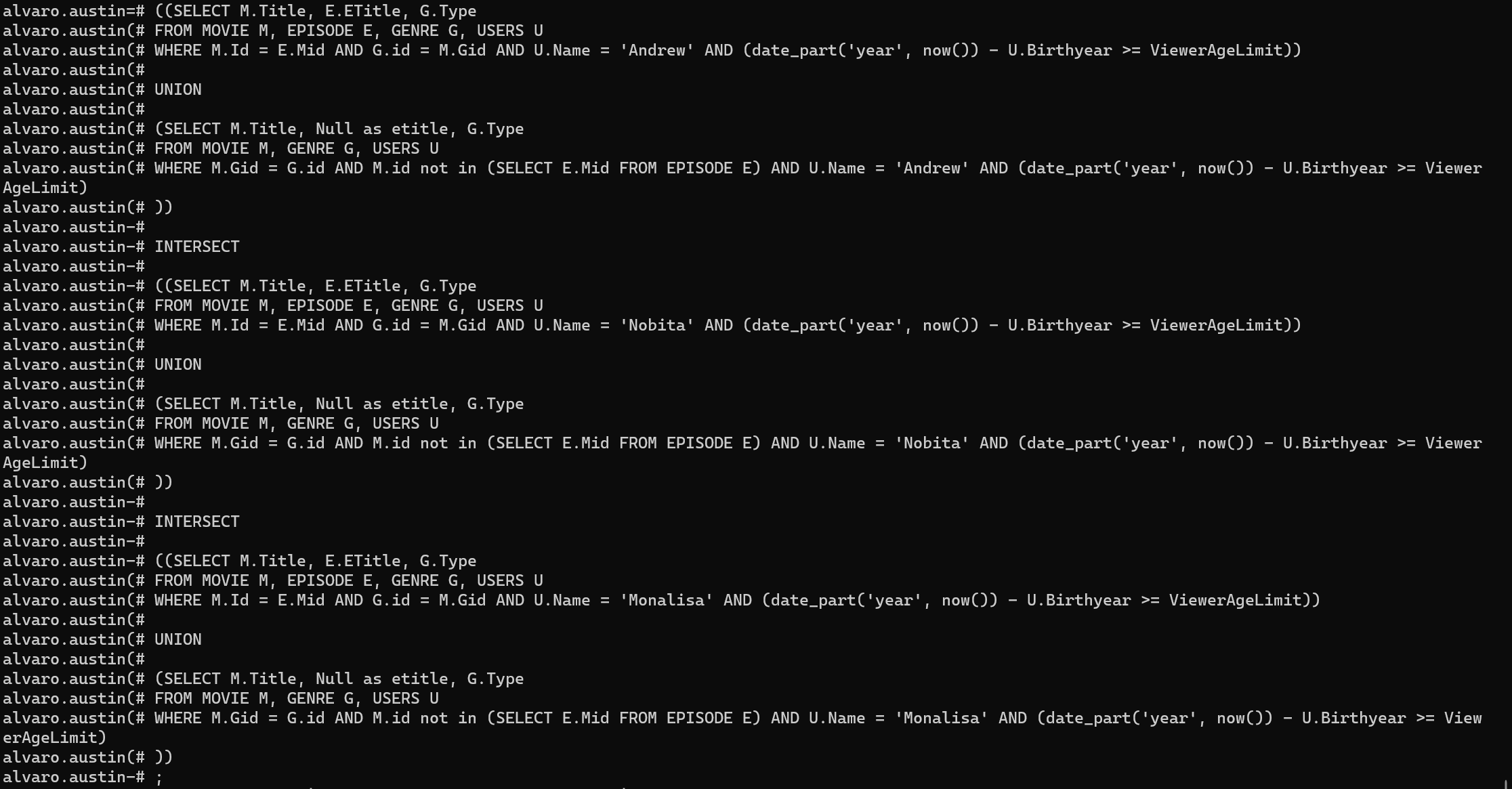
WHERE WH.Uid = U.Email

GROUP BY U.Name, U.Email, WH.Uid

HAVING SUM(WH.Duration) > '01:00:00';

1. Tampilkan semua informasi movie (judul movie, judul episode, nama genre) mana saja yang dapat ditonton oleh user bernama 'Andrew', 'Nobita', dan 'Monalisa' sesuai usia mereka. Perhatikan bahwa movie yang tidak memiliki episode yang memenuhi requirement ini juga tetap ditampilkan, dengan judul episode nya bernilai NULL Petunjuk Anda dapat menggunakan fungsi date\_part(‘year’, TANGGAL) untuk mendapatkan tahun dari suatu variabel TANGGAL yang bertipe date.

Asumsi: yang dimaksud adalah jumlah episode bernilai NULL





Code:

((SELECT M.Title, E.ETitle, G.Type

FROM MOVIE M, EPISODE E, GENRE G, USERS U

WHERE M.Id = E.Mid AND G.id = M.Gid AND U.Name = 'Andrew' AND (date\_part('year', now()) - U.Birthyear >= ViewerAgeLimit))

UNION

(SELECT M.Title, Null as etitle, G.Type

FROM MOVIE M, GENRE G, USERS U

WHERE M.Gid = G.id AND M.id not in (SELECT E.Mid FROM EPISODE E) AND U.Name = 'Andrew' AND (date\_part('year', now()) - U.Birthyear >= ViewerAgeLimit)

))

INTERSECT

((SELECT M.Title, E.ETitle, G.Type

FROM MOVIE M, EPISODE E, GENRE G, USERS U

WHERE M.Id = E.Mid AND G.id = M.Gid AND U.Name = 'Nobita' AND (date\_part('year', now()) - U.Birthyear >= ViewerAgeLimit))

UNION

(SELECT M.Title, Null as etitle, G.Type

FROM MOVIE M, GENRE G, USERS U

WHERE M.Gid = G.id AND M.id not in (SELECT E.Mid FROM EPISODE E) AND U.Name = 'Nobita' AND (date\_part('year', now()) - U.Birthyear >= ViewerAgeLimit)

))

INTERSECT

((SELECT M.Title, E.ETitle, G.Type

FROM MOVIE M, EPISODE E, GENRE G, USERS U

WHERE M.Id = E.Mid AND G.id = M.Gid AND U.Name = 'Monalisa' AND (date\_part('year', now()) - U.Birthyear >= ViewerAgeLimit))

UNION

(SELECT M.Title, Null as etitle, G.Type

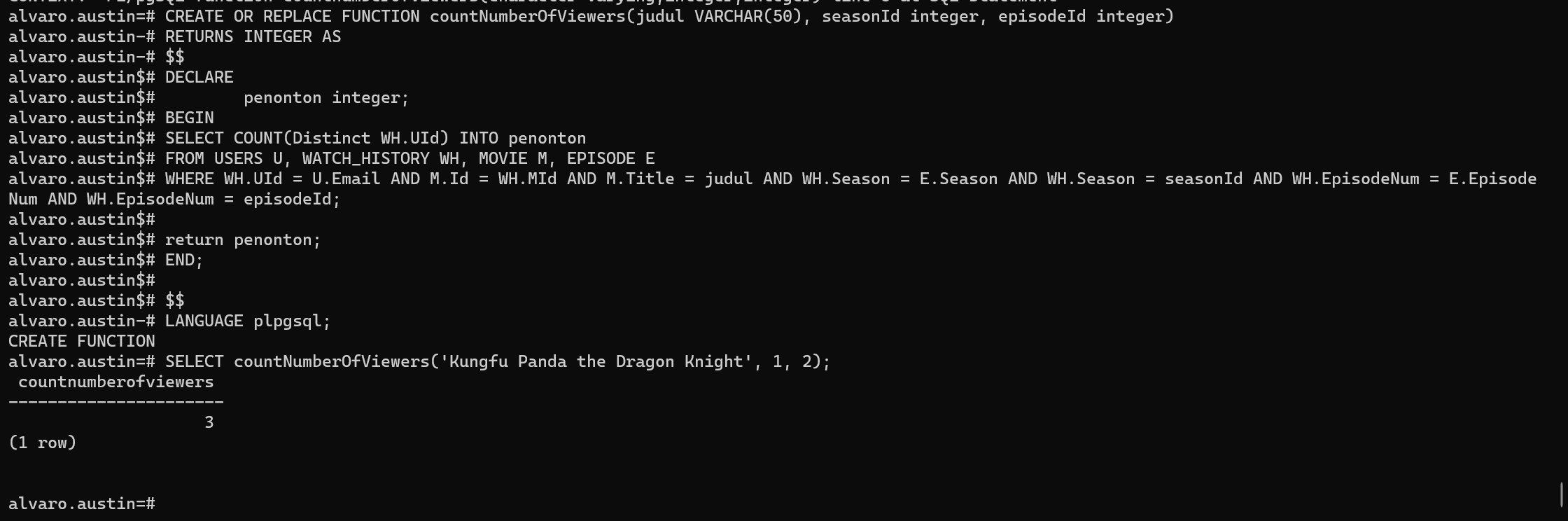
FROM MOVIE M, GENRE G, USERS U

WHERE M.Gid = G.id AND M.id not in (SELECT E.Mid FROM EPISODE E) AND U.Name = 'Monalisa' AND (date\_part('year', now()) - U.Birthyear >= ViewerAgeLimit)

))

;

1. Buatlah stored procedure countNumberOfViewers untuk menghitung jumlah user yang telah menonton movie untuk suatu season dan episode tertentu. Stored procedure ini menerima 3 buah argumen, yaitu: judul movie, nomor season, dan nomor episode. Asumsi nilai judul movie, nomor season dan nomor episode yang diinputkan pada fungsi ini pasti bukan NULL. Contoh: jumlah viewer (user yang telah menonton) film Kungfu Panda The Dragon Knight season 1 episode 2 adalah 3 orang. Perhatikan bahwa jika seorang user yang sama menonton suatu episode berkali-kali maka akan tetap dihitung sebagai 1 viewer. Setelah stored procedure dibuat, lakukan pengetesan dengan memanggil: SELECT countNumberOfViewers(‘Kungfu Panda the Dragon Knight’, 1, 2);



Code:

CREATE OR REPLACE FUNCTION countNumberOfViewers(judul VARCHAR(50), seasonId integer, episodeId integer)

RETURNS INTEGER AS

$$

DECLARE

penonton integer;

BEGIN

SELECT COUNT(Distinct WH.UId) INTO penonton

FROM USERS U, WATCH\_HISTORY WH, MOVIE M, EPISODE E

WHERE WH.UId = U.Email AND M.Id = WH.MId AND M.Title = judul AND WH.Season = E.Season AND WH.Season = seasonId AND WH.EpisodeNum = E.EpisodeNum AND WH.EpisodeNum = episodeId;

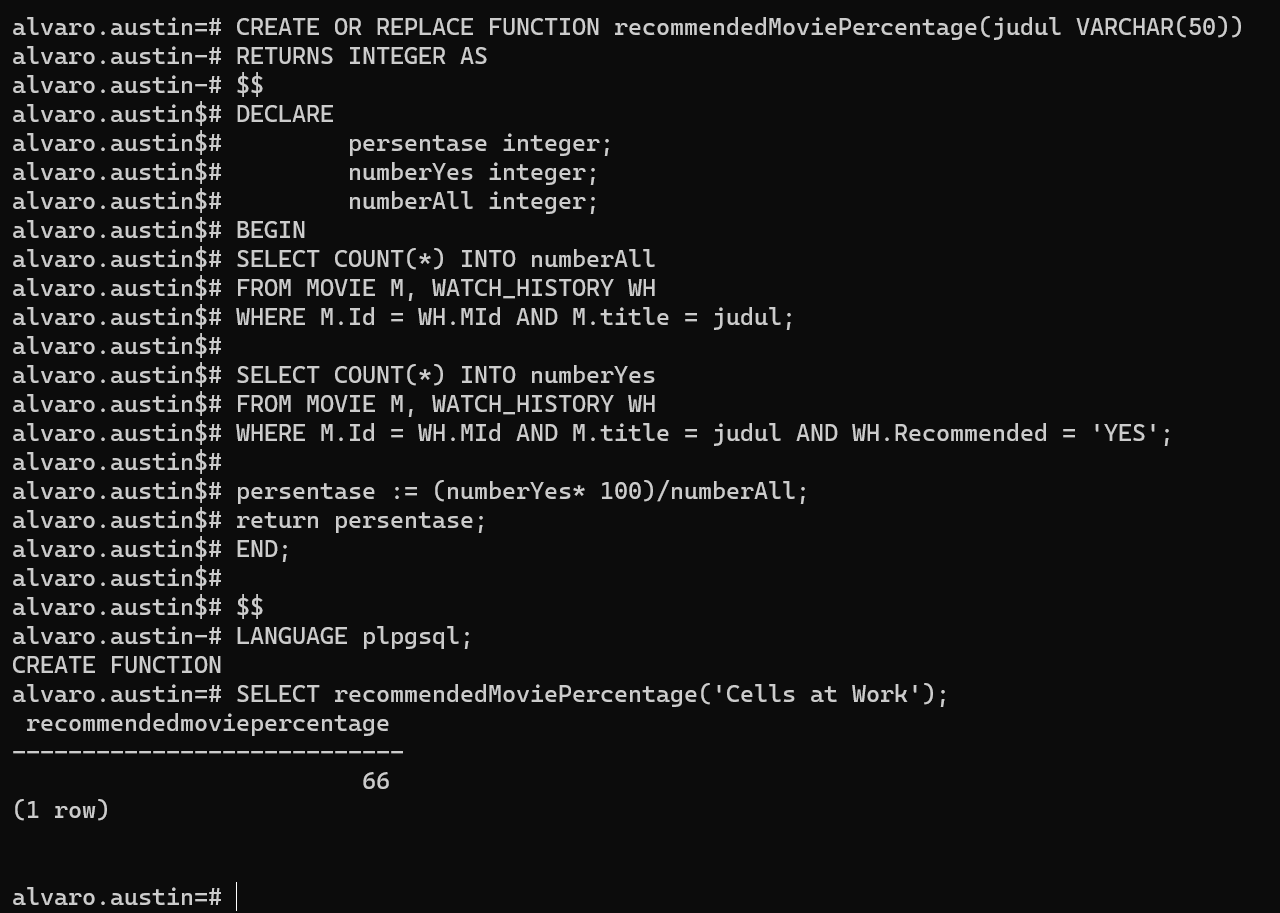
return penonton;

END;

$$

LANGUAGE plpgsql;

1. Buatlah stored procedure recommendedMoviePercentage untuk menghitung persentase jumlah movie direkomendasikan (recommended=”YES”) oleh user yang sudah menonton movie tertentu. Stored procedure ini menerima 1 buah argumen, yaitu judul movie. Contoh: pemanggilan recommendedMoviePercentage(‘Cells at Work’) akan mengembalikan nilai 66 karena ada 2 nilai rekomendasi ”YES” dari total 3 nilai rekomendasi yang diberikan untuk movie “Cells at Work”, sehingga hasilnya menjadi 100% \* (⅔) = 66%. Setelah stored procedure dibuat, lakukan pengetesan dengan memanggil: SELECT recommendedMoviePercentage(‘Cells at Work’);



Code:

CREATE OR REPLACE FUNCTION recommendedMoviePercentage(judul VARCHAR(50))

RETURNS INTEGER AS

$$

DECLARE

persentase integer;

numberYes integer;

numberAll integer;

BEGIN

SELECT COUNT(\*) INTO numberAll

FROM MOVIE M, WATCH\_HISTORY WH

WHERE M.Id = WH.MId AND M.title = judul;

SELECT COUNT(\*) INTO numberYes

FROM MOVIE M, WATCH\_HISTORY WH

WHERE M.Id = WH.MId AND M.title = judul AND WH.Recommended = 'YES';

persentase := (numberYes\* 100)/numberAll;

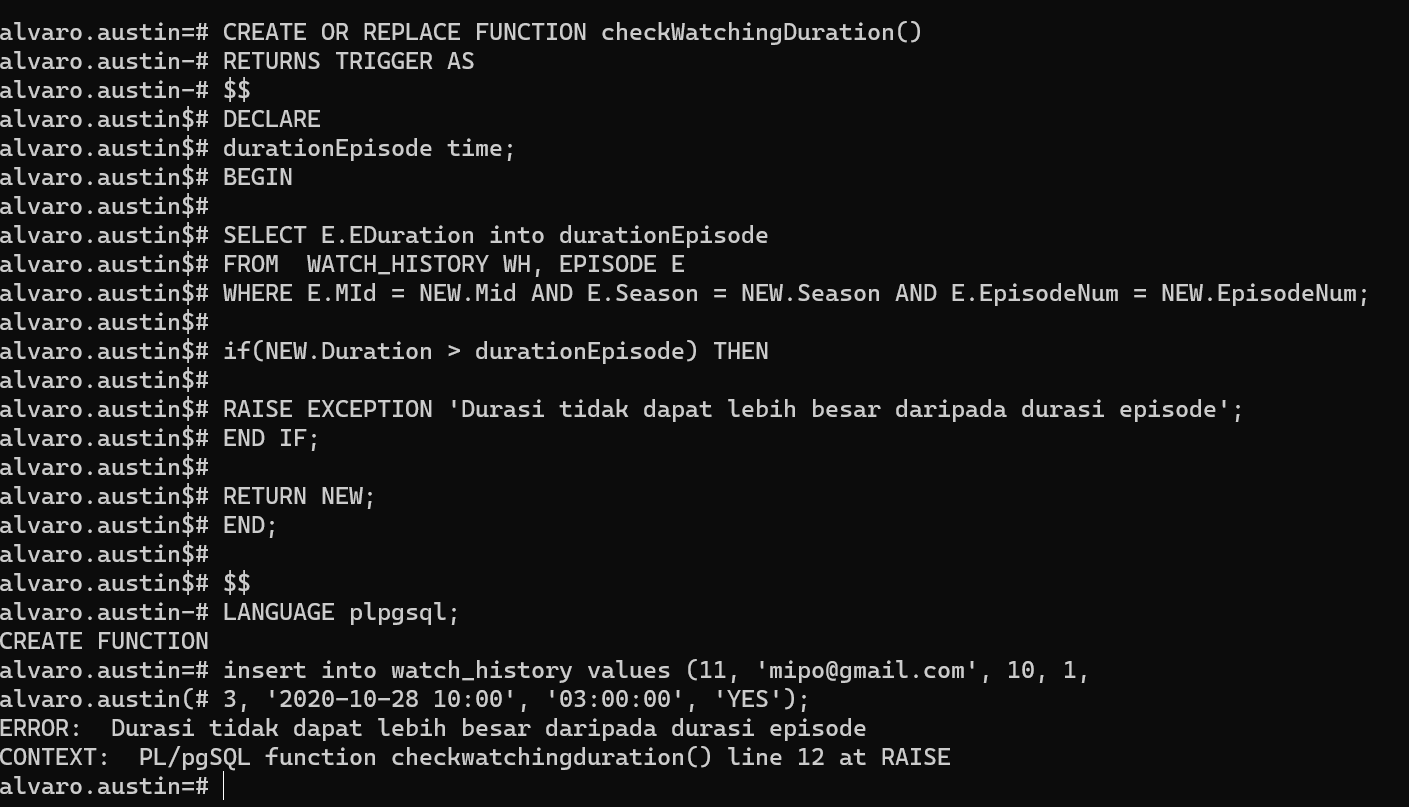
return persentase;

END;

$$

LANGUAGE plpgsql;

1. Buatlah stored procedure & trigger checkWatchingDuration untuk memastikan bahwa durasi menonton suatu episode movie dari seorang pengguna tidak mungkin lebih besar dari durasi episode itu sendiri. Perhatikan event apa saja yang perlu mengaktifkan trigger yang Anda buat. Setelah stored procedure & trigger dibuat, lakukan pengetesan untuk satu perintah SQL berikut ini: insert into watch\_history values (11, 'mipo@gmail.com', 10, 1, 3, '2020-10-28 10:00', '03:00:00', 'YES')



Code:

CREATE OR REPLACE FUNCTION checkWatchingDuration()

RETURNS TRIGGER AS

$$

DECLARE

durationEpisode time;

BEGIN

SELECT E.EDuration into durationEpisode

FROM WATCH\_HISTORY WH, EPISODE E

WHERE E.MId = NEW.Mid AND E.Season = NEW.Season AND E.EpisodeNum = NEW.EpisodeNum;

if(NEW.Duration > durationEpisode) THEN

RAISE EXCEPTION 'Durasi tidak dapat lebih besar daripada durasi episode';

END IF;

RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE TRIGGER triggerCheckWatchingDuration

BEFORE INSERT OR UPDATE ON WATCH\_HISTORY

FOR EACH ROW EXECUTE PROCEDURE checkWatchingDuration();