

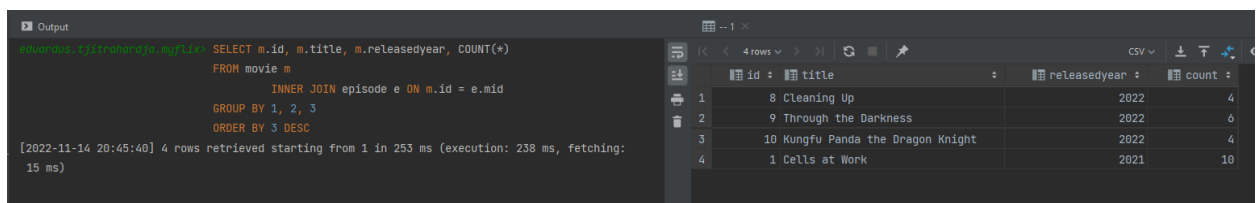
Nama: Eduardus Tjitrahardja

Kelas: Basdat A

NPM: 2106653602

Quiz 2: Advance SQL Query and Trigger & Stored Procedure

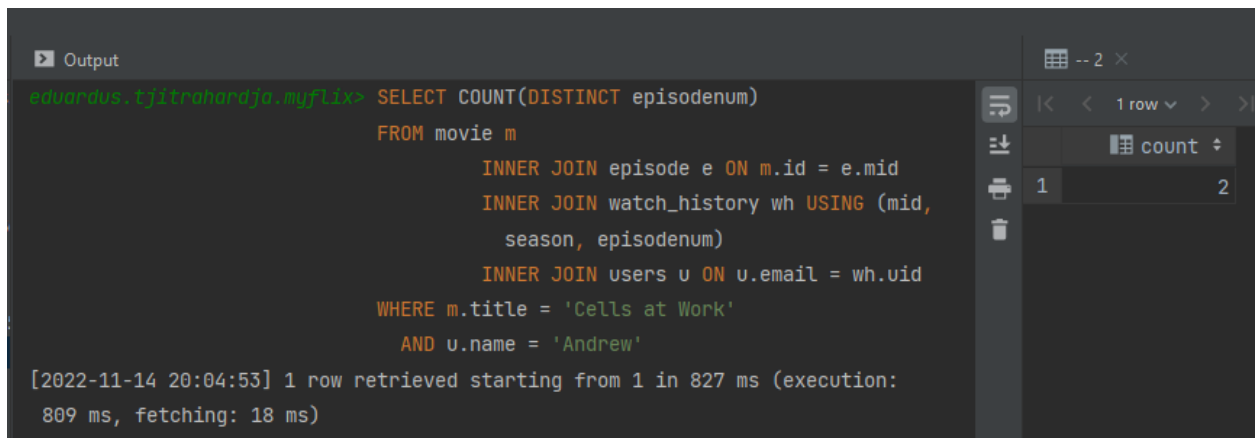
1. Untuk setiap movie, tampilkan id movie, judul movie, tahun rilis, dan jumlah episode nya, lalu urutkan dari tahun rilis terbaru hingga terlama. Untuk movie yang tidak memiliki episode tidak perlu ditampilkan.



The screenshot shows a SQL query executed in a terminal window. The query is: `SELECT m.id, m.title, m.releasedyear, COUNT(*) FROM movie m INNER JOIN episode e ON m.id = e.mid GROUP BY 1, 2, 3 ORDER BY 3 DESC`. The output is a table with 4 rows and 4 columns: id, title, releasedyear, and count. The data is as follows:

id	title	releasedyear	count
1	8 Cleaning Up	2022	4
2	9 Through the Darkness	2022	6
3	10 Kungfu Panda the Dragon Knight	2022	4
4	1 Cells at Work	2021	10

2. 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 1.



The screenshot shows a SQL query executed in a terminal window. The query is: `SELECT COUNT(DISTINCT episodenum) FROM movie m INNER JOIN episode e ON m.id = e.mid INNER JOIN watch_history wh USING (mid, season, episodenum) INNER JOIN users u ON u.email = wh.uid WHERE m.title = 'Cells at Work' AND u.name = 'Andrew'`. The output is a single row with a single column: count. The value is 2.

count
2

3. 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).

```
Output
eduardus.tj@rahadja.myflit> SELECT DISTINCT title,
                                watch_count
                                FROM (SELECT m.id, m.title, e.episodenum, COUNT(e.episodenum) watch_count
                                FROM movie m
                                INNER JOIN episode e ON m.id = e.mid
                                INNER JOIN watch_history wh USING (mid, season, episodenum)
                                INNER JOIN users u ON u.email = wh.uid
                                GROUP BY 1, 2, 3) AS ew_count
                                WHERE watch_count > 1
[2022-11-14 20:45:07] 2 rows retrieved starting from 1 in 415 ms (execution: 393 ms, fetching: 22 ms)
```

title	watch_count
Cells at Work	2
Kungfu Panda the Dragon Knight	4

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

```
(execution: 309 ms, fetching: 17 ms)
eduardus.tj@rahadja.myflit> SELECT u.name, u.email
                                FROM users u
                                INNER JOIN watch_history wh ON
                                u.email = wh.uid
                                GROUP BY 1, 2
                                HAVING EXTRACT(HOUR FROM MIN(duration)) >=
                                1
[2022-11-14 20:03:50] 1 row retrieved starting from 1 in 276 ms
(execution: 260 ms, fetching: 16 ms)
```

name	email
Nate	mipo@gmail.com

5. 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.

```
Output
eduardus.tj@rahadja.myflit> SELECT m.title, e.etitle, g.type
                                FROM movie m
                                INNER JOIN genre g ON g.id = m.gid
                                LEFT JOIN episode e ON m.id = e.mid
                                WHERE m.viewerageLimit <= (SELECT DATE_PART('year', CURRENT_DATE) -
                                MAX(birthyear)
                                FROM users
                                WHERE name = 'Andrew'
                                OR name = 'Nobita'
                                OR name = 'Monalisa')
[2022-11-14 20:44:39] 7 rows retrieved starting from 1 in 647 ms (execution: 632 ms, fetching: 15 ms)
```

title	etitle	type
Kungfu Panda the Dragon Knight A cause for the Paws		Comedies
Kungfu Panda the Dragon Knight The Knights Code		Comedies
Kungfu Panda the Dragon Knight The Lotus		Comedies
Kungfu Panda the Dragon Knight The Legend of Master Longtooth		Comedies
Yes Day	<null>	Family Movie
A Cinderella Story	<null>	Comedies
Feel the Beat	<null>	Teen Movies

6. 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 pengujian dengan memanggil:

```
SELECT countNumberOfViewers('Kungfu Panda the Dragon Knight', 1, 2);
```

```
eduardus.tjitrahardja.myfLix> -- 6
CREATE OR REPLACE FUNCTION countNumberOfViewers(mtitle varchar(50), eseaon
integer, enum integer)
  RETURNS integer AS
  $$
  DECLARE
    watch_count integer;
  BEGIN
    SELECT COUNT(DISTINCT u.email)
    INTO watch_count
    FROM movie m
      INNER JOIN episode e ON m.id = e.mid
      LEFT JOIN watch_history wh USING (mid, season, episodenum)
      LEFT JOIN users u ON u.email = wh.uid
    WHERE m.title = mtitle
      AND e.season = eseaon
      AND e.episodenum = enum;

    RETURN watch_count;
  END;
  $$ LANGUAGE plpgsql

[2022-11-14 20:27:33] completed in 409 ms
```

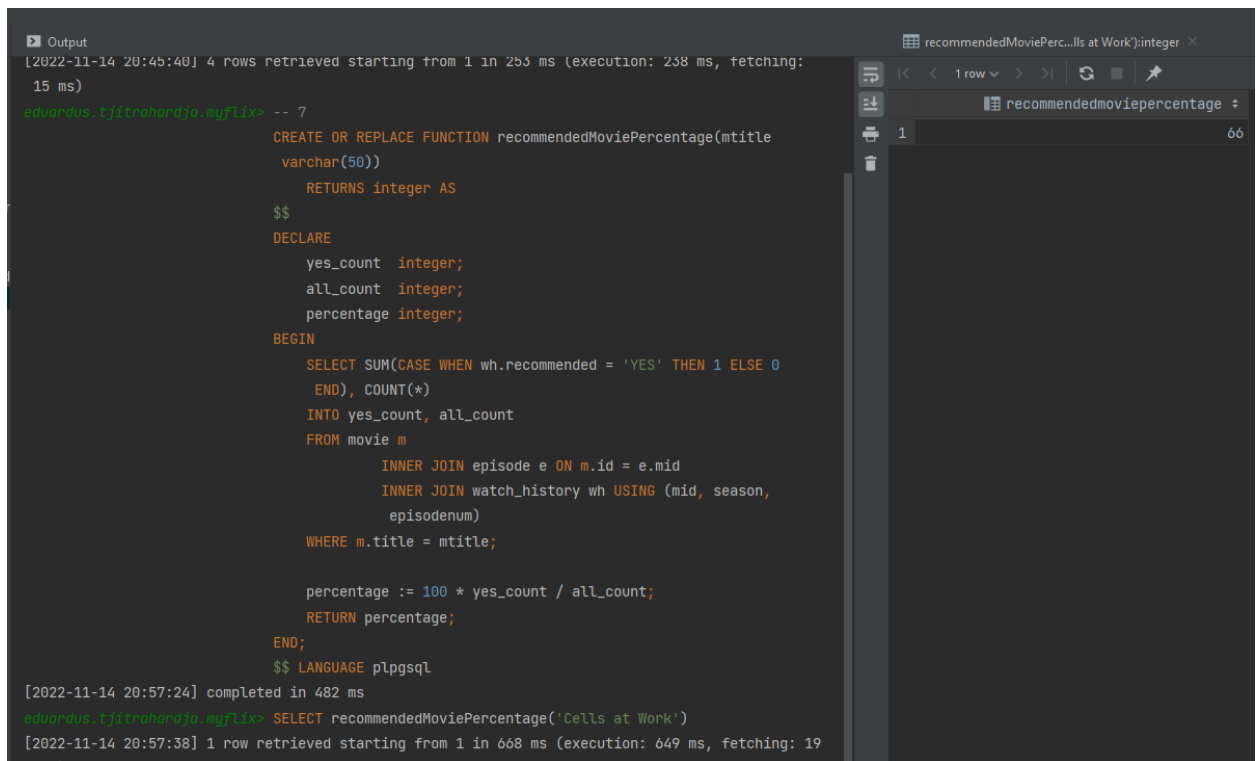
Output

```
eduardus.tjitrahardja.myfLix> SELECT countNumberOfViewers('Kungfu Panda the Dragon Knight', 1, 2)
[2022-11-14 20:44:12] 1 row retrieved starting from 1 in 743 ms (execution: 727 ms, fetching: 16 ms)
```

countNumberOfViewers...night', 1, 2)integer
1

7. 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 $100\% * (\frac{2}{3}) = 66\%$. Setelah stored procedure dibuat, lakukan pengujian dengan memanggil:

```
SELECT recommendedMoviePercentage('Cells at Work');
```



The screenshot shows a database IDE with a dark theme. The main editor displays a PL/SQL function named `recommendedMoviePercentage` that takes a movie title as input and returns an integer percentage. The function logic involves querying a `movie` table and a `watch_history` table to calculate the percentage of recommended movies. The function is executed, and the output is shown in a panel on the right. The output panel displays a single row with the value 66 for the movie 'Cells at Work'.

```
[2022-11-14 20:45:40] 4 rows retrieved starting from 1 in 253 ms (execution: 238 ms, fetching: 15 ms)
eduardo.tj@trahardjo.myflinx> -- 7
CREATE OR REPLACE FUNCTION recommendedMoviePercentage(mtitle
    varchar(50))
    RETURNS integer AS
$$
DECLARE
    yes_count integer;
    all_count integer;
    percentage integer;
BEGIN
    SELECT SUM(CASE WHEN wh.recommended = 'YES' THEN 1 ELSE 0
    END), COUNT(*)
    INTO yes_count, all_count
    FROM movie m
        INNER JOIN episode e ON m.id = e.mid
        INNER JOIN watch_history wh USING (mid, season,
        episodenum)
    WHERE m.title = mtitle;

    percentage := 100 * yes_count / all_count;
    RETURN percentage;
END;
$$ LANGUAGE plpgsql
[2022-11-14 20:57:24] completed in 482 ms
eduardo.tj@trahardjo.myflinx> SELECT recommendedMoviePercentage('Cells at Work')
[2022-11-14 20:57:38] 1 row retrieved starting from 1 in 668 ms (execution: 649 ms, fetching: 19
```

recommendedMoviePercentage('Cells at Work')
66

8. 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');
```

```
Output
adw@adw:~/jitrondia-nyflax$ -- 8
CREATE OR REPLACE FUNCTION checkWatchingDuration()
    RETURNS trigger AS
$$
DECLARE
    e_duration time;
BEGIN
    SELECT eduration
    INTO e_duration
    FROM episode
    WHERE mid = NEW.mid
        AND season = NEW.season
        AND episodenum = NEW.episodenum;

    IF (NEW.duration > e_duration) THEN
        RAISE EXCEPTION
            'Durasi menonton suatu episode movie dari seorang pengguna tidak mungkin lebih besar dari durasi episode itu sendiri.';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql
[2022-11-14 20:41:04] completed in 396 ms
adw@adw:~/jitrondia-nyflax$ CREATE TRIGGER checkWatchingDurationTrigger
    BEFORE INSERT OR UPDATE
    ON watch_history
    FOR EACH ROW
    EXECUTE PROCEDURE checkWatchingDuration()
[2022-11-14 20:41:36] completed in 8 s 296 ms
adw@adw:~/jitrondia-nyflax$ INSERT INTO watch_history
    VALUES (11, 'mipo@gmail.com', 10, 1, 3, '2020-10-28 10:00', '03:00:00', 'YES')
[2022-11-14 20:42:00] [P0001] ERROR: Durasi menonton suatu episode movie dari seorang pengguna tidak mungkin lebih besar dari durasi episode itu sendiri.
```