

LAPORAN TUGAS BESAR BASIS DATA

Music Streaming Service

Laporan ini disusun untuk memenuhi Tugas Mata Kuliah Sistem Basis Data.



Disusun oleh:

Arkan Ramadhan Nugraha	241524033
Fauzi Ismail	241524042
Nurahma Rahayu	241524058
Virli Nasyila Putri	241524062

**PROGRAM STUDI SARJANA TERAPAN TEKNIK INFORMATIKA
JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA
POLITEKNIK NEGERI BANDUNG**

2025

DAFTAR ISI

DAFTAR ISI.....	i
DAFTAR GAMBAR	ii
DAFTAR TABEL	iv
1. DESKRIPSI	1
2. RUANG LINGKUP	4
3. BUSINESS RULE	7
4. PERANCANGAN PENGOLAHAN DATA.....	13
4.1 Entity-Relationship Diagram (E-RD).....	13
4.2 Logical Data Model	14
4.3 Physical Data Model	14
4.4 Kamus Data	15
4.4.1 Table	15
4.4.2 Sequence.....	30
4.4.3 Indexing.....	31
4.5 Function/ Procedure	36
4.6 Trigger	59
4.7 View	64
4.8 Cron.....	67
5. IMPLEMENTASI PENGOLAHAN DATA	72
5.1 DDL.....	72
5.2 DML.....	106
5.3 PL/SQL.....	128
6. KESIMPULAN.....	247
7. PEMBAGIAN TUGAS	249
8. LESSON LEARNED.....	252
9. REPOSITORY GITHUB	254

DAFTAR GAMBAR

Gambar 4. 1 Entity-Relationship Diagram (ERD)	13
Gambar 4. 2 Logical Data Model (LDM).....	14
Gambar 4. 3 Physical Data Model (PDM)	14
Gambar 4. 4 Flow Chart Function register_user	36
Gambar 4. 5 Flow Chart Function register_user	37
Gambar 4. 6 Flow Chart Function login_user	37
Gambar 4. 7 Flow Chart Procedure toggle_follow_user	38
Gambar 4. 8 Flow Chart Function get_followers.....	38
Gambar 4. 9 Flow Chart Function get_following.....	39
Gambar 4. 10 Flow Chart Procedure create_review	39
Gambar 4. 11 Flow Chart Procedure toggle_like_review.....	40
Gambar 4. 12 Flow Chart Function search	41
Gambar 4. 13 Flow Chart Function get_artist_detail	41
Gambar 4. 14 Flow Chart Function get_artist_content.....	41
Gambar 4. 15 Flow Chart Procedure toggle_follow_artist.....	42
Gambar 4. 16 Flow Chart Function get_collection_tracks	42
Gambar 4. 17 Flow Chart Function get_new_releases.....	43
Gambar 4. 18 Flow Chart Procedure add_artist_promotion	43
Gambar 4. 19 Flow Chart Function get_artist_tours.....	44
Gambar 4. 20 Flow Chart Function get_song_audio_features.....	44
Gambar 4. 21 Flow Chart Function get_recently_played.....	45
Gambar 4. 22 Flow Chart Procedure log_listen.....	45
Gambar 4. 23 Flow Chart Procedure rate_song	46
Gambar 4. 24 Flow Chart Procedure toggle_like_song	47
Gambar 4. 25 Flow Chart Function get_song_detail	48
Gambar 4. 26 Flow Chart Function get_playlist_tracks.....	49
Gambar 4. 27 Flow Chart Function get_playlist_detail	49
Gambar 4. 28 Flow Chart Procedure remove_song_from_playlist.....	50
Gambar 4. 29 Flow Chart Procedure add_song_to_playlist.....	51
Gambar 4. 30 Flow Chart Procedure create_playlist.....	52
Gambar 4. 31 Flow Chart Function get_collection_genres	53
Gambar 4. 32 Flow Chart Function get_collection_average_rating.....	54
Gambar 4. 33 Flow Chart Function get_song_average_rating.....	55
Gambar 4. 34 Flow Chart Function get_playlist_duration_minutes	56
Gambar 4. 35 Flow Chart Function get_artist_total_plays	57
Gambar 4. 36 Flow Chart Function get_album_total_duration	58
Gambar 4. 37 Flow Chart Trigger update_collection_top3_genres	59
Gambar 4. 38 Flow Chart Trigger update_song_rating	60
Gambar 4. 39 Flow Chart Trigger trg_validate_prerelease_date.....	60
Gambar 4. 40 Flow Chart Trigger update_review_timestamp.....	61
Gambar 4. 41 Flow Chart Trigger trg_fix_playlist_collaborators	61
Gambar 4. 42 Flow Chart Trigger trg_update_artist_follow_count.....	62
Gambar 4. 43 Flow Chart Trigger trg_update_collection_rating	62
Gambar 4. 44 Flow Chart Trigger trg_validate_tour_date	63
Gambar 4. 45 Flow Chart Trigger reorder_playlist_sequence	63

Gambar 4. 46 Flow Chart Trigger enforce_block_logic	64
Gambar 4. 47 Flow Chart Cron delete_expired_tours_daily.....	68
Gambar 4. 48 Flow Chart Cron recalc_listen_count_daily	68
Gambar 4. 49 Flow Chart Cron recalculate_song_popularity_daily.....	69
Gambar 4. 50 Flow Chart Cron recalculate_monthly_listeners_daily.....	70
Gambar 4. 51 Flow Chart Cron update_prerelease_job_daily	71

DAFTAR TABEL

Tabel 1 Aturan Bisnis Pengolahan Data	7
Tabel 2 Tabel Users	15
Tabel 3 Tabel Artists	15
Tabel 4 Tabel Songs	16
Tabel 5 Tabel Genres	17
Tabel 6 Tabel Playlists	18
Tabel 7 Tabel PI_Library	19
Tabel 8 Tabel Add_Songs_Playlists	19
Tabel 9 Tabel Like_Songs	20
Tabel 10 Tabel Listens	20
Tabel 11 Tabel Follow_Users	21
Tabel 12 Tabel Follow_Artists	21
Tabel 13 Tabel Collections	22
Tabel 14 Tabel Songs_Genres	23
Tabel 15 Tabel Collection_Top_3_Genre	23
Tabel 16 Tabel Collections_Songs	23
Tabel 17 Tabel Releases	24
Tabel 18 Tabel Create_Songs	24
Tabel 19 Tabel Artist_Promotion	25
Tabel 20 Tabel Tours	25
Tabel 21 Tabel Artists_Tours	26
Tabel 22 Tabel Rate_Songs	26
Tabel 23 Tabel Block_Users	27
Tabel 24 Tabel Blocklist_Users	27
Tabel 25 Tabel Collection_Library	28
Tabel 26 Tabel Like_Reviews	28
Tabel 27 Tabel Reviews	28
Tabel 28 Tabel Socials	29
Tabel 29 Sequence	30
Tabel 30 Indexing	32
Tabel 31 View Album Tracklist	64
Tabel 32 View Artist Header	65
Tabel 33 View Full Song Details	65
Tabel 34 View Top Charts	66
Tabel 35 View User Library Stats	66
Tabel 36 DDL – Tabel add_songs_playlist	72
Tabel 37 DDL – Tabel Artists	73
Tabel 38 DDL – Tabel Artists_tours	74
Tabel 39 DDL – Tabel Artist_Promotion	74
Tabel 40 DDL – Tabel Blocklist_artists	75
Tabel 41 DDL – Tabel block_users	76
Tabel 42 DDL – Tabel collections	77
Tabel 43 DDL – Tabel Collections_Songs	78
Tabel 44 DDL – Tabel Collection_library	79
Tabel 45 DDL – Tabel collection_top_3_genres	80

Tabel 46 DDL – Tabel create_songs	81
Tabel 47 DDL – Tabel follow_artists	82
Tabel 48 DDL – Tabel follow_users.....	83
Tabel 49 DDL – Tabel genres.....	84
Tabel 50 DDL – Tabel like_reviews	85
Tabel 51 DDL – Tabel like_songs.....	86
Tabel 52 DDL – Tabel listens	87
Tabel 53 DDL – Tabel playlists	88
Tabel 54 DDL – Tabel PI_library	89
Tabel 55 DDL – Tabel Rate_songs.....	90
Tabel 56 DDL – Tabel releases	91
Tabel 57 DDL – Tabel reviews	92
Tabel 58 DDL – Tabel socials.....	93
Tabel 59 DDL – Tabel songs	94
Tabel 60 DDL – Tabel song_genres.....	95
Tabel 61 DDL – Tabel tours	96
Tabel 62 DDL – Tabel users	97
Tabel 63 DDL - Sequence	98
Tabel 64 DDL - Constraints.....	102
Tabel 65 DML – Follow user.....	106
Tabel 66 DML – Rilis Collection.....	107
Tabel 67 DML – Promosi Rilis.....	107
Tabel 68 DML – Manajemen Tur	108
Tabel 69 DML – Like Review	109
Tabel 70 DML – playlist_collaborators	109
Tabel 71 DML – Update follow count	111
Tabel 72 DML – update collection rating.....	114
Tabel 73 DML – Update collection top3 genres	117
Tabel 74 DML – Update review timestamp	120
Tabel 75 DML – Update song rating	123
Tabel 76 PL/SQL – Function get_song_audio_features.....	128
Tabel 77 PL/SQL - Function get_recently_played.....	131
Tabel 78 PL/SQL – Function get_song_detail	134
Tabel 79 PL/SQL – Function get_playlist_tracks.....	138
Tabel 80 PL/SQL - Function get_playlist_detail	139
Tabel 81 PL/SQL – Procedure log_listen.....	143
Tabel 82 PL/SQL – Procedure rate_song	149
Tabel 83 PL/SQL – Procedure toggle_like_song	152
Tabel 84 PL/SQL – Procedure remove_song_from_playlist.....	156
Tabel 85 PL/SQL – Procedure add_song_to_playlist	160
Tabel 86 PL/SQL – Procedure create_playlist.....	164
Tabel 87 PL/SQL – Function Register User.....	168
Tabel 88 PL/SQL – Function Login User	169
Tabel 89 PL/SQL – Procedure Toggle Follow User.....	170
Tabel 90 PL/SQL – Function Get Followers.....	171
Tabel 91 PL/SQL – Function Get Following.....	172
Tabel 92 PL/SQL – Procedure Create Review	173
Tabel 93 PL/SQL – Procedure Toggle Like Reivew.....	174

Tabel 94 PL/SQL – Function Search	175
Tabel 95 PL/SQL – Function Get Artist Detail	177
Tabel 96 PL/SQL – Function Get Artist Content.....	179
Tabel 97 PL/SQL – Procedure Toggle Follow Artist	181
Tabel 98 PL/SQL – Function Get Collection Detail.....	182
Tabel 99 PL/SQL – Function Get Collection Tracks.....	183
Tabel 100 PL/SQL – Function Get New Releases.....	184
Tabel 101 PL/SQL – Procedure Add Artist Promotion	185
Tabel 102 PL/SQL – Function Get Artist Tours.....	187
Tabel 103 PL/SQL – Function delete_expired_tours	189
Tabel 104 PL/SQL – Function recalc_listen_count	190
Tabel 105 PL/SQL – Function recalculate_all_song_popularity	191
Tabel 106 PL/SQL - Function recalculate_monthly_listeners	192
Tabel 107 PL/SQL - Function update_prerelease_daily.....	194
Tabel 108 PL/SQL - Function get_collection_genres	195
Tabel 109 PL/SQL - Function get_collection_average_rating.....	196
Tabel 110 PL/SQL - Function get_song_average_rating.....	197
Tabel 111 PL/SQL - Function get_playlist_duration_minutes	198
Tabel 112 PL/SQL - Function get_artist_total_plays	199
Tabel 113 PL/SQL - Function get_album_total_duration	200
Tabel 113 PL/SQL - Trigger validate_tour_date.....	201
Tabel 115 PL/SQL - Trigger trg_update_collection_top3_genres	203
Tabel 116 PL/SQL - Trigger trg_update_song_rating	207
Tabel 117 PL/SQL – Trigger trg_validate_prerelease_date	214
Tabel 118 PL/SQL - Trigger trg_update_review_timestamp	219
Tabel 119 PL/SQL - Trigger trg_fix_playlist_collaborators	223
Tabel 120 PL/SQL - Trigger trg_update_artist_follower_count	227
Tabel 121 PL/SQL - Trigger trg_update_collection_rating	233
Tabel 122 PL/SQL - Trigger trg_reorder_playlist_sequence.....	240
Tabel 123 PL/SQL - Trigger trg_enforce_block_logic	243
Tabel Kontribusi Pengerjaan.....	249
Tabel Daftar Pengerjaan	249

1. DESKRIPSI

Layanan *streaming* musik digital, seperti Spotify, telah menjadi platform utama bagi masyarakat untuk mengonsumsi konten audio. Keberhasilan layanan ini sangat bergantung pada kemampuannya untuk mengelola basis data yang masif dan kompleks. Masalah utama yang dihadapi adalah bagaimana menyimpan, mengelola, dan menghubungkan jutaan data lagu, album, dan artis, sekaligus melayani permintaan jutaan pengguna secara *real-time* dengan akurat. Kita perlu memperhatikan, bagaimana data-data tersebut disimpan, bagaimana juga data tersebut berkembang seiring berjalananya waktu, bagaimana data tersebut ketika dijalankan, bagaimana keamanannya. Sistem basis data yang dirancang dengan baik sangat krusial untuk memastikan data dapat diakses dengan cepat, rekomendasi yang dipersonalisasi dapat diberikan, dan integritas data tetap terjaga.

Proses pengolahan data (proses bisnis) utama dapat dibagi menjadi dua alur besar: alur pengguna (konsumen) dan alur konten (kreator/admin).

1. Alur Pengguna (Konsumen)

Proses dimulai saat User mendaftar dan melakukan login menggunakan email dan password. Setelah masuk, user dapat melakukan berbagai aktivitas inti seperti streaming (mendengarkan) Lagu, mencari Artis, menelusuri Album (atau EP dan kompilasi), dan memfilter berdasarkan Genre. Alur data yang dipersonalisasi terjadi ketika user mulai berinteraksi dengan sistem, seperti:

1. Menyukai Lagu, album, atau EP
2. Mengikuti (Follow) Artis
3. Mengikuti (Follow) User lain
4. Membuat Playlist, yaitu koleksi lagu custom dengan urutan custom yang spesifik.
5. Dapat memberi rating dan memberi *review* lagu/album/EP.

2. Alur Konten (Kreator/Admin)

Alur ini mengelola bagaimana konten musik itu sendiri direpresentasikan dalam database

1. Satu lagu dapat memiliki satu atau lebih genre.
2. Satu lagu dapat dimiliki oleh satu atau lebih Artis
3. Setiap *Lagu* memiliki metadata audio (seperti *valence*, *acousticness*) yang mendeskripsikan "vibe" atau *mood* lagu tersebut.
4. *Artis* memiliki *Lagu* dan *Collection* (sebuah entitas yang merepresentasikan Album, EP, atau Kompilasi).

5. Sistem juga mengelola data *Tour*, yang mencatat jadwal dan lokasi tur seorang *Artis* selama periode waktu tertentu. Tour bisa berupa kolaborasi artis.
6. Artis bisa promote Satu release nya entah itu lagu atau collection di page-nya dengan message tertentu.
7. Collection bisa terdiri dari beberapa disc. Setiap disc di-reset numbering nya

Masalah unik yang diangkat dalam studi kasus ini adalah implementasi sistem *review* kuantitatif yang tidak ada di layanan referensi. Sistem ini memungkinkan pengguna memberikan rating numerik (skala 1–100) dan komentar terhadap lagu maupun koleksi musik (album/EP). Ini menciptakan alur pengolahan data baru di mana data *rating* dan komentar yang dibuat pengguna harus disimpan, diagregasi (untuk menghitung *rating* rata-rata), dan dihubungkan secara efisien ke setiap entitas musik untuk dapat ditampilkan kembali kepada *user* lain.

Untuk mendukung proses tersebut, sistem basis data harus mampu melakukan agregasi, relasi, dan pengambilan data secara efisien tanpa mengorbankan performa.

Dalam skala besar seperti Spotify yang memiliki lebih dari 100 juta lagu, hampir 7 juta podcast, serta lebih dari 713 juta pengguna termasuk 281 juta pengguna yang berlangganan, desain database tidak boleh dilakukan secara sembarangan. Terdapat 3 permasalahan utama dalam hal menyusun database design ini, yaitu:

- Performance. Basis data harus mampu menangani permintaan (query) dalam jumlah besar secara cepat dan efisien.
- Scalability. Struktur data harus mudah dikembangkan seiring berjalannya waktu.
- Security. Data pengguna dan artis harus terlindungi dari kejahatan siber.

Perancangan basis data ini dilakukan untuk membangun sebuah basis data yang efisien, terstruktur, dan mudah dikembangkan untuk mendukung operasi layanan dengan skala besar. Sistem ini juga dirancang agar mampu mengelola banyak entitas seperti users, lagu, artis, collection (Album/EP/Kompilasi), Genre, Playlist, Tour, serta interaksi antar entitas secara real-time dengan tingkat konsistensi data yang cukup tinggi.

Tujuan utama mencakup:

- Mewujudkan pengelolaan data musik digital yang terintegrasi dan terorganisir.
- Mendukung seluruh proses bisnis, mulai dari registrasi pengguna hingga pengelolaan tur artis.
- Meningkatkan kecepatan pencarian dan akses data untuk mendukung fitur real-time seperti rekomendasi dan histori pemutaran lagu.

- Menjamin integritas, konsistensi, dan keamanan data pengguna maupun konten.
- Menyediakan fondasi untuk fitur analitik dan personalisasi, seperti tren popularitas, rating rata-rata, dan rekomendasi berbasis preferensi pengguna.

Sistem basis data yang dirancang untuk layanan Music Streaming Service ini memberikan berbagai manfaat dalam konteks aplikasi musik digital, antara lain:

- Efisiensi pengelolaan data, memungkinkan penyimpanan dan pengolahan jutaan lagu, artis, album, dan pengguna dengan performa tinggi.
- Personalisasi pengalaman pengguna, melalui fitur rekomendasi lagu, playlist dinamis, dan sistem rating yang interaktif.
- Kemudahan bagi kreator dan admin, dalam mengelola konten musik, tur, serta promosi album secara terstruktur.
- Peningkatan keterlibatan pengguna, karena adanya fitur interaksi sosial seperti like, follow, dan rating.
- Keamanan dan integritas data yang lebih baik, mendukung keberlanjutan layanan musik digital yang andal dan modern.

2. RUANG LINGKUP

Ruang lingkup dari studi kasus ini berfokus pada perancangan dan implementasi sistem basis data relasional untuk layanan Music Streaming Service yang meniru konsep aplikasi musik digital seperti Spotify. Sistem ini mencakup pengelolaan data pengguna, artis, lagu, koleksi (album/EP/kompilasi), genre, playlist, serta aktivitas interaksi pengguna seperti follow, like, dan rating.

Perancangan difokuskan pada bagaimana data antar entitas tersebut saling berelasi, disimpan, dan diolah secara efisien untuk mendukung proses bisnis utama layanan streaming musik digital.

Berikut adalah komponen utama yang termasuk dalam ruang lingkup sistem:

1. Manajemen User

a. Registrasi dan Login

Pengguna dapat membuat akun baru dan masuk ke sistem menggunakan email dan password.

b. Profil Pengguna

Pengguna dapat mengubah nama dan profile picture.

c. Streaming dan Riwayat Listening

Pengguna dapat streaming/memutar lagu dan histori pemutaran disimpan untuk mendukung fitur rekomendasi pengguna.

d. Interaksi Sosial

Pengguna dapat follow pengguna lain, follow artis, melihat aktivitas dan playlist pengguna lain yang publik, like lagu, review collection, review bisa di like oleh pengguna lain, pengguna dapat memblokir pengguna lain, mem-blacklist artis, dan memasukkan playlist orang lain dan collection ke library. Playlist dapat bersifat publik atau privat, serta dapat dibuat untuk digunakan (termasuk menambah lagu) bersama (collaborative) oleh beberapa pengguna.

2. Manajemen Konten Musik

a. Entitas Lagu

Menyimpan informasi dasar seperti judul, durasi, artis, genre, serta metadata audio seperti *valence*, *energy*, *danceability*, dan *acousticness*.

b. Entitas Collection (Album/EP/Koleksi/Single)

Menyimpan beberapa lagu, satu lagu dapat dimiliki oleh beberapa collection. Setiap lagu dalam sebuah collection memiliki nomor disc dan nomor urut track.

c. Genre

Lagu dapat dikategorikan dalam satu atau lebih genre.

d. Kolaborasi Artis

Satu lagu dapat dimiliki oleh beberapa artis (many-to-many).

3. Manajemen Playlist dan Koleksi Pribadi Pengguna

a. Pembuatan Playlist

Menyimpan beberapa lagu user dapat memiliki beberapa playlist. Playlist dapat diatur agar tampil di profil pengguna (*isOnProfile*), bersifat publik/privat (*isPublic*), serta dapat dibuat bersama pengguna lain (*isCollaborative*).

b. Pengaturan Urutan Lagu

Urutan lagu dapat disusun manual oleh user atau berdasarkan jumlah pemutaran yang tercatat.

c. Penambahan Lagu ke Playlist

4. Manajemen Artis

a. Profil Artis

Menyimpan data seperti biografi, profile picture, banner, jumlah pengikut, daftar lagu dan collections milik artis.

b. Rilis Collection

Artis dapat merilis lagu atau collection, dicatat tanggal rilis nya. Suatu rilis bisa berupa Pre-Release, dimana wujudnya sudah dapat terlihat (tracklist, judul collection, tipe collection, cover album/lagu), tetapi lagu-lagu yang belum dirilis belum dapat didengar sampai release date.

c. Promosi Rilis

Artis dapat memilih satu rilisnya (lagu atau collection) untuk di rekomendasi kan di profilnya.

d. Manajemen Tur

Daftar tur artis tersebut, termasuk artis lain yang ada bareng, tanggal tur, tempat tur, dan lain-lain.

5. Sistem Rating dan Review

a. Pemberian Rating

Pengguna dapat memberikan nilai numerik (skala 1-100) terhadap lagu atau koleksi. Nilai ini menjadi indikator tingkat kepuasan pengguna terhadap konten tersebut.

b. Komentar Review

Pengguna dapat menulis komentar yang berisikan pendapat atau ulasan pribadi tentang koleksi. Untuk lagu, hanya ada rating saja.

c. Agregasi

Menghitung rata-rata rating dari seluruh pengguna yang memberi penilaian terhadap suatu lagu atau koleksi. Nilai agregasi ini disimpan dan digunakan untuk menampilkan peringkat atau rekomendasi

d. Tampilan Rating

Nilai rata-rata rating serta jumlah review akan ditampilkan pada halaman detail lagu (hanya rating dan jumlah rating) atau koleksi.

6. Pencarian dan Rekomendasi

Fitur ini merupakan proses turunan dari data yang tersimpan dalam basis data relasional

a. Pencarian berdasarkan entitas

Pengguna dapat mencari lagu, artis, atau album berdasarkan nama, genre, atau kata kunci lainnya.

b. Filter dan Sortir

Hasil pencarian dapat difilter berdasarkan genre, popularitas, rating, atau waktu rilis.

c. Personalized Recommendations

Sistem dapat menampilkan rekomendasi berdasarkan histori mendengarkan, artis yang diikuti, atau rating yang diberikan pengguna.

7. Keamanan dan Integritas Data

a. Keamanan Akses

Sistem mendukung pengaturan hak akses di tingkat aplikasi untuk memastikan keamanan data pengguna.

- b. Integritas Relasi

Seluruh entitas (User, Song, Artist, Collection, Genre, Playlist, Rating, Tour) terhubung melalui kunci utama dan kunci asing untuk menjaga konsistensi.

8. Batasan Sistem

Ruang lingkup kasus ini tidak mencakup:

- a. Implementasi streaming audio secara aktual
- b. Sistem pembayaran, langganan, atau manajemen lisensi lagu.

3. BUSINESS RULE

Pada bagian ini dijelaskan aturan-aturan bisnis yang mengatur proses pengolahan data dalam sistem Music Streaming Service. Aturan bisnis ini berfungsi untuk menggambarkan bagaimana setiap entitas saling berinteraksi serta batasan-batasan yang berlaku dalam pengelolaan data di sistem.

Tabel berikut menyajikan daftar proses bisnis utama beserta entitas yang terlibat dan aturan yang diterapkan dalam masing-masing proses:

Tabel 1 Aturan Bisnis Pengolahan Data

No.	Nama Proses Bisnis	Entitas	Aturan
1.	Registrasi dan Login User	User	Email dan username harus unik. Password harus memenuhi panjang minimum (hashnya harus lebih dari 8 karakter). Email dan username wajib ada. Username tidak boleh berupa whitespace. Email harus mengikuti format email valid.
		User	User tidak dapat mengakses fitur seperti streaming, like, dan rating tanpa melakukan login terlebih dahulu.

		User	User dapat memiliki satu atau lebih playlist pribadi setelah berhasil login.
2.	Pembuatan Playlist oleh user	User, Playlist	User dapat membuat banyak playlist dengan nama yang berbeda
		Playlist	Satu playlist dapat berisi satu atau lebih lagu. Playlist harus memiliki satu pemilik. Playlist default bersifat private dan non-collaborative.
		Song	Satu lagu dapat dimasukkan ke banyak playlist yang berbeda.
		Song	Satu playlist dapat terdiri dari beberapa lagu.
		Song	User dapat mengatur urutan lagu dalam playlist.
		User, Playlist	User dapat menyimpan playlist user lain
		User, Playlist, Song	Urutan lagu harus bernilai positif dan berurutan untuk setiap playlist.
3.	Streaming dan History (Riwayat Listening)	User, Song	Pengguna dapat streaming/memutar lagu dan Riwayat listening-nya disimpan sebagai history.
		User, Song	Durasi dengar harus positif. Setiap record memiliki timestamp otomatis.
4.	Rating dan Review	User, Collection	Satu user hanya dapat memberi satu review per

			koleksi beserta rating yang berada pada rentang 1–100.
		User, Song	User hanya boleh memberikan satu rating per lagu. Rating berada dalam rentang 1–100.
		Review	Satu user hanya dapat memberikan satu review untuk setiap collection
		Song, Collection	Nilai rata – rata rating dihitung dari data review (untuk collection) dan data rating (untuk song)
		User, Review	User dapat memberikan like pada hasil review dari orang lain. Satu user hanya boleh like satu review sekali. Tidak boleh like review berulang.
5.	Sistem Follow	User, Artis	User dapat mengikuti artis. User hanya dapat follow artis satu kali.
		User	User dapat mengikuti user lain. User tidak dapat mem-follow dirinya sendiri. Satu user tidak dapat follow user yang sama lebih dari satu kali.
6.	Manajemen Koleksi Musik	Artis, Collection, Song	Satu artis dapat memiliki banyak koleksi (collection)

		Collection	Tipe koleksi harus salah satu dari: Album, EP, Single, Compilation.
		Collection, Song	Satu lagu dapat masuk ke beberapa koleksi . Nomor disc dan nomor track harus bernilai positif untuk setiap lagu dalam suatu koleksi.
		Collection	Koleksi berstatus prerelease harus memiliki release_date di masa depan. Pada hari rilis, status prerelease tidak lagi berlaku.
		Collection, Genre	Koleksi hanya dapat menyimpan maksimal 3 genre teratas berdasarkan lagu-lagunya.
7.	Manajemen Lagu	Artis, Song, Genre	Lagu dapat dibuat oleh satu atau lebih artis (kolaborasi). Setiap lagu harus memiliki minimal satu artis. Pasangan lagu-artis tidak boleh duplikat
		Song, Genre	Lagu dapat memiliki satu atau lebih genre(maks 3). Satu lagu dapat memiliki beberapa genre, tetapi tidak boleh duplikat pasangan.
		Song	Durasi lagu harus positif. Popularitas berada dalam rentang 0–100. Nilai audio metrics berada dalam rentang 0–1 dengan 3 angka di desimal. Popularitas

			dihitung dengan rumus berikut: popularitas = (jumlah_play_30_hari_terakhir / jumlah_play_lagu_terbanyak_di_sistem_30_hari_terakhir) * 100
		User, Song	User hanya boleh like satu lagu satu kali. Timestamp tercatat otomatis.
8.	Promosi Lagu oleh Artis	Artis, Collection, Song	Artis hanya dapat mempromosikan koleksi yang benar-benar dirilis oleh artis tersebut.
9.	Manajemen Tur Artis	Artis, Tour	Artis dapat memiliki satu atau lebih jadwal tour
		Tour	Tour mencatat tour_name, venue, dan tour_date. Tanggal tour tidak boleh berada di masa lalu.
		Artis, Tour	Satu tour dapat melibatkan beberapa artis
10.	Pencarian dan Rekomendasi	User, Song, Artis, Genre	User dapat mencari lagu, artis, atau koleksi berdasarkan kata kunci
		User, Song	Sistem rekomendasi didasarkan pada histori mendengarkan dan artis yang diikuti.
11.	Keamanan dan Integritas Data	User	Atribut password pada User dilindungi dengan function/algoritma bcrypt beserta salt.

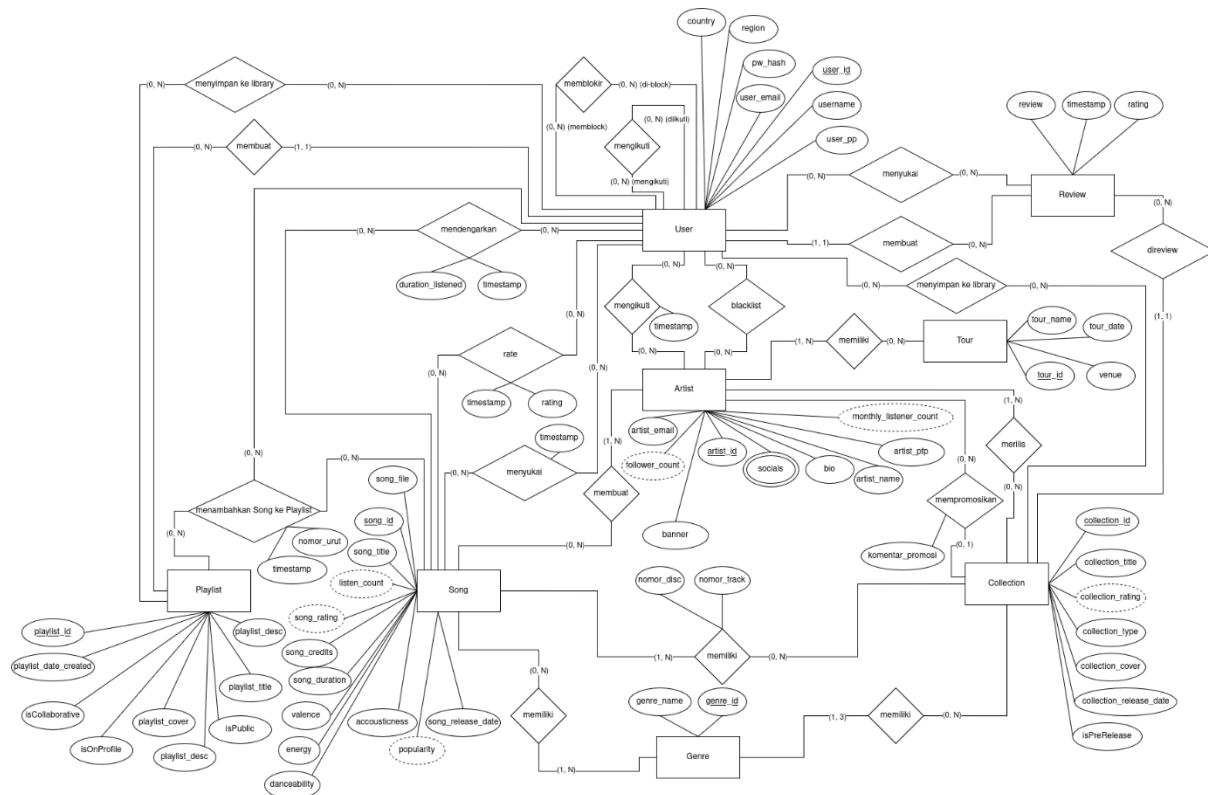
12.	Blocking User dan Blacklist Artis	User	User tidak dapat mem-block dirinya sendiri. Block bersifat unik per pasangan user.
		User, Artis	Setiap user dapat mem-blacklist seorang/beberapa artis. User hanya dapat mem-block artis satu kali.
13.	Manajemen sosial media artis	Artis	Artis dapat memiliki banyak social media link
14.	Manajemen Library User	User, Playlist	User hanya dapat menyimpan playlist satu kali; tidak boleh duplikat
		User, Collection	Satu koleksi hanya dapat disimpan sekali per user. Tidak boleh duplikat.

4. PERANCANGAN PENGOLAHAN DATA

4.1 Entity-Relationship Diagram (E-RD)

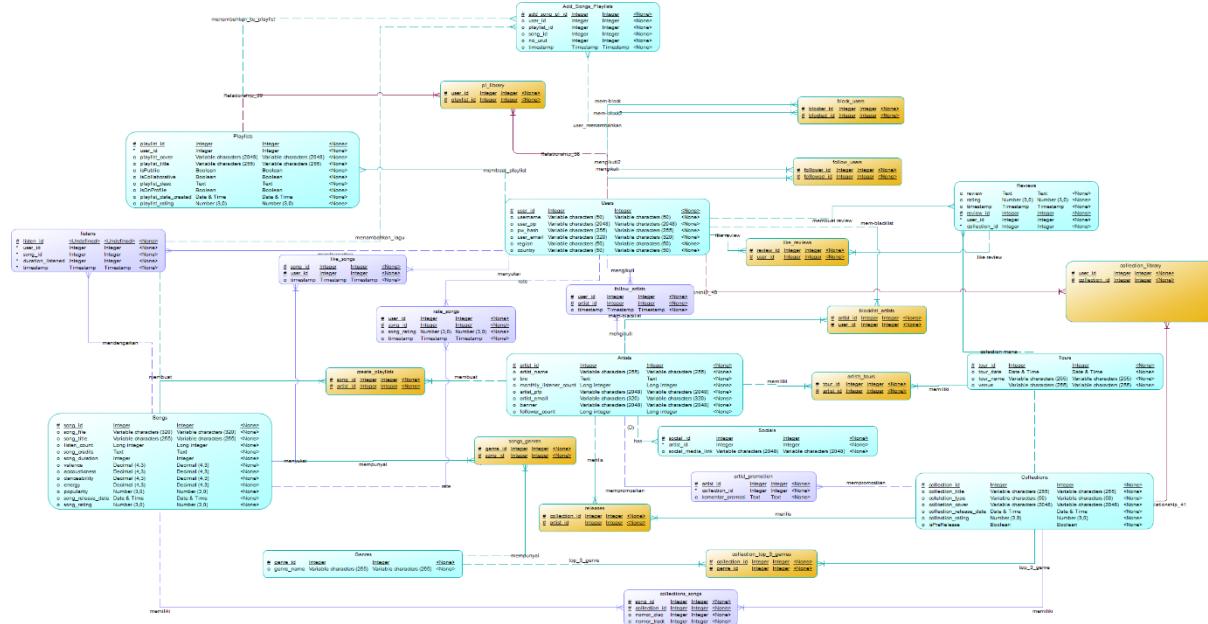
Untuk menggambarkan hubungan antar entitas dalam sistem yang dirancang, dibuat sebuah Entity-Relationship Diagram (ERD) dengan menggunakan notasi Chen. Diagram ini bertujuan untuk memodelkan struktur data secara konseptual, termasuk entitas utama, atribut yang dimiliki, serta hubungan antar entitas beserta kardinalitasnya (minimum dan maksimum).

ERD berikut menunjukkan bagaimana setiap entitas saling berinteraksi dalam sistem basis data yang akan dibangun:



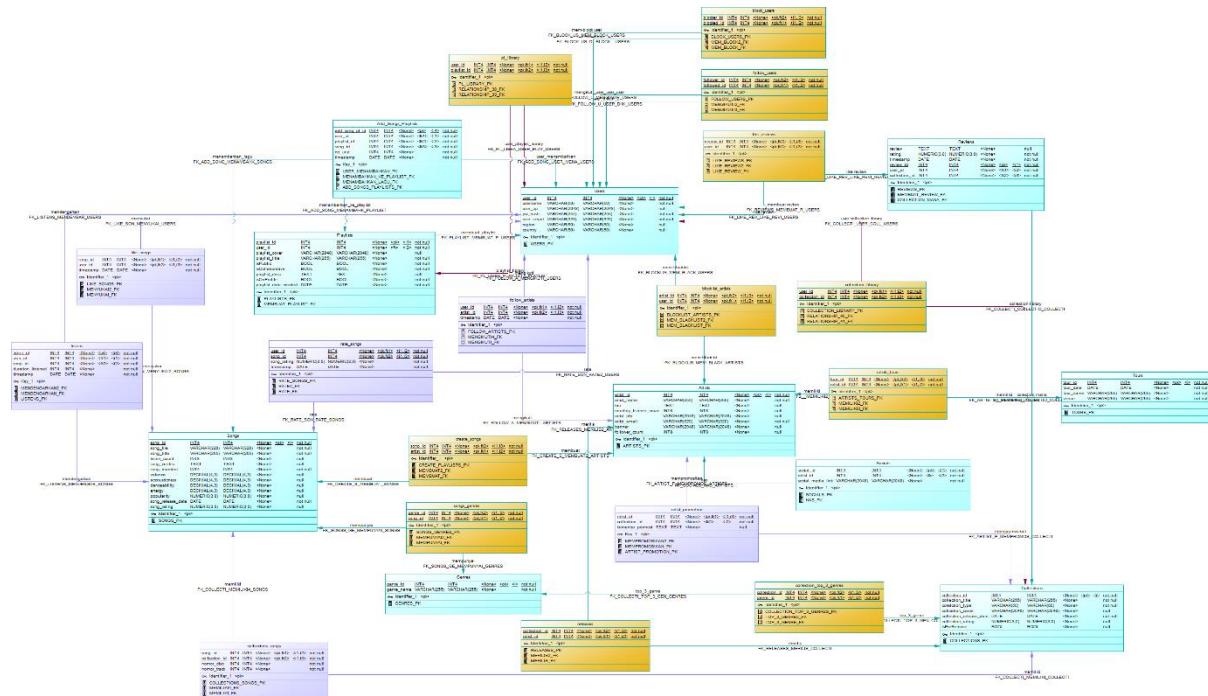
Gambar 4. 1 Entity-Relationship Diagram (ERD)

4.2 Logical Data Model



Gambar 4. 2 Logical Data Model (LDM)

4.3 Physical Data Model



Gambar 4. 3 Physical Data Model (PDM)

4.4 Kamus Data

4.4.1 Table

Tabel 2 Tabel Users

Nama		USERS			
Deskripsi					
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	USER_ID	INT4	PK, NOT NULL	ID unik untuk setiap user	1023
2	USERNAME	VARCHAR(50)	NOT NULL, LENGTH(TRIM (USERNAME)) > 0	Nama akun yang ditampilkan	"aurorabeat s"
3	USER_PFP	VARCHAR(2048)	NULL	URL foto profil user	"https://img.com/u/pp123.png"
4	PW_HASH	VARCHAR(255)	NOT NULL, LENGTH >= 8	Hash password user	"\$2a\$10\$9sd8asd8asd..."
5	USER_EMAIL	VARCHAR(320)	NOT NULL, LIKE '%_@__%.__%'	Email untuk login/verifikasi	"aurora@mail.com"
6	REGION	VARCHAR(50)	NULL	Wilayah tertentu dari user	"West Java"
7	COUNTRY	VARCHAR(50)	NULL	Negara asal user	"Indonesia"

Tabel 3 Tabel Artists

Nama		ARTISTS
Deskripsi		
		Menyimpan data artis, termasuk nama, foto profil, banner, email, jumlah pendengar, dan koleksi (album/EP) yang terkait. Tabel ini menggunakan composite primary key (ARTIST_ID, COLLECTION_ID).

No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	ARTIST_ID	INT4	PK, NOT NULL	ID unik artis	11
2	ARTIST_NAME	VARCHAR(255)	NOT NULL, LENGTH(TRIM(ARTIST_NAME)) > 0	Nama artis	“The Weekend”
3	BIO	TEXT	NULL	Biografi artis	“Canadian singer and producer...”
4	MONTHLY_LISTENER_COUNT	INT8	NULL, default 0, value >= 0	Jumlah pendengar bulanan	67500000
5	ARTIST_PFP	VARCHAR(2048)	NULL	Foto profil artis	“https://cdn.com/a11.png”
6	ARTIST_EMAIL	VARCHAR(320)	NOT NULL	Email artis	“tw@music.com”
7	BANNER	VARCHAR(2048)	NULL	Link gambar banner profil artis	https://cdn.com/banner11.png
8	FOLLOWER_COUNT	INT8	NULL, default 0	Jumlah pengikut	45000000

Tabel 4 Tabel Songs

Nama		SONGS			
Deskripsi		Menampung informasi detail lagu seperti file path, judul, durasi, statistik popularitas, metrik audio (valence, acousticness, energy), tanggal rilis, dan rating lagu.			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	SONG_ID	INT4	PK, NOT NULL	ID unik lagu	510
2	SONG_FILE	VARCHAR(320)	NOT NULL	Lokasi file audio	“/songs/track 510.mp3”
3	SONG_TITLE	VARCHAR(255)	NOT NULL	Judul lagu	“Blinding Lights”

4	LISTEN_COUNT	INT8	NULL, default 0	Total jumlah diputar	2300000000
5	SONG_CREDITS	TEXT	NULL	Keterangan produser/artis	"Produced by Max Martin"
6	SONG_DURATION	INT4	NOT NULL, value > 0	Durasi lagu (detik)	201
7	VALENCE	DECIMAL(4,3)	NULL, 0 <= value <= 1	Mood positif lagu	0.520
8	ACCOUSTICNESS	DECIMAL(4,3)	NULL, 0 <= value <= 1	Kelembutan suara	0.11
9	DANCEABILITY	DECIMAL(4,3)	NULL, 0 <= value <= 1	Kelayakan lagu untuk menari	0.803
10	ENERGY	DECIMAL(4,3)	NULL, 0 <= value <= 1	Energi audio	0.734
11	POPULARITY	NUMERIC(3,0)	NULL, 0 <= value <= 100	Skor popularitas (0-100). Rumus seperti berikut: popularitas = (jumlah_play_30_hari_terakhir / jumlah_play_lagu_terbanyak_dalam_sistem_30_hari_terakhir) * 100	87
12	SONG_RELEASE_DATE	DATE	NOT NULL	Tanggal rilis	"2020-03-20"
13	SONG_RATING	NUMERIC(3,0)	NULL	Rating rata-rata	9

Tabel 5 Tabel Genres

Nama		GENRES			
Deskripsi		Menyimpan daftar genre musik yang digunakan untuk mengategorikan lagu dan koleksi.			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	GENRE_ID	INT4	PK, NOT NULL	ID unik genre	30

2	GENRE_NAME	VARCHAR(255)	NOT NULL	Nama genre	"Noise Pop"
---	------------	--------------	----------	------------	-------------

Tabel 6 Tabel Playlists

Nama		PLAYLISTS			
Deskripsi		Menyimpan informasi playlist yang dibuat oleh user. Playlist dapat berisi banyak lagu dan memiliki sampul, deskripsi, serta status publik/privat.			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	PLAYLIST_ID	INT4	PK, NOT NULL	ID unik playlist	7001
2	PLAYLIST_TITLE	VARCHAR(255)	NOT NULL	Nama playlist	"Late Night Vibes"
3	ISPUBLIC	BOOL	NOT NULL, default false	Status publik/privat	TRUE
4	PLAYLIST_COVER	VARCHAR(2048)	NULL	URL sampul playlist	https://img.com/pl7001.jpg
5	USER_ID	INT4	FK USERS.USER_ID	Pembuat playlist	101
6	ISCOLLABORATIVE	BOOL	NOT NULL, default false	Menandakan apakah playlist bersifat kolaboratif	TRUE
7	PLAYLIST_DESC	TEXT	NULL	Deskripsi playlist	"A curated mix of synthwave and chill electronic beats."
8	ISONPROFILE	BOOL	NOT NULL	Menandakan apakah playlist ditampilkan di halaman profil user	TRUE

9	PLAYLIST_DATE_CREATE	DATE	NOT NULL	Tanggal saat playlist dibuat oleh user.	"2024-12-20"
---	----------------------	------	----------	---	--------------

Tabel 7 Tabel PL_Library

Nama		PL_LIBRARY			
Deskripsi		Menunjukkan playlist apa saja yang disimpan (added to library) oleh user. Berbeda dari playlist yang dibuat sendiri.			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	USER_ID	INT4	PK, FK USER.USER_ID, NOT NULL	User pemilik library	101
2	PLAYLIST_ID	INT4	PK, FK PLAYLISTS.PLAY LIST_ID, NOT NULL	Playlist yang disimpan	7001

Tabel 8 Tabel Add_Songs_Playlists

Nama		ADD_SONGS_PLAYLISTS			
Deskripsi		Menyimpan daftar lagu yang ditambahkan ke playlist beserta urutan lagu dalam playlist.			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	PLAYLIST_ID	INT4	FK PLAYLISTS.PLAY LIST_ID, NOT NULL	Playlist	7001
2	SONG_ID	INT4	FK SONGS.SONG_I D, NOT NULL	Lagu yang masuk playlist	510
3	NO_URUT	INT4	NOT NULL, value > 0	Urutan lagu dalam playlist	1
4	USER_ID	INT4	FK, NOT NULL	User yang menambahkan lagu ke playlist	101

5	TIMESTAMP	TIMESTAMP	NULL	Timestamp lagu ditambahkan ke playlist	"2025-01-19 00:00:00"
6	ADD_SONG_PL_ID	INT4	PK	ID entry user memasukkan lagu ke playlist	1

Tabel 9 Tabel Like_Songs

Nama		LIKE_SONGS			
Deskripsi		Menyimpan daftar lagu yang disukai oleh user.			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	USER_ID	INT4	PK, FK USERS.USER_ID	User yang menyukai	101
2	SONG_ID	INT4	PK, FK SONGS.SONG_ID	Lagu yang disukai	510
3	TIMESTAMP	TIMESTAMP	NOT NULL, default current_timestamp	Timestamp user memberikan like pada lagu	"2025-01-19 00:00:00"

Tabel 10 Tabel Listens

Nama		LISTENS			
Deskripsi		Mencatat riwayat pemutaran lagu oleh user (untuk rekomendasi dan statistik personal).			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	USER_ID	INT4	FK USERS.USER_ID , NOT NULL	User yang mendengarkan	101
2	SONG_ID	INT4	FK SONGS.SONG_ID , NOT NULL	Lagu yang diputar	510

3	DURATION_LISTENED	INT4	NOT NULL	Selama berapa detik lagu diputar	122
4	TIMESTAMP	DATE	NOT NULL, default current_timestamp	Timestamp aktivitas mendengarkan lagu	"2025-01-20"
5	LISTEN_ID	INT4	NOT NULL, PK	ID satu aktivitas listening	3

Tabel 11 Tabel Follow_Users

Nama		FOLLOW_USERS			
Deskripsi		Mencatat hubungan "follow" antar sesama pengguna. User dapat mengikuti user lain untuk melihat aktivitas dan playlist mereka.			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	FOLLOWER_ID	INT4	PK, FK USERS.USER_ID , FOLLOWER_ID <> FOLLOWED_ID	User yang mengikuti	101
2	FOLLOWED_ID	INT4	PK, FK USERS.USER_ID , FOLLOWER_ID <> FOLLOWED_ID	User yang di-follow	205

Tabel 12 Tabel Follow_Artists

Nama		FOLLOW_ARTISTS			
Deskripsi		Mencatat user yang mengikuti artis. Data ini digunakan untuk personalisasi konten dan statistik follower artis.			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	USER_ID	INT4	PK, FK USERS.USER_ID	User yang mengikuti artis	101

2	ARTIST_ID	INT4	PK, FK ARTISTS.ARTIST_ID	Artis yang di-follow	11
3	TIMESTAMP	TIMESTAMP	NOT NULL, default current_timestamp	Timestamp mengikuti artist	"2025-01-20 00:00:00"

Tabel 13 Tabel Collections

Nama		COLLECTIONS			
Deskripsi		Merepresentasikan kumpulan lagu seperti Album, EP, atau Kompilasi. Setiap collection dimiliki oleh artis, memiliki tanggal rilis, deskripsi, rating, dan dapat memiliki banyak disc.			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	COLLECTION_ID	INT4	PK, NOT NULL	ID unik collection	2
2	COLLECTION_TITLE	VARCHAR(255)	NOT NULL	Nama album/EP/ko mpilasi	"After Hours"
3	COLLECTION_RELEASE_DATE	DATE	NOT NULL	Tanggal rilis	"2020-03-20"
4	COLLECTION_TYPE	CHAR(11)	NOT NULL, value in (‘Album’, ‘EP’, ‘Single’, ‘Compilation’)	Jenis collection (Album/EP/etc.)	"Album"
5	COLLECTION_COVER	VARCHAR(2048)	NULL	URL cover album atau EP	https://img.com/coll1001.jpg
6	COLLECTION_RATING	NUMERIC(3,0)	NULL	Rating rata-rata	9
7	ISPRERELEASE	BOOL	NULL	Menandakan apakah collection masih prerelease	FALSE

				(belum dirilis penuh)	
--	--	--	--	-----------------------	--

Tabel 14 Tabel Songs_Genres

Nama		SONGS_GENRES			
Deskripsi		Menghubungkan lagu dan genre dalam relasi many-to-many (karena satu lagu dapat memiliki beberapa genre).			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	GENRE_ID	INT4	PK, FK GENRES.GENRE_ID, NOT NULL	Genre lagu	30
2	SONG_ID	INT4	PK, FK SONGS.SONG_ID, NOT NULL	Lagu yang memiliki genre	510

Tabel 15 Tabel Collection_Top_3_Genre

Nama		COLLECTION_TOP_3_GENRE			
Deskripsi		Menghubungkan collection (album/EP/kompilasi) dengan genre. Satu album dapat mencakup beberapa genre.			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	COLLECTION_ID	INT4	PK, FK COLLECTIONS.COLLECTION_ID, NOT NULL	Koleksi/album	20
2	GENRE_ID	INT4	PK, FK GENRES.GENRE_ID, NOT NULL	Genre yang terkait	30

Tabel 16 Tabel Collections_Songs

Nama		COLLECTIONS_SONGS
Deskripsi		Relasi yang menghubungkan lagu dengan collection (album/EP/kompilasi). Karena satu collection dapat berisi banyak lagu, dan sebuah lagu dapat

		muncul di beberapa collection (misalnya kompilasi), tabel ini menjadi penghubung many-to-many.			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	COLLECTION_ID	INT4	PK, FK COLLECTIONS.COLLECTION_ID	ID collection	1001
2	SONG_ID	INT4	PK, FK SONGS.SONG_ID	ID lagu	510
3	NOMOR_DISC	INT4	NULL, value > 0	Disc ke berapa lagu ditempatkan (untuk album dengan banyak disc)	1
4	NOMOR_TRACK	INT4	NULL, value > 0	Track number (urutan lagu) dalam disc	3

Tabel 17 Tabel Releases

Nama		RELEASES			
Deskripsi		Menghubungkan artis dengan collection (album/ EP/ kompilasi) secara eksplisit. Diperlukan untuk kasus kolaborasi album atau kompilasi dengan banyak artis.			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	ARTIST_ID	INT4	PK, FK ARTISTS.ARTIST_ID, NOT NULL	Artis pemilik	11
2	COLLECTION_ID	INT4	PK, FK COLLECTIONS.COLLECTION_ID, NOT NULL	Collection terkait	12

Tabel 18 Tabel Create_Songs

Nama	CREATE_SONGS
------	--------------

Deskripsi		Menghubungkan artis dengan lagu. Satu lagu bisa memiliki banyak artis (kolaborasi), sementara satu artis punya banyak lagu.			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	ARTIST_ID	INT4	PK, FK ARTISTS.ARTIST _ID, NOT NULL	Artis yang berkontribusi	11
2	SONG_ID	INT4	PK, FK SONGS.SONG_I D, NOT NULL	Lagu hasil karya/artis tersebut	510

Tabel 19 Tabel Artist_Promotion

Nama		ARTIST_PROMOTION			
Deskripsi		Fitur unik sistem: artis dapat mempromosikan satu lagu atau satu collection di halaman profilnya beserta pesan promosi.			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	ARTIST_ID	INT4	PK, FK ARTISTS.ARTIST _ID, NOT NULL, REFERENCES RELEASES (ARTIST_ID, COLLECTION_ID)	Artis yang melakukan promosi	11
2	COLLECTION_ID	INT4	FK COLLECTIONS.C LLECTION_ID, NOT NULL, REFERENCES 	Collection yang dipromosikan (opsional)	34
3	KOMENTAR_PROM OSI	TEXT	NULL	Pesan promosi	"New album out now — enjoy!"

Tabel 20 Tabel Tours

Nama	TOURS				
Deskripsi	Menyimpan informasi dasar mengenai tur artis, termasuk nama tur dan jangka waktu penyelenggaraan. Satu artis dapat memiliki banyak tur, dan setiap tur dapat berlangsung di beberapa kota.				
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	TOUR_ID	INT4	PK, NOT NULL	ID unik tur	301
2	TOUR_NAME	VARCHAR(255)	NOT NULL	Nama tur	"After Hours World Tour"
3	TOUR_DATE	DATE	NOT NULL	Tanggal mulai tur	"2023-07-10"
4	VENUE	VARCHAR(255)	NOT NULL	Tanggal selesai tur	"VENUE WAWA, Jakarta, ID"

Tabel 21 Tabel Artists_Tours

Nama	ARTISTS_TOURS				
Deskripsi	Menghubungkan artis dengan tur tertentu. Diperlukan untuk tur dengan lebih dari satu artis (kolaborasi event atau festival).				
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	ARTIST_ID	INT4	PK, FK ARTISTS.ARTIST_ID	Artis yang tampil	11
2	TOUR_ID	INT4	PK, FK TOURS.TOUR_ID	Tur yang diikuti	301

Tabel 22 Tabel Rate_Songs

Nama	RATE_SONGS	
Deskripsi	Menyimpan rating numerik (1–100) yang diberikan user terhadap lagu tertentu. Digunakan untuk menghitung rata-rata rating lagu yang tersimpan di tabel SONGS.	

No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh Data
1	USER_ID	INT4	PK, FK USERS.USER_ID	User yang memberi rating	101
2	SONG_ID	INT4	PK, FK SONGS.SONG_ID	Lagu yang diberi rating	510
3	SONG_RATING	NUMERIC(3,0)	NOT NULL, 1-100	Nilai rating (1-100)	9
4	TIMESTAMP	TIMESTAMP	NOT NULL, default current_timestamp	TIMESTAMP rating song	"2025-01-20 00:00:00"

Tabel 23 Tabel Block_Users

Nama		BLOCK_USERS			
Deskripsi		Menyimpan relasi “user mem-block user lain”. Digunakan agar user tertentu tidak dapat melihat/berinteraksi satu sama lain.			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh
1	BLOCKER_ID	INT4	PK, FK USERS.USER_ID , NOT NULL, BLOCKER_ID <> BLOCKED_ID	User yang memblokir	5
2	BLOCKED_ID	INT4	PK, FK USERS.USER_ID , NOT NULL, BLOCKER_ID <> BLOCKED_ID	User yang diblokir	12

Tabel 24 Tabel Blocklist_Users

Nama		BLOCKLIST_ARTISTS			
Deskripsi		Daftar artist yang diblokir oleh user (hide artist).			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh

1	ARTIST_ID	INT4	PK, FK ARTISTS.ARTIST_ID, NOT NULL	Artist yang diblokir	7
2	USER_ID	INT4	PK, FK USERS.USER_ID, NOT NULL	User yang memblokir	3

Tabel 25 Tabel Collection_Library

Nama		COLLECTION_LIBRARY			
Deskripsi		Menyimpan koleksi (album/EP/single/Compilation) yang disimpan oleh user (add to library).			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh
1	USER_ID	INT4	PK, FK USERS.USER_ID, NOT NULL	User yang menyimpan koleksi	10
2	COLLECTION_ID	INT4	PK, FK COLLECTIONS.COLLECTION_ID, NOT NULL	Koleksi yang disimpan	2

Tabel 26 Tabel Like_Reviews

Nama		LIKE_REVIEWS			
Deskripsi		Daftar “like” yang diberikan user terhadap ulasan (review).			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh
1	REVIEW_ID	INT4	PK, FK REVIEWS.REVIEW_ID, NOT NULL	Review yang dilike	14
2	USER_ID	INT4	PK, FK USERS.USER_ID, NOT NULL	User yang memberikan like	22

Tabel 27 Tabel Reviews

Nama	REVIEWS

Deskripsi		Review dan rating user terhadap sebuah collection (album/EP/single).			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh
1	REVIEW	TEXT	NULL	Isi review	"Albumnya sangat keren!"
2	RATING	NUMERIC(3,0)	NOT NULL, CHECK 1–100	Rating koleksi	85
3	TIMESTAMP	TIMESTAMP	NOT NULL, DEFAULT current_timestamp	Waktu review ditulis	2025-11-26 10:00
4	REVIEW_ID	INT4	PK	ID review	18
5	USER_ID	INT4	FK USERS.USER_ID , NOT NULL	User yang review	4
6	COLLECTION_ID	INT4	FK COLLECTIONS.COLLECTION_ID, NOT NULL	Koleksi yang direview	7

Tabel 28 Tabel Socials

Nama		SOCIALS			
Deskripsi		Daftar link sosial media milik artis.			
No	Nama Kolom	Tipe Data	Constraint	Deskripsi	Contoh
1	SOCIAL_ID	INT4	PK	ID sosial media	5
2	ARTIST_ID	INT4	FK ARTISTS.ARTIST_ID, NOT NULL	Artist pemilik akun	12
3	SOCIAL_MEDIA_LINK	VARCHAR(2048)	NOT NULL	URL sosial media	https://instagram.com/artist

4.4.2 Sequence

Pada studi kasus ini, sequence digunakan untuk mengotomatisasi pengisian nilai primary key pada seluruh tabel yang memerlukan ID unik. Sequence memastikan bahwa setiap record baru mendapatkan nilai ID yang berurutan, konsisten, serta tidak terjadi duplikasi ID. Penggunaan sequence juga mempermudah proses insert data dan menjaga integritas referensial antar tabel:

Tabel 29 Sequence

No	Nama Tabel	Nama Kolom	Deskripsi
1.	Users	user_id	Sequence ini digunakan untuk menghasilkan nilai user_id secara otomatis dan berurutan saat data pengguna baru ditambahkan. Sequence ini menjamin setiap pengguna memiliki ID unik tanpa perlu input manual.
2.	Artists	artist_id	Sequence menghasilkan nilai artist_id secara otomatis untuk setiap artis baru. Digunakan untuk menjaga konsistensi ID pada tabel artis.
3.	Songs	Song_id	Sequence digunakan sebagai auto-increment pada kolom song_id saat lagu baru ditambahkan ke sistem.
4.	Playlists	playlist_id	Sequence menghasilkan nilai playlist_id secara otomatis sehingga ID playlist selalu unik dan tidak bentrok.
5.	Collections	collection_id	Sequence digunakan untuk mengisi collection_id secara otomatis ketika koleksi baru ditambahkan.

6.	Genres	genre_id	Sequence menghasilkan nilai genre_id secara otomatis untuk setiap genre yang dibuat.
7.	Reviews	review_id	Sequence digunakan untuk mengisi kolom review_id secara otomatis agar setiap review memiliki ID unik tanpa input manual.
8.	Tours	tour_id	Sequence menghasilkan tour_id secara otomatis untuk setiap data tur konser yang ditambahkan.
9.	Socials	social_id	Sequence pada kolom social_id memastikan data media sosial yang terhubung pada artis memiliki ID unik dan bertambah otomatis.
10.	Listen	Listen_id	Sequence pada kolom listen_id memastikan data listening memiliki ID unik dan bertambah otomatis
11.	Add_Songs_Playlist	Add_Song_PL_id	Sequence pada kolom ini memastikan data entry user menambahkan lagu ke playlist memiliki ID unik dan bertambah secara otomatis

4.4.3 Indexing

Pada studi kasus ini, indexing digunakan untuk meningkatkan performa query dan menjaga integritas data. Setiap primary key otomatis memiliki index unik yang memastikan pencarian record berdasarkan ID dapat dilakukan dengan cepat. Selain itu, foreign key juga dibuatkan index untuk mempercepat operasi join antar tabel serta mendukung proses update dan delete secara efisien. Penggunaan index secara tepat membantu mengoptimalkan akses data, mengurangi waktu eksekusi query, dan menjaga konsistensi referensial di seluruh tabel.

Tabel 30 Indexing

No.	Nama Tabel	Nama Index	Nama Kolom	Keterangan
1	ADD_SONGS_PLAYLISTS	USER_MENAMBAHKAN_FK	USER_ID	index FK ke USERS
2	ADD_SONGS_PLAYLISTS	MENAMBAHKAN_KE_PLAYLIST_FK	PLAYLIST_ID	index FK ke PLAYLISTS
3	ADD_SONGS_PLAYLISTS	MENAMBAHKAN_LAGU_FK	SONG_ID	index FK ke SONGS
4	ARTISTS	ARTISTS_PK	ARTIST_ID	primary key
5	ARTISTS_TOURS	ARTISTS_TOURS_PK	TOUR_ID, ARTIST_ID	primary key
6	ARTISTS_TOURS	MEMILIKI2_FK	ARTIST_ID	index FK ke ARTISTS
7	ARTISTS_TOURS	MEMILIKI3_FK	TOUR_ID	index FK ke TOURS
8	ARTIST_PROMOTION	PK_ARTIST_PROMOTION	ARTIST_ID, COLLECTION_ID	primary key
9	ARTIST_PROMOTION	MEMPROMOSIKAN2_FK	ARTIST_ID	index FK ke ARTISTS
10	ARTIST_PROMOTION	MEMPROMOSIKAN_FK	COLLECTION_ID	index FK ke COLLECTIONS
11	BLOCKLIST_ARTISTS	BLOCKLIST_ARTISTS_PK	ARTIST_ID, USER_ID	primary key
12	BLOCKLIST_ARTISTS	MEM_BLACKLIST_FK	ARTIST_ID	index FK ke ARTISTS
13	BLOCKLIST_ARTISTS	MEM_BLACKLIST2_FK	USER_ID	index FK ke USERS

14	BLOCK_USERS	BLOCK_USERS_PK	BLOCKER_ID, BLOCKED_ID	primary key
15	BLOCK_USERS	MEM_BLOCK_FK	BLOCKER_ID	index FK ke USERS
16	BLOCK_USERS	MEM_BLOCK2_FK	BLOCKED_ID	index FK ke USERS
17	COLLECTIONS	COLLECTIONS_PK	COLLECTION_ID	primary key
18	COLLECTIONS_SONGS	COLLECTIONS_SONGS_PK	SONG_ID, COLLECTION_ID	primary key
19	COLLECTIONS_SONGS	MEMILIKI_FK	COLLECTION_ID	index FK ke COLLECTIONS
20	COLLECTIONS_SONGS	MEMILIKI4_FK	SONG_ID	index FK ke SONGS
21	COLLECTION_LIBRARY	COLLECTION_LIBRARY_PK	USER_ID, COLLECTION_ID	primary key
22	COLLECTION_LIBRARY	RELATIONSHIP_40_FK	USER_ID	index FK ke USERS
23	COLLECTION_LIBRARY	RELATIONSHIP_41_FK	COLLECTION_ID	index FK ke COLLECTIONS
24	COLLECTION_TOP_3_GENRES	COLLECTION_TOP_3_GENRES_PK	COLLECTION_ID, GENRE_ID	primary key
25	COLLECTION_TOP_3_GENRES	TOP_3_GENRE_FK	COLLECTION_ID	index FK ke COLLECTIONS
26	COLLECTION_TOP_3_GENRES	TOP_3_GENRE2_FK	GENRE_ID	index FK ke GENRES

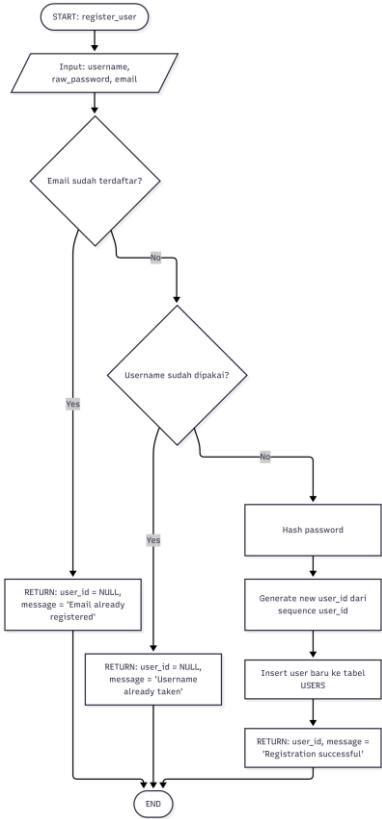
27	CREATE_PLAYLISTS	CREATE_PLAYLISTS_PK	SONG_ID, ARTIST_ID	primary key
28	CREATE_PLAYLISTS	MEMBUAT_FK	SONG_ID	index FK ke SONGS
29	CREATE_PLAYLISTS	MEMBUAT2_FK	ARTIST_ID	index FK ke ARTISTS
30	FOLLOW_ARTISTS	FOLLOW_ARTISTS_PK	USER_ID, ARTIST_ID	primary key
31	FOLLOW_ARTISTS	MENGIKUTI_FK	ARTIST_ID	index FK ke ARTISTS
32	FOLLOW_ARTISTS	MENGIKUTI4_FK	USER_ID	index FK ke USERS
33	FOLLOW_USERS	FOLLOW_USERS_PK	FOLLOWER_ID, FOLLOWED_ID	primary key
34	FOLLOW_USERS	MENGIKUTI2_FK	FOLLOWED_ID	index FK ke USERS
35	FOLLOW_USERS	MENGIKUTI3_FK	FOLLOWER_ID	index FK ke USERS
36	GENRES	GENRES_PK	GENRE_ID	primary key
37	LIKE_REVIEWS	LIKE_REVIEWS_PK	REVIEW_ID, USER_ID	primary key
38	LIKE_REVIEWS	LIKE_REVIEW_FK	REVIEW_ID	index FK ke REVIEWS
39	LIKE_REVIEWS	LIKE REVIEW2_FK	USER_ID	index FK ke USERS
40	LIKE_SONGS	LIKE_SONGS_PK	SONG_ID, USER_ID	primary key
41	LIKE_SONGS	MENYUKAI_FK	USER_ID	index FK ke USERS
42	LIKE_SONGS	MENYUKAI2_FK	SONG_ID	index FK ke SONGS
43	LISTENS	LISTENS_PK	USER_ID, SONG_ID	primary key

44	LISTENS	MENDENGARKAN_FK	SONG_ID	index FK ke SONGS
45	LISTENS	MENDENGARKAN2_FK	USER_ID	index FK ke USERS
46	PLAYLISTS	PLAYLISTS_PK	PLAYLIST_ID	primary key
47	PLAYLISTS	MEMBUAT_PLAYLIST_FK	USER_ID	index FK ke USERS
48	PL_LIBRARY	PL_LIBRARY_PK	USER_ID, PLAYLIST_ID	primary key
49	PL_LIBRARY	RELATIONSHIP_38_FK	USER_ID	index FK ke USERS
50	PL_LIBRARY	RELATIONSHIP_39_FK	PLAYLIST_ID	index FK ke PLAYLISTS
51	RATE_SONGS	RATE_SONGS_PK	USER_ID, SONG_ID	primary key
52	RATE_SONGS	RATE_FK	SONG_ID	index FK ke SONGS
53	RATE_SONGS	RATE2_FK	USER_ID	index FK ke USERS
54	RELEASES	RELEASES_PK	COLLECTION_ID, ARTIST_ID	primary key
55	RELEASES	MERILIS_FK	COLLECTION_ID	index FK ke COLLECTIONS
56	RELEASES	MERILIS2_FK	ARTIST_ID	index FK ke ARTISTS
57	REVIEWS	REVIEWS_PK	REVIEW_ID	primary key
58	REVIEWS	MEMBUAT_REVIEW_FK	USER_ID	index FK ke USERS

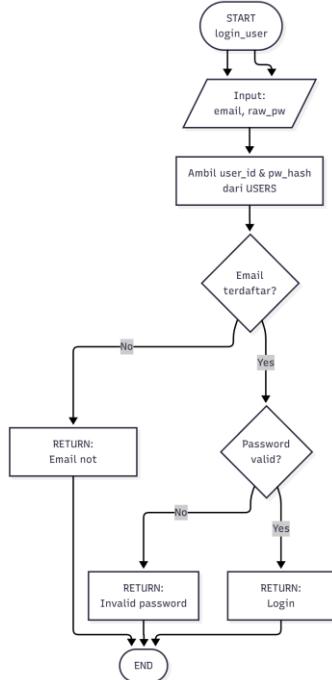
59	REVIEWS	COLLECTION_MANA_FK	COLLECTION_ID	index FK ke COLLECTIONS
60	SOCIALS	SOCIALS_PK	SOCIAL_ID	primary key
61	SOCIALS	HAS_FK	ARTIST_ID	index FK ke ARTISTS
62	SONGS	SONGS_PK	SONG_ID	primary key
63	SONGS_GENRES	SONGS_GENRES_PK	GENRE_ID, SONG_ID	primary key
64	SONGS_GENRES	MEMPUNYAI_FK	GENRE_ID	index FK ke GENRES
65	SONGS_GENRES	MEMPUNYAI2_FK	SONG_ID	index FK ke SONGS
66	TOURS	TOURS_PK	TOUR_ID	primary key
67	USERS	USERS_PK	USER_ID	primary key

4.5 Function/ Procedure

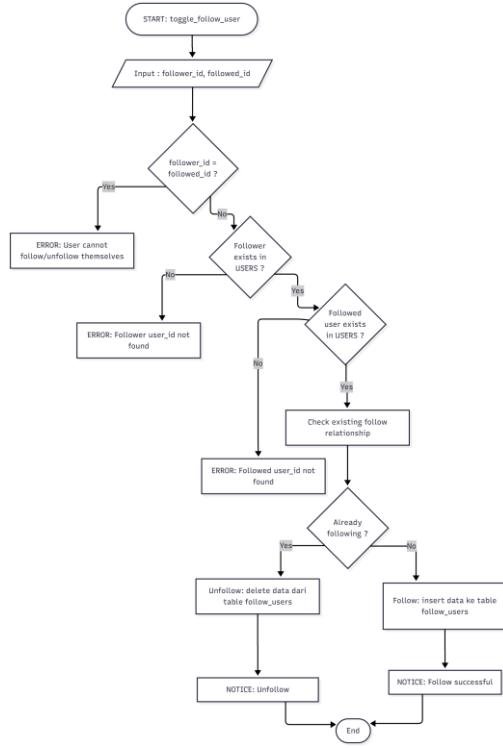
Gambar 4. 4 Flow Chart Function register_user



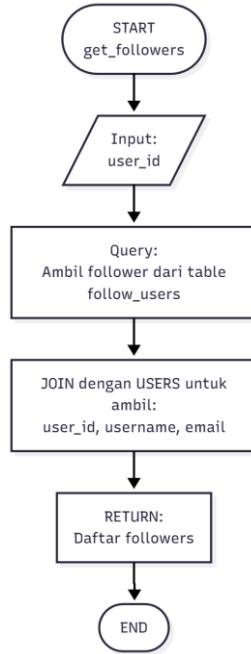
Gambar 4. 5 Flow Chart Function register_user



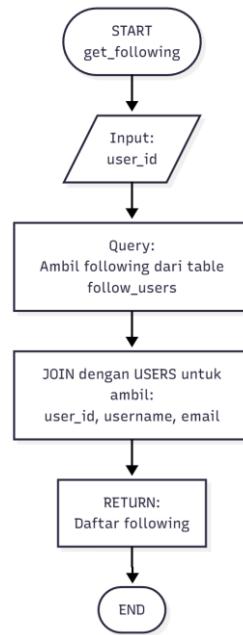
Gambar 4. 6 Flow Chart Function login_user



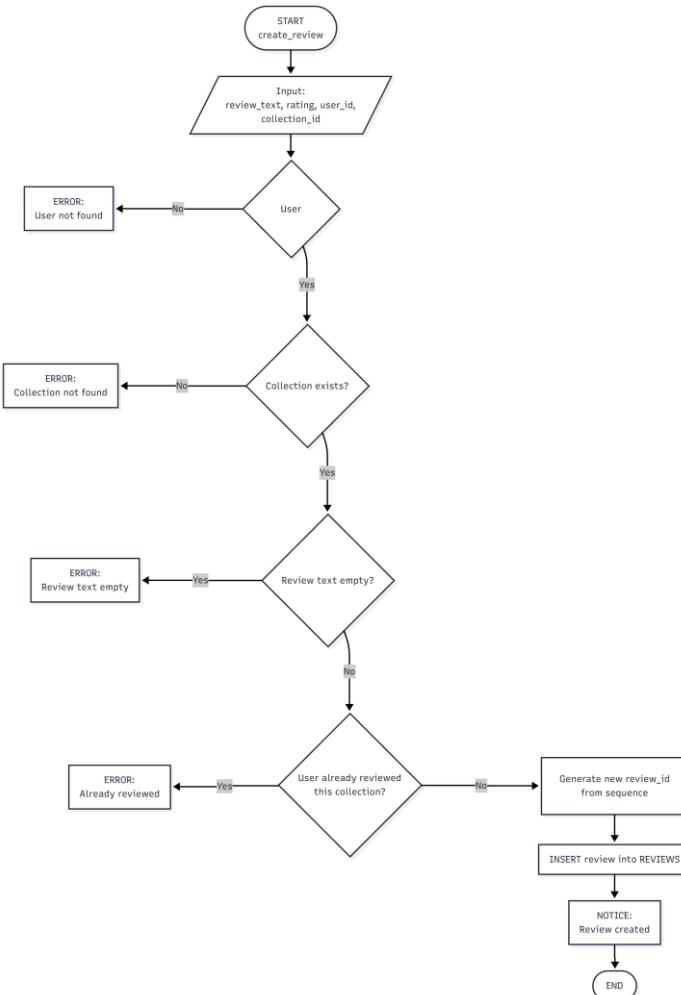
Gambar 4. 7 Flow Chart Procedure `toggle_follow_user`



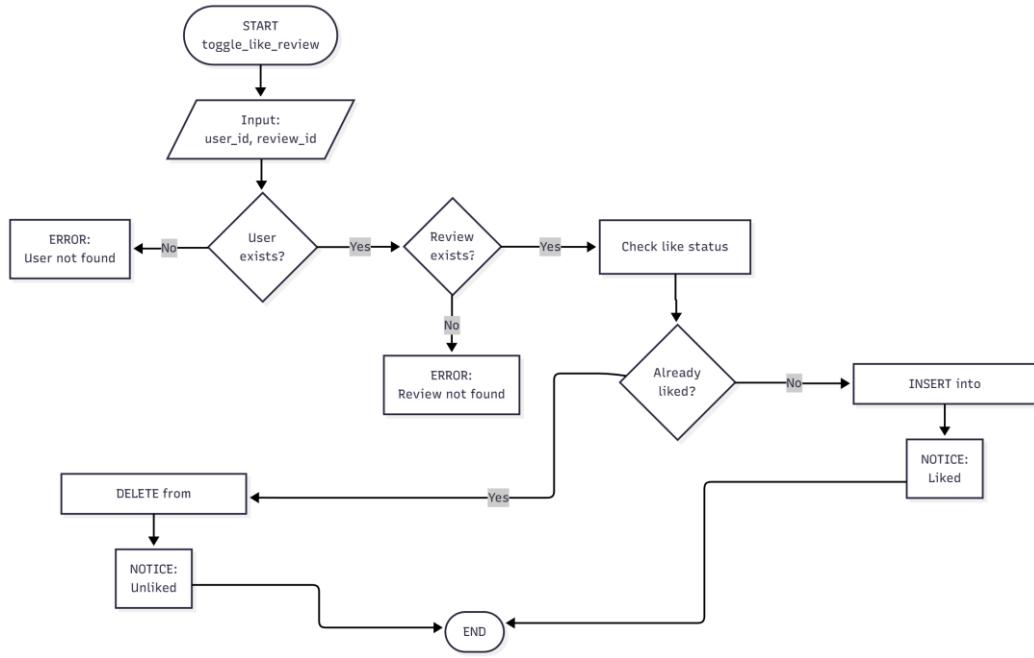
Gambar 4. 8 Flow Chart Function `get_followers`



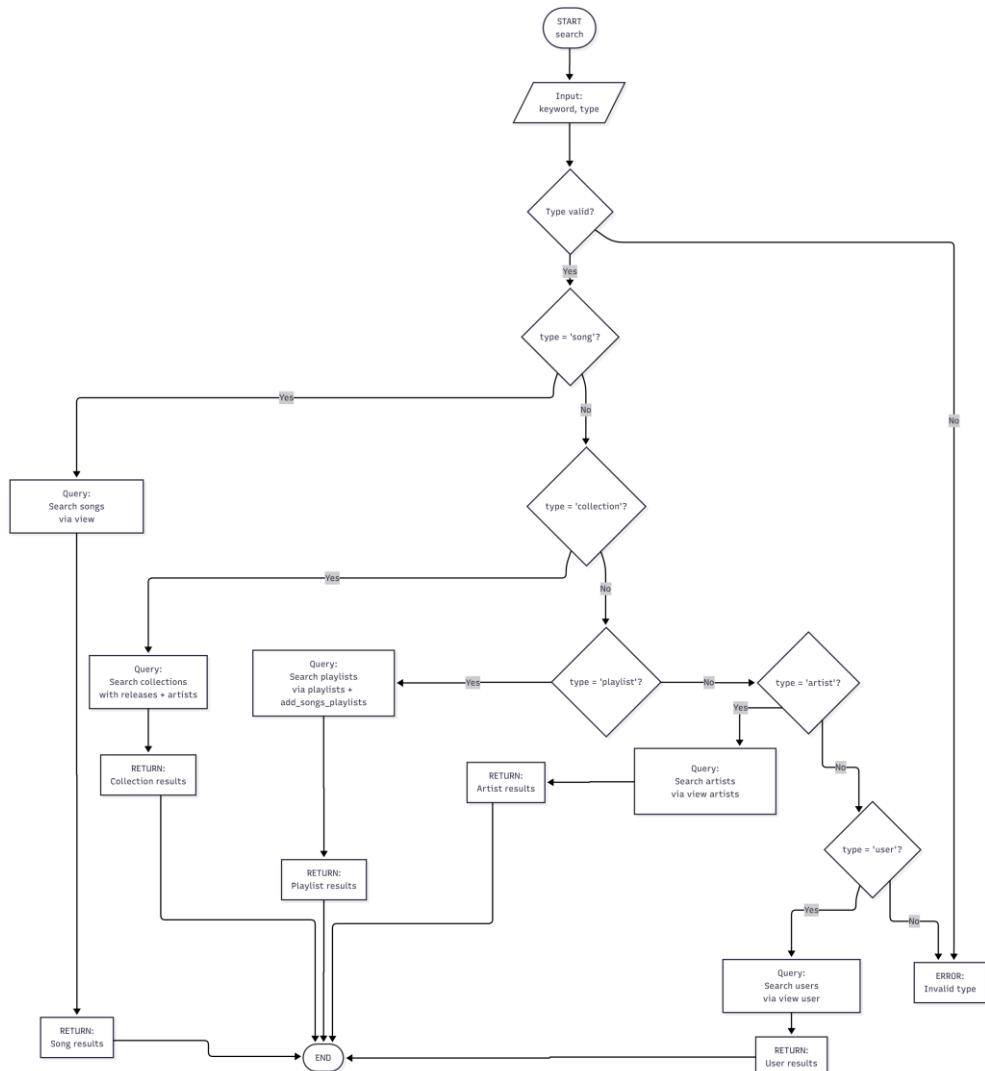
Gambar 4. 9 Flow Chart Function get_following



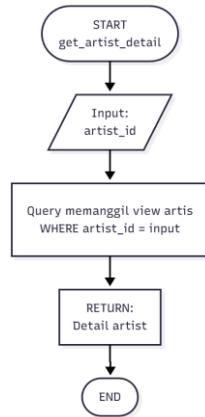
Gambar 4. 10 Flow Chart Procedure create_review



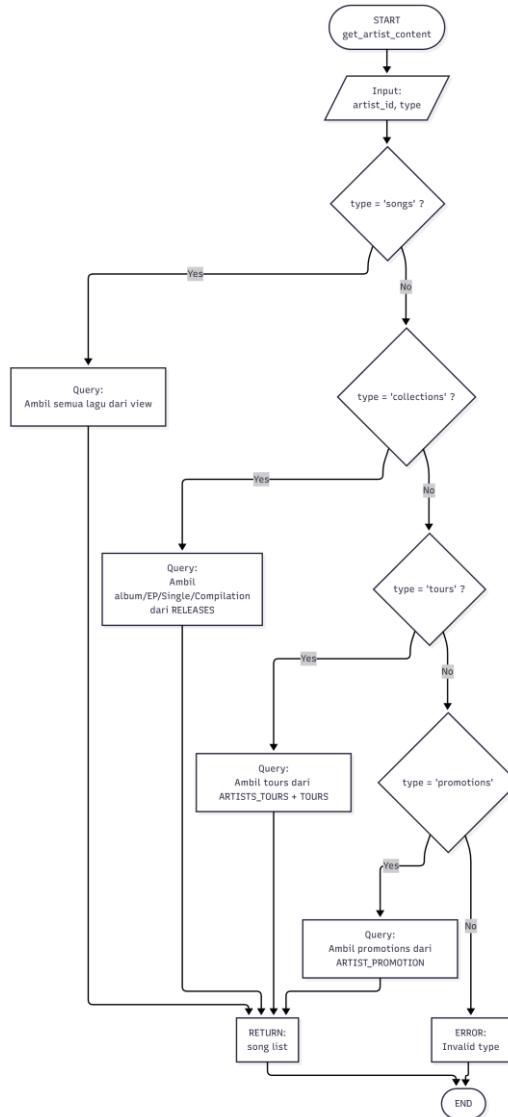
Gambar 4. 11 Flow Chart Procedure `toggle_like_review`



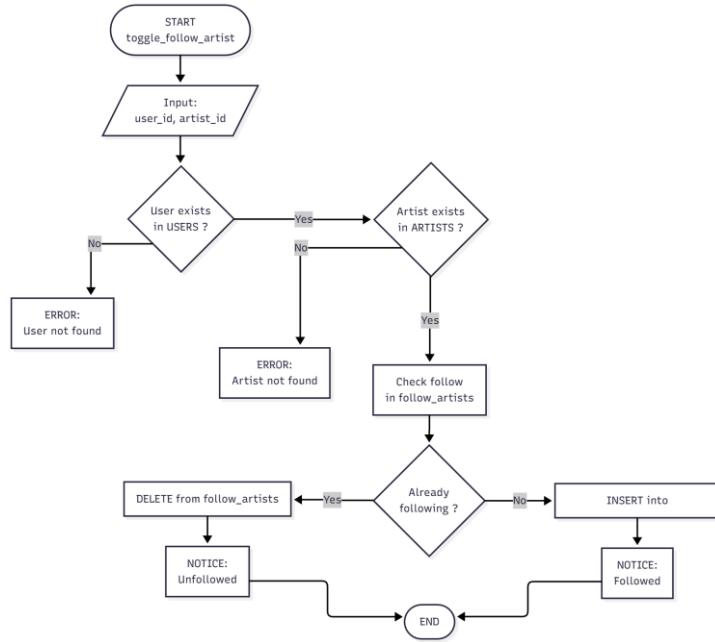
Gambar 4. 12 Flow Chart Function search



Gambar 4. 13 Flow Chart Function get_artist_detail



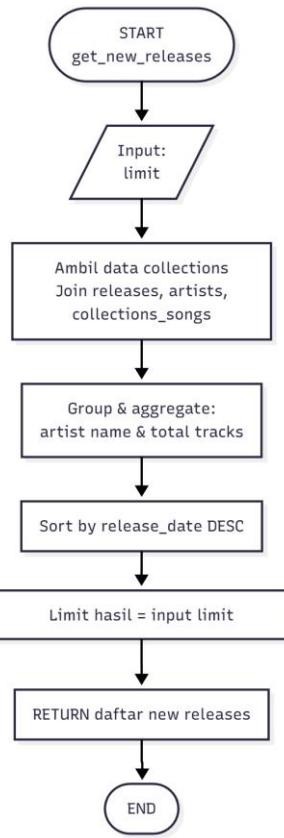
Gambar 4. 14 Flow Chart Function get_artist_content



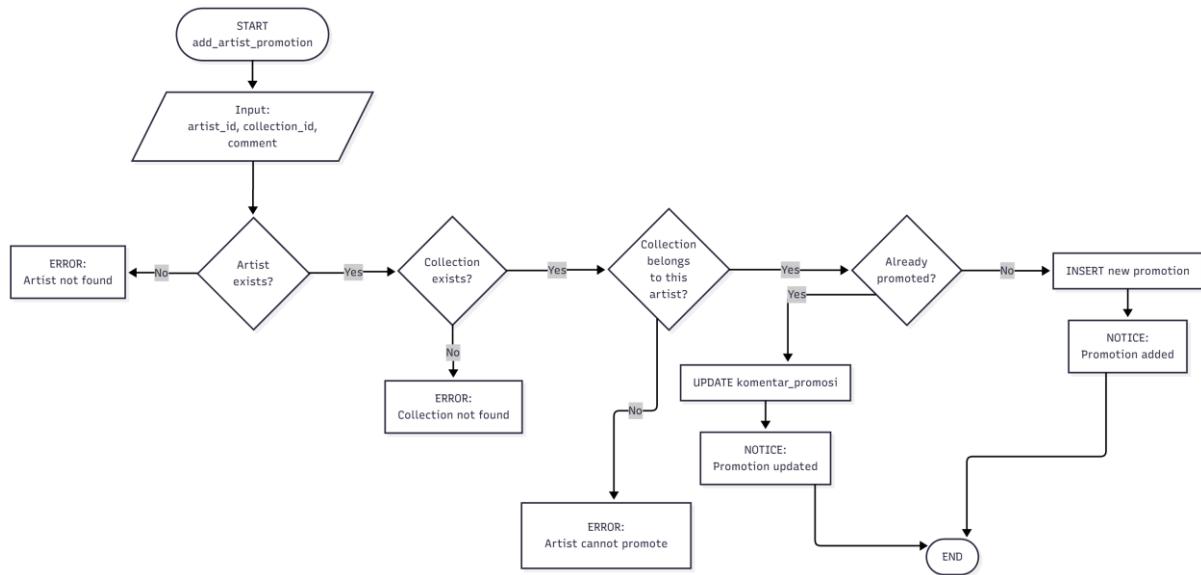
Gambar 4. 15 Flow Chart Procedure `toggle_follow_artist`



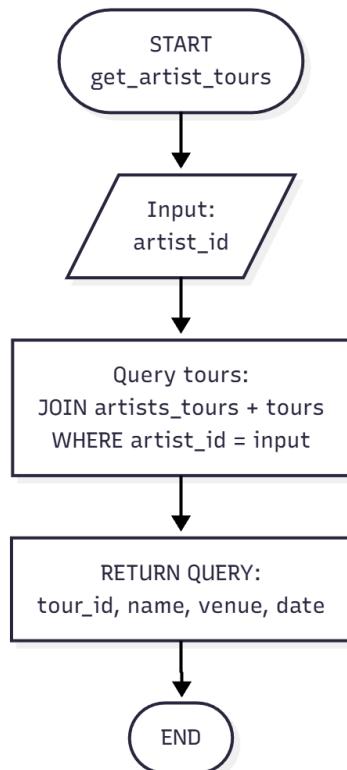
Gambar 4. 16 Flow Chart Function `get_collection_tracks`



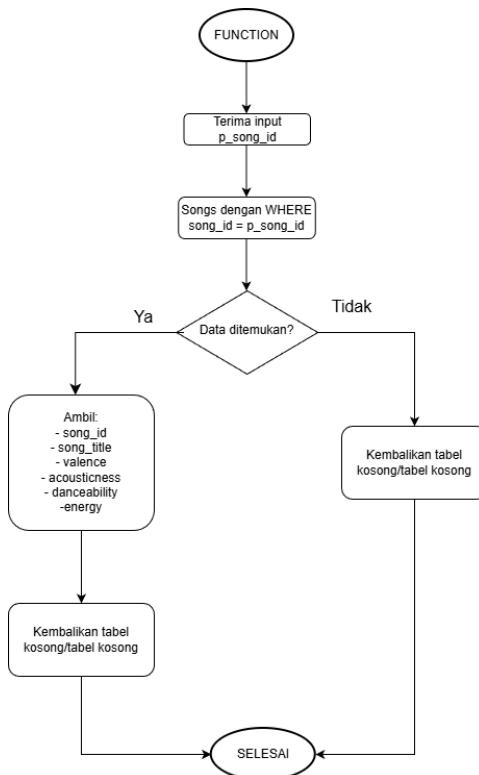
Gambar 4. 17 Flow Chart Function get_new_releases



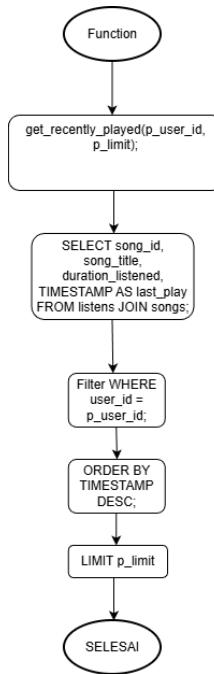
Gambar 4. 18 Flow Chart Procedure add_artist_promotion



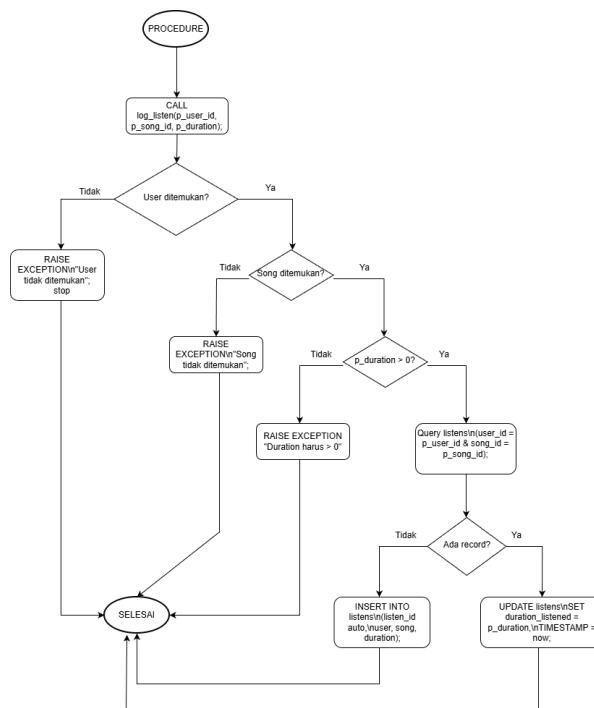
Gambar 4. 19 Flow Chart Function get_artist_tours



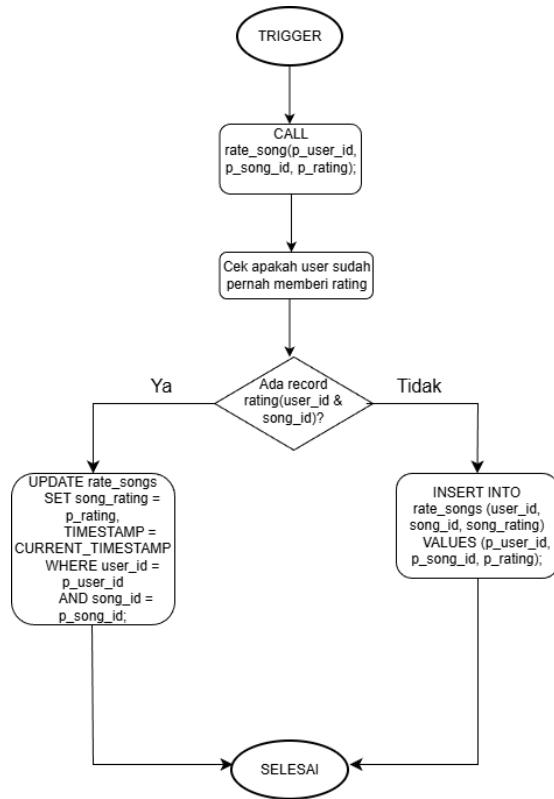
Gambar 4. 20 Flow Chart Function get_song_audio_features



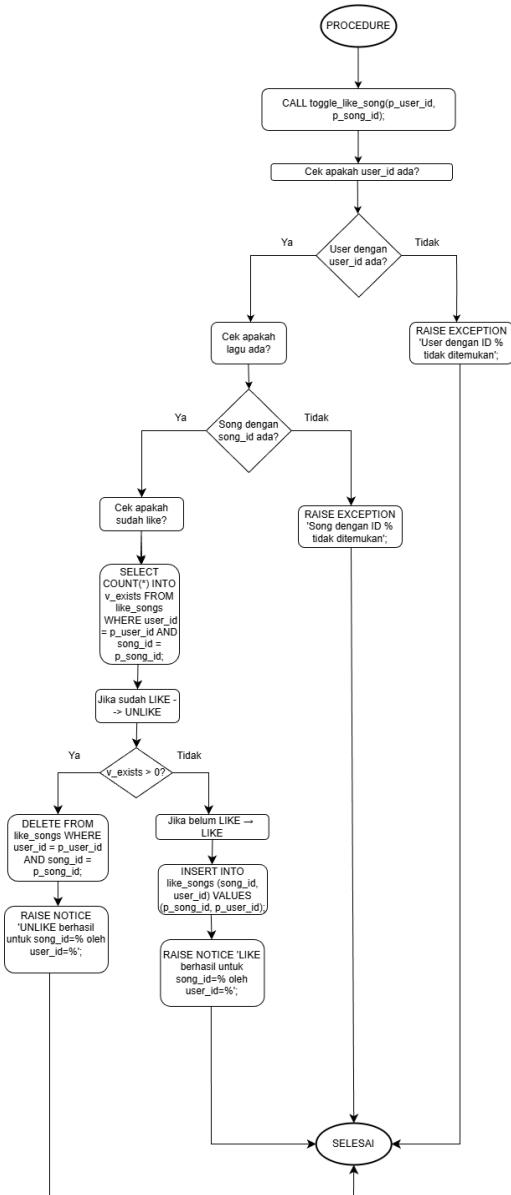
Gambar 4. 21 Flow Chart Function get_recently_played



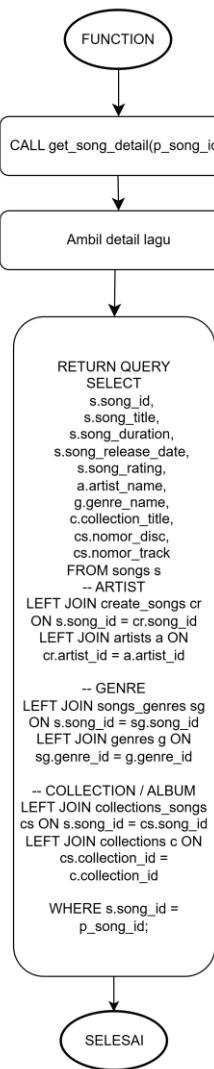
Gambar 4. 22 Flow Chart Procedure log_listen



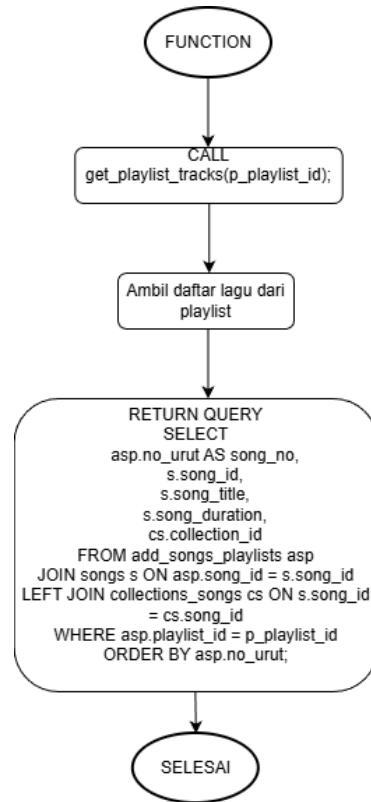
Gambar 4. 23 Flow Chart Procedure rate_song



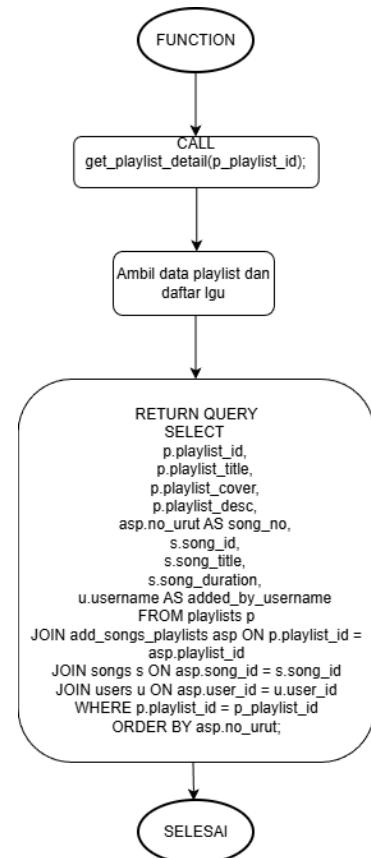
Gambar 4. 24 Flow Chart Procedure `toggle_like_song`



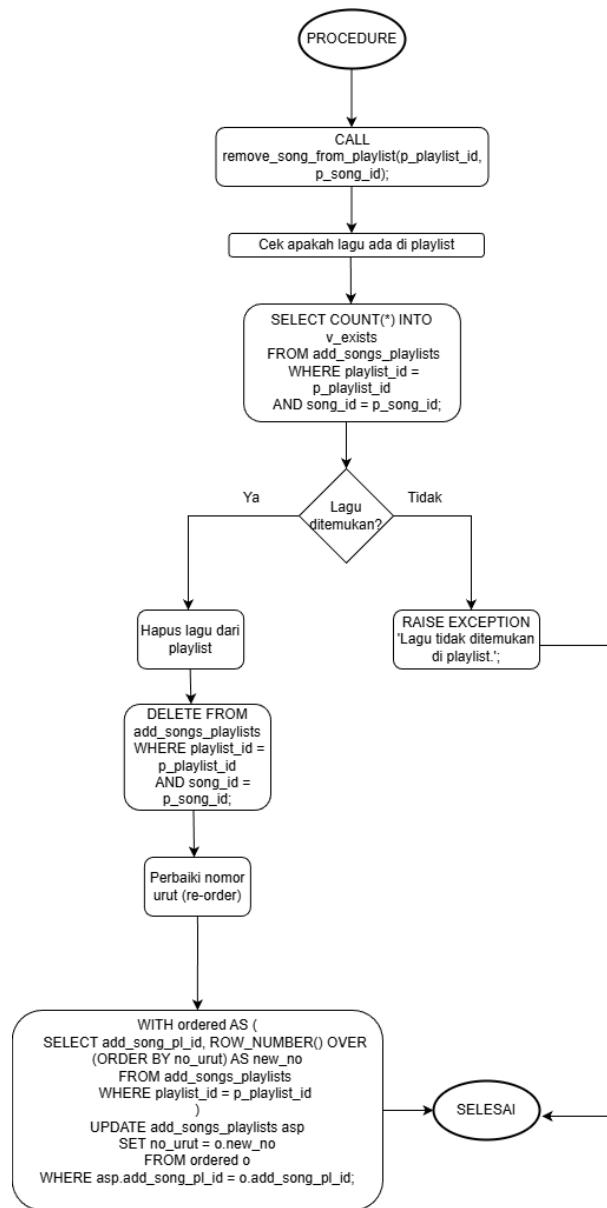
Gambar 4. 25 Flow Chart Function get_song_detail



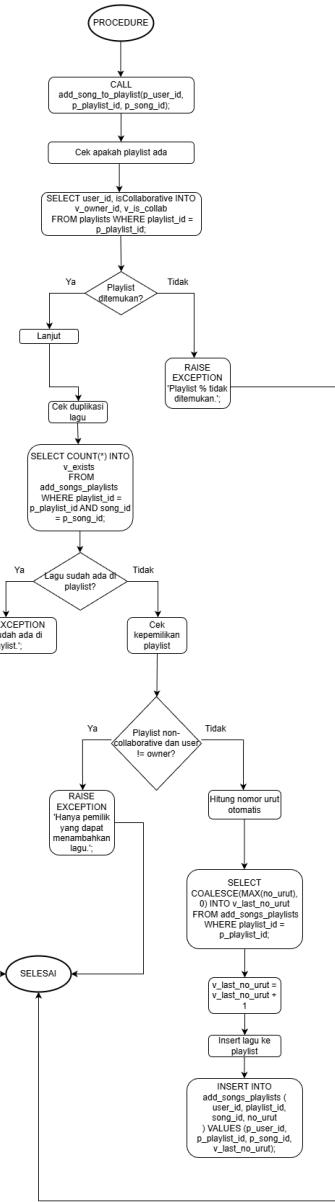
Gambar 4. 26 Flow Chart Function get_playlist_tracks



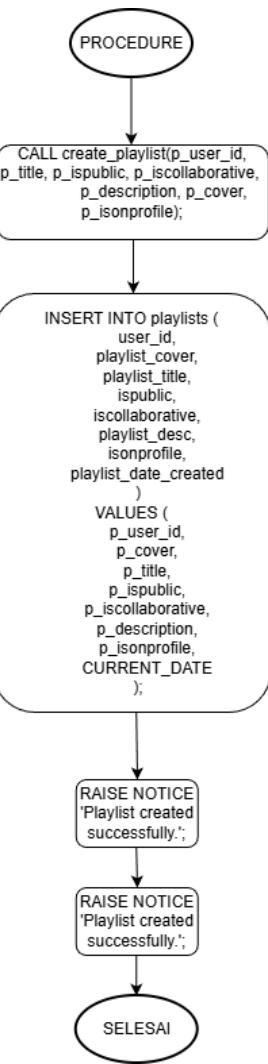
Gambar 4. 27 Flow Chart Function get_playlist_detail



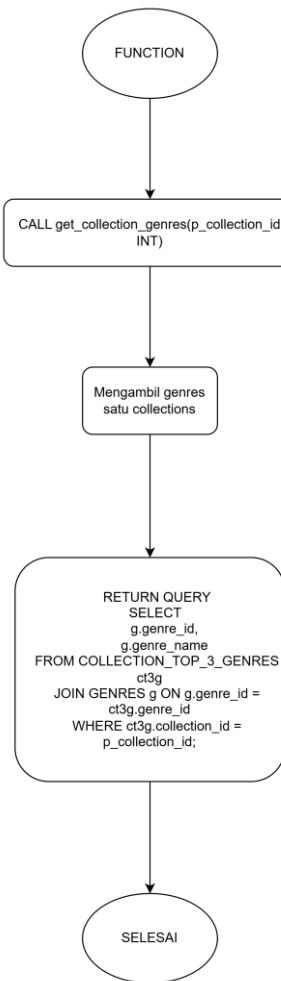
Gambar 4. 28 Flow Chart Procedure remove_song_from_playlist



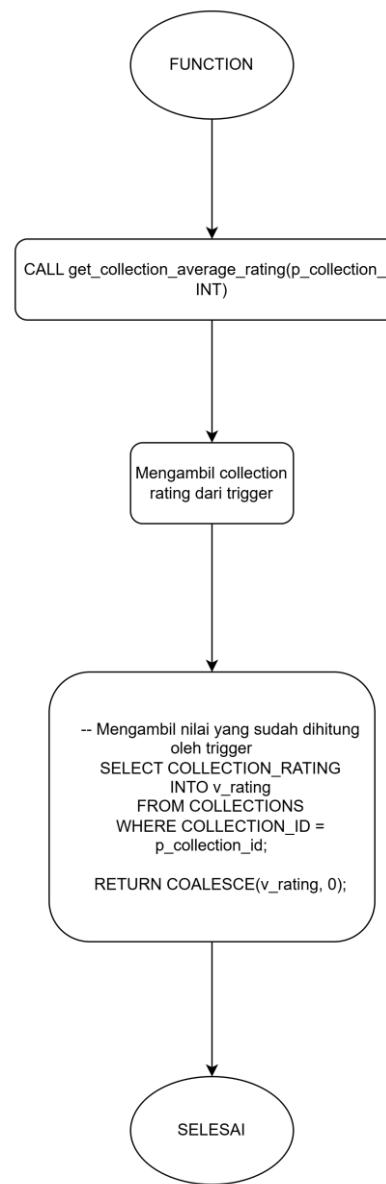
Gambar 4. 29 Flow Chart Procedure add_song_to_playlist



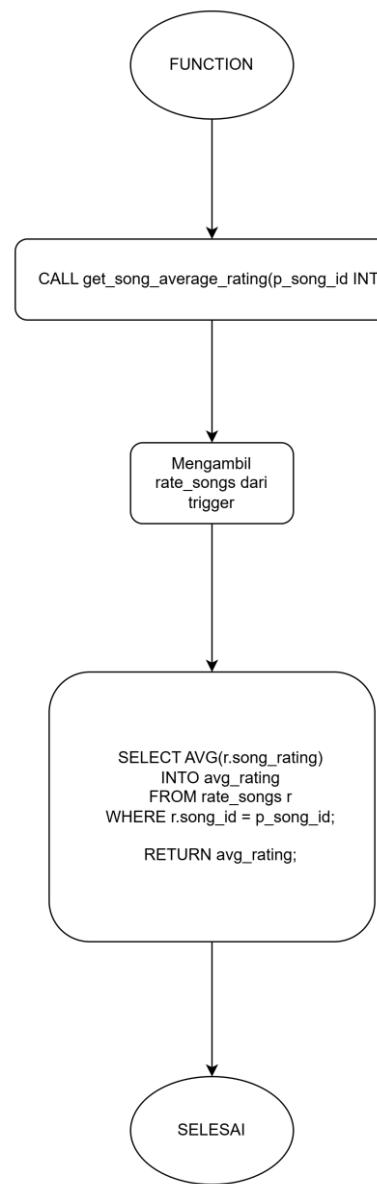
Gambar 4. 30 Flow Chart Procedure create_playlist



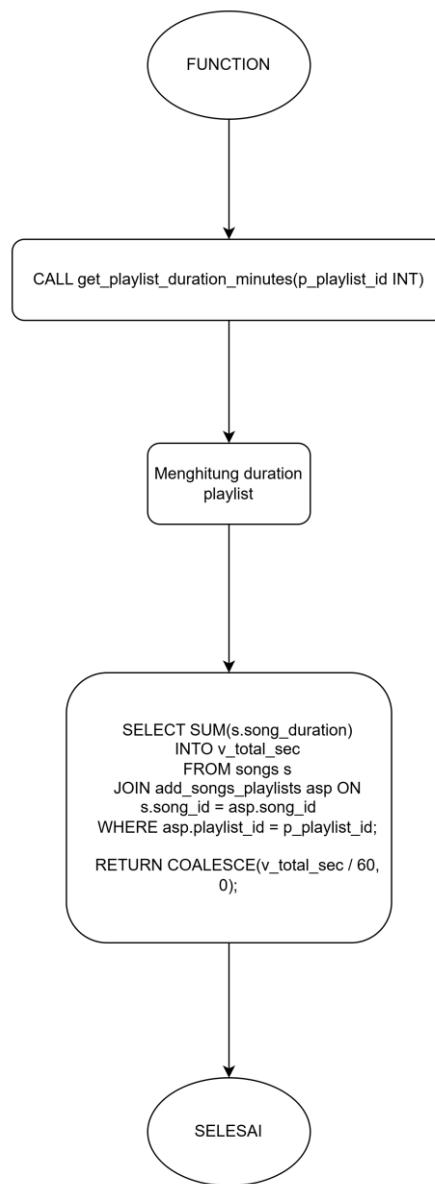
Gambar 4. 31 Flow Chart Function get_collection_genres



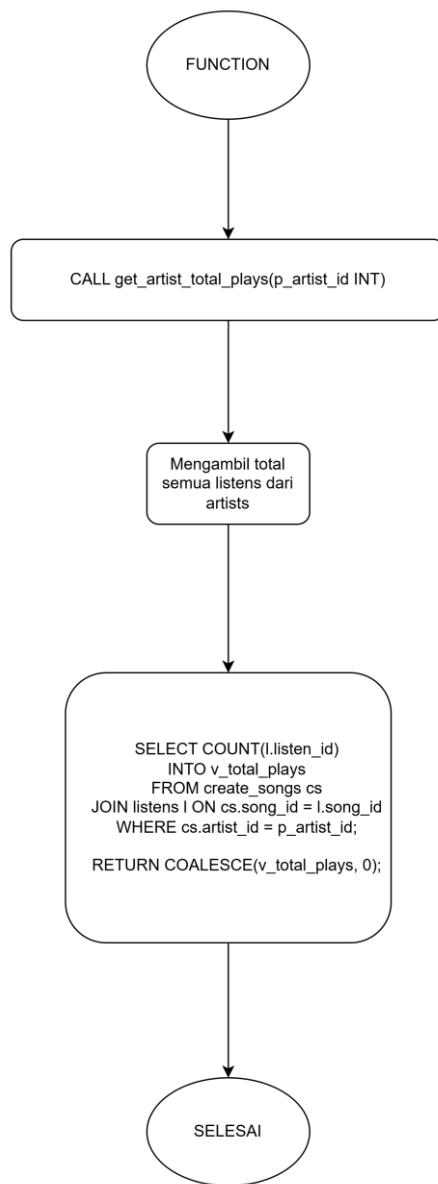
Gambar 4. 32 Flow Chart Function `get_collection_average_rating`



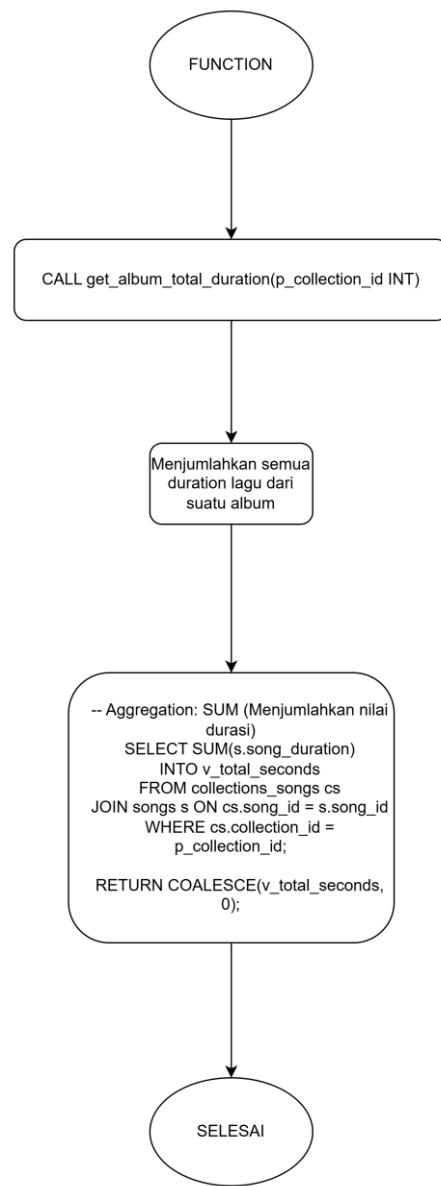
Gambar 4. 33 Flow Chart Function `get_song_average_rating`



Gambar 4. 34 Flow Chart Function `get_playlist_duration_minutes`

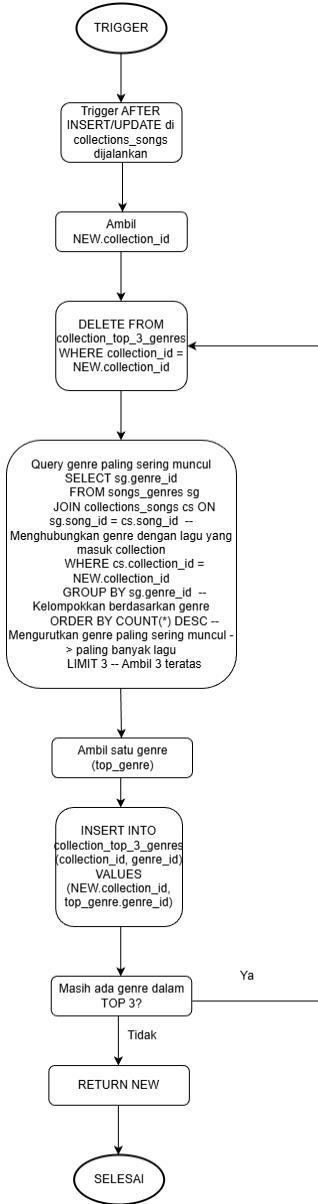


Gambar 4. 35 Flow Chart Function `get_artist_total_plays`

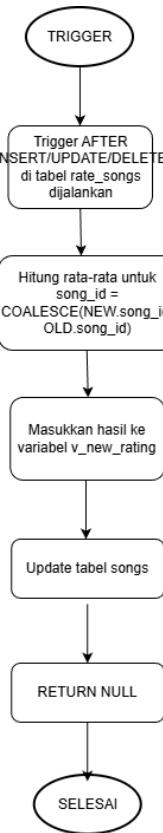


Gambar 4. 36 Flow Chart Function get_album_total_duration

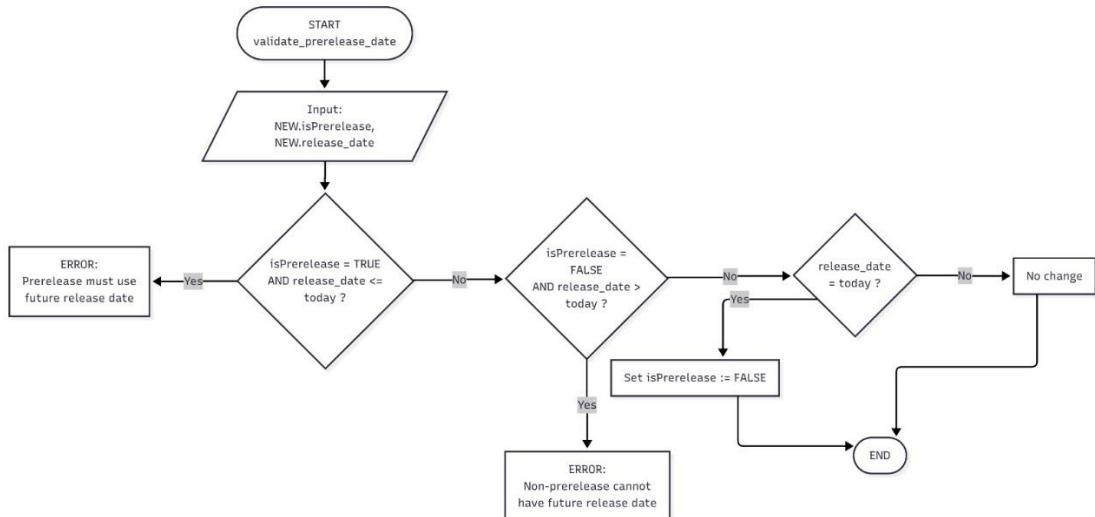
4.6 Trigger



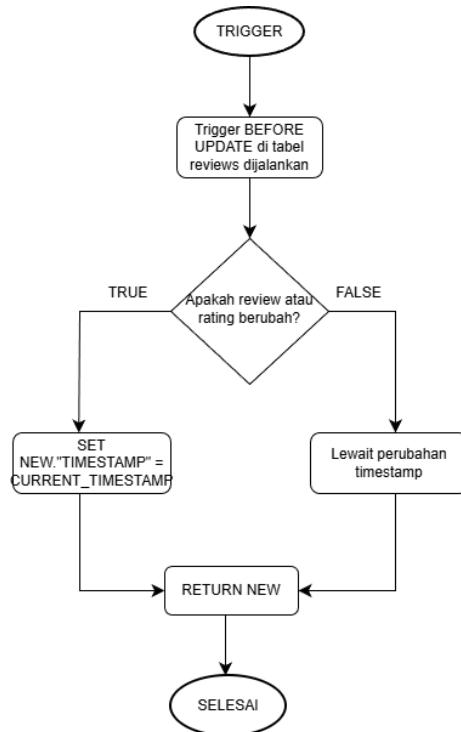
Gambar 4. 37 Flow Chart Trigger update_collection_top3_genres



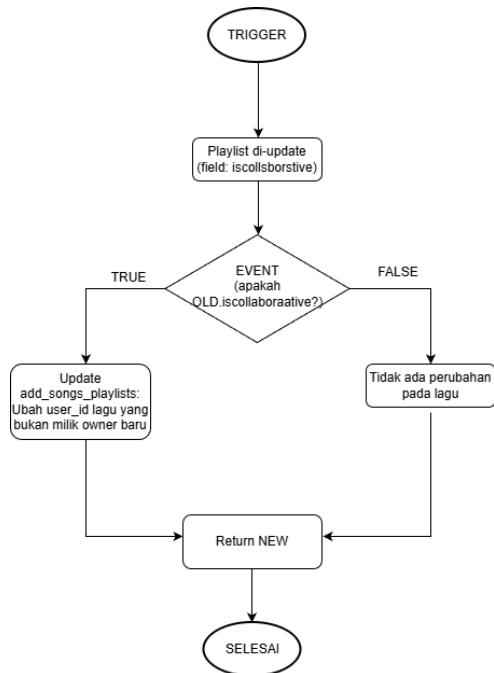
Gambar 4. 38 Flow Chart Trigger update_song_rating



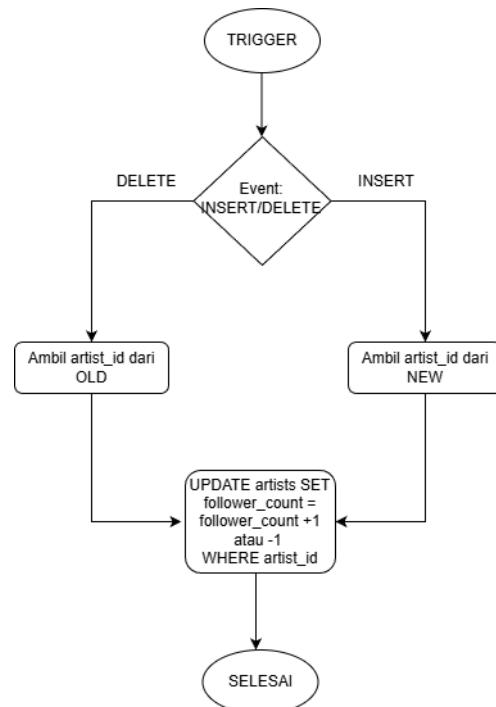
Gambar 4. 39 Flow Chart Trigger trg_validate_prerelease_date



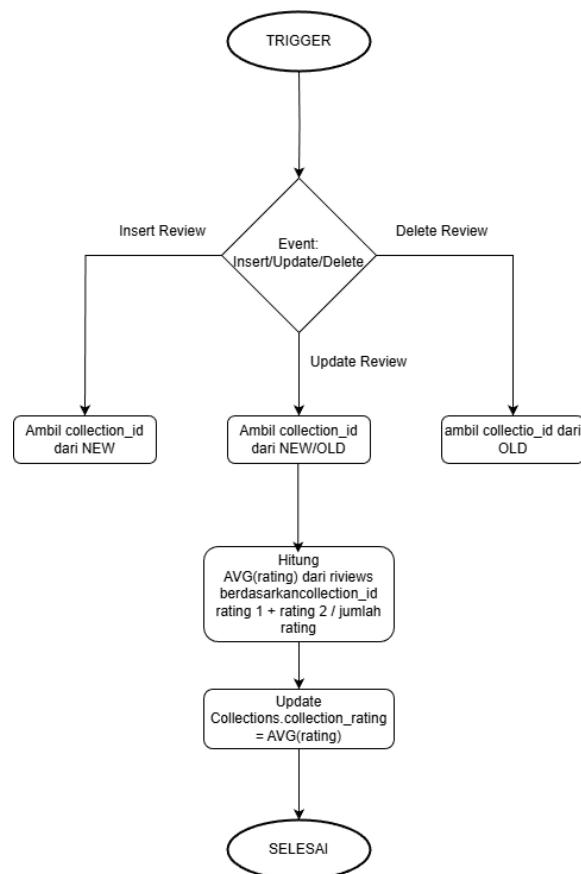
Gambar 4. 40 Flow Chart Trigger update_review_timestamp



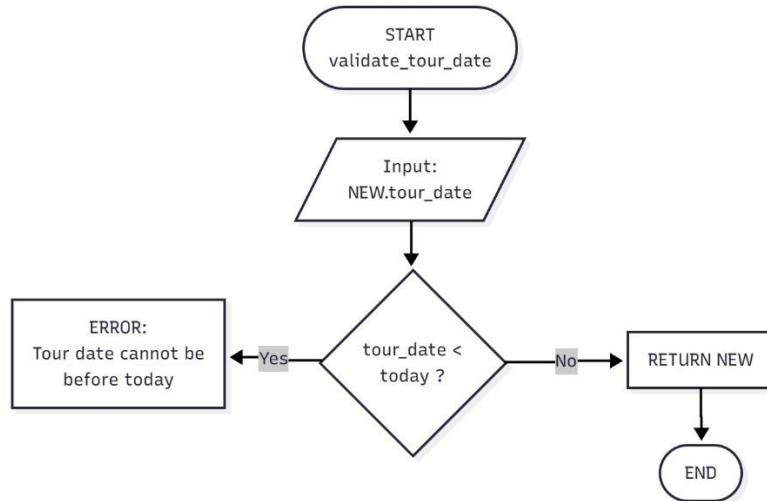
Gambar 4. 41 Flow Chart Trigger trg_fix_playlist_collaborators



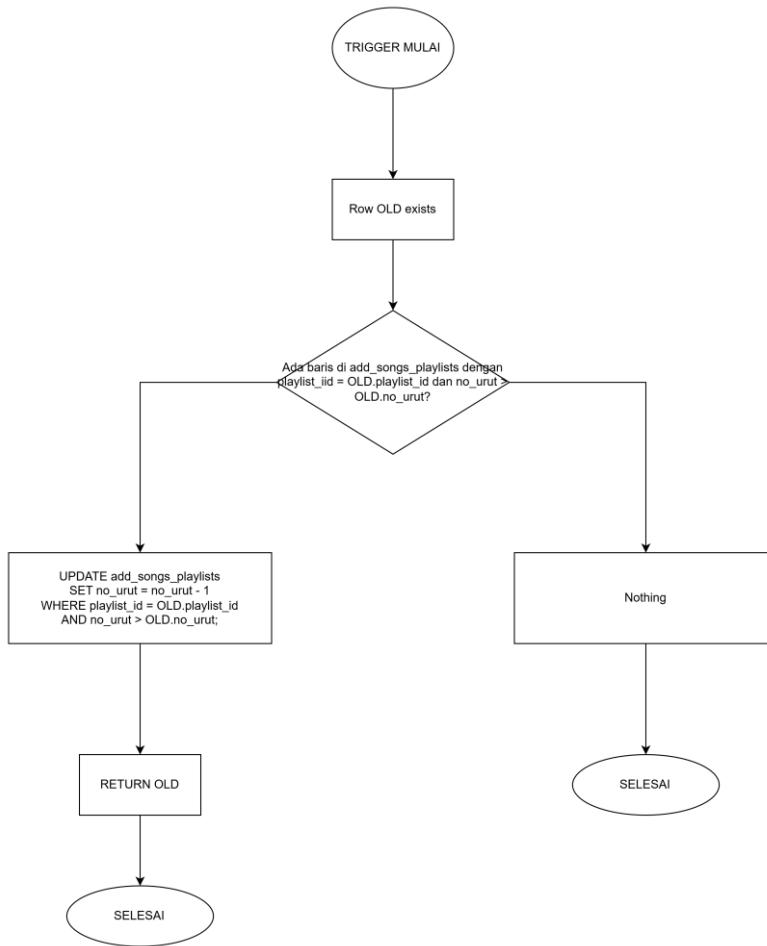
Gambar 4. 42 Flow Chart Trigger trg_update_artist_follow_count



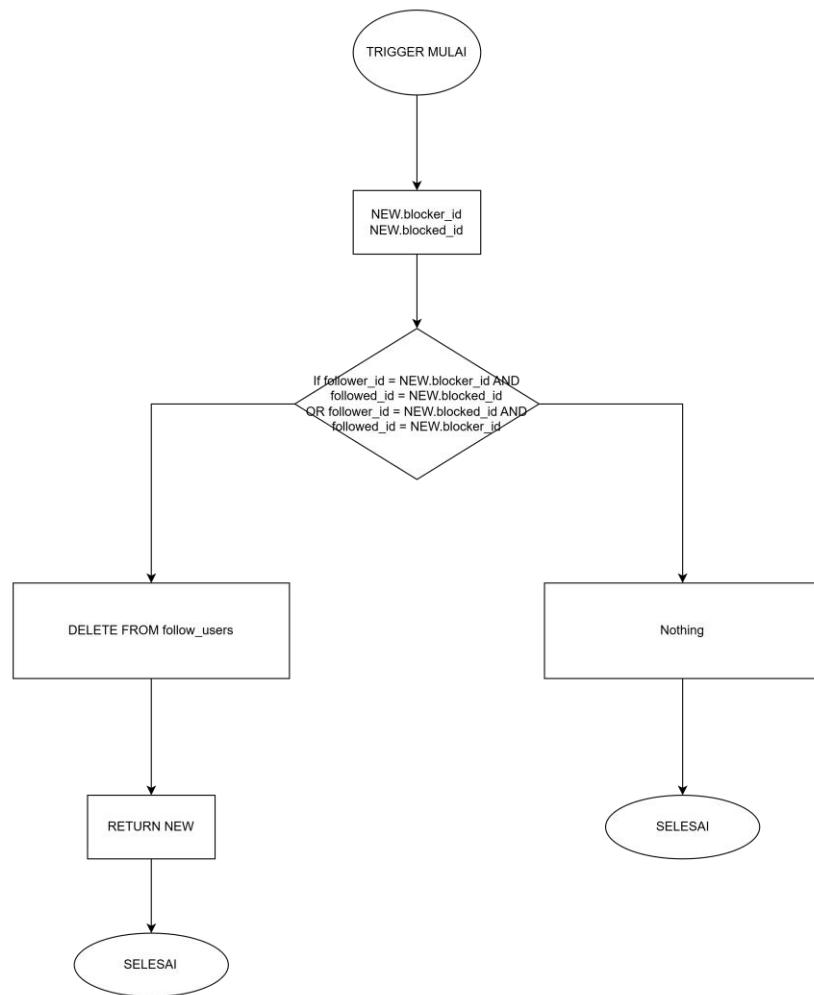
Gambar 4. 43 Flow Chart Trigger trg_update_collection_rating



Gambar 4. 44 Flow Chart Trigger trg_validate_tour_date



Gambar 4. 45 Flow Chart Trigger reorder_playlist_sequence



Gambar 4. 46 Flow Chart Trigger enforce_block_logic

4.7 View

Tabel 31 View Album Tracklist

Nama	VIEW_ALBUM_TRACKLIST	
Deskripsi	View ini menampilkan daftar lagu dalam sebuah collection beserta artis album, artis lagu, urutan track, dan durasi yang sudah diformat. Digunakan untuk menampilkan tracklist album secara lengkap tanpa duplikasi.	
No.	Nama Kolom	Tipe Data
1.	COLLECTION_ID	INT4
2.	COLLECTION_TITLE	VARCHAR(255)
3.	ALBUM_ARTISTS	TEXT
4.	NOMOR_DISC	INT4

5.	NOMOR_TRACK	INT4
6.	SONG_TITLE	VARCHAR(255)
7.	SONG_ARTISTS	TEXT
8.	DURATION	TEXT

Tabel 32 View Artist Header

Nama			VIEW_ARTIST_PROFILE_HEADER
Deskripsi		View ini ditujukan untuk memudahkan pengambilan data header artis termasuk data dasar, total album, total lagu, follower count, dan monthly listener.	
No.	Nama Kolom		Tipe Data
1.	ARTIST_ID		INT4
2.	ARTIST_NAME		VARCHAR(255)
3.	BIO		TEXT
4.	ARTIST_PFP		VARCHAR(2048)
5.	BANNER		VARCHAR(2048)
6.	ARTIST_EMAIL		VARCHAR(320)
7.	MONTHLY_LISTENER_COUNT		INT8
8.	FOLLOWER_COUNT		INT8
9.	TOTAL_ALBUMS		INT8
10.	TOTAL_TRACKS		INT8

Tabel 33 View Full Song Details

Nama			VIEW_FULL_SONG_DETAILS
Deskripsi		View ini ditujukan untuk memudahkan pengambilan data lagu termasuk collection-collection apa saja yang lagu terdapat didalamnya, sehingga terdapat duplikat lagu dengan asosiasi collection yang berbeda.	
No.	Nama Kolom		Tipe Data

1.	SONG_ID	INT4
2.	SONG_TITLE	VARCHAR(255)
3.	ARTISTS_NAME	TEXT
4.	ALBUM_NAME	VARCHAR(255)
5.	SONG_DURATION	INT4
6.	POPULARITY	NUMERIC(3,0)
7.	SONG_FILE	VARCHAR(320)
8.	DURATION_FORMATTED	TEXT

Tabel 34 View Top Charts

Nama		VIEW_TOP_CHARTS
Deskripsi		View ini ditujukan untuk memudahkan pengambilan data lagu-lagu diurutkan berdasarkan popularitas. Berbeda dengan VIEW_FULL_SONG_DETAILS, disini lagu tidak akan duplikat, supaya urutan sesuai.
No.	Nama Kolom	Tipe Data
1.	RANK	INT8
2.	SONG_TITLE	VARCHAR(255)
3.	ARTISTS	TEXT
4.	ALBUM	VARCHAR(255)
5.	POPULARITY	NUMERIC(3,0)

Tabel 35 View User Library Stats

Nama		VIEW_USER_LIBRARY_STATS
Deskripsi		View ini ditujukan untuk memudahkan pengambilan data user termasuk jumlah follower, jumlah following user dan artis, dan jumlah playlist public
No.	Nama Kolom	Tipe Data
1.	USER_ID	INT4

2.	USERNAME	VARCHAR(50)
3.	USER_PFP	VARCHAR(2048)
4.	PUBLIC_PLAYLIST	INT8
5.	FOLLOWING_COUNT	INT8
6.	FOLLOWERS_COUNT	INT8

4.8 Cron

Cron adalah mekanisme task scheduling yang memungkinkan sistem basis data menjalankan perintah secara otomatis pada waktu tertentu tanpa interaksi pengguna. Dalam sistem ini, cron digunakan melalui pg_cron, sebuah ekstensi PostgreSQL yang memungkinkan penjadwalan fungsi PL/pgSQL secara periodik (harian, mingguan, atau interval tertentu). Ekstensi ini tidak tersedia di Sistem Operasi Windows, tetapi tersedia di Sistem Operasi Linux, sehingga sangat cocok digunakan pada lingkungan server database yang umumnya berjalan pada sistem operasi Linux. Di Arch Linux, ada package di AUR (Arch User Repository) bernama ‘pg_cron’.

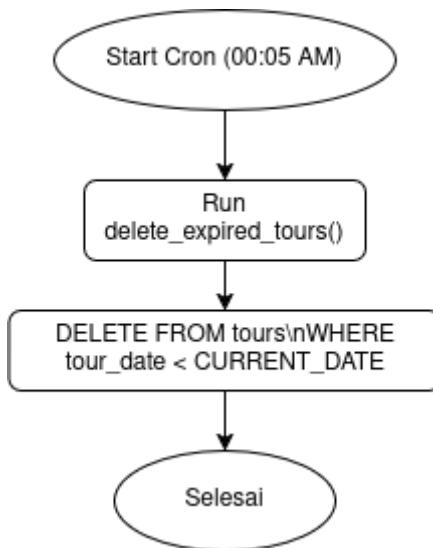
Secara default, pg_cron menggunakan timezone UTC, bukan timezone lokal server. Artinya, jika server berada di zona waktu Asia/Jakarta (UTC+7), maka job yang dijadwalkan pada pukul 08:00 pagi harus dikonversi menjadi 01:00 UTC.

Penggunaan cron penting untuk proses-proses yang bersifat time-driven, yaitu proses yang harus berjalan berdasarkan waktu, bukan berdasarkan event database seperti INSERT, UPDATE, atau DELETE. Oleh karena itu, cron tidak dapat digantikan oleh trigger.

Dalam perancangan sistem layanan musik ini, terdapat beberapa proses otomatis yang tidak dapat mengandalkan trigger, karena harus dijalankan secara berkala berdasarkan tanggal dan waktu tertentu. Contoh proses tersebut:

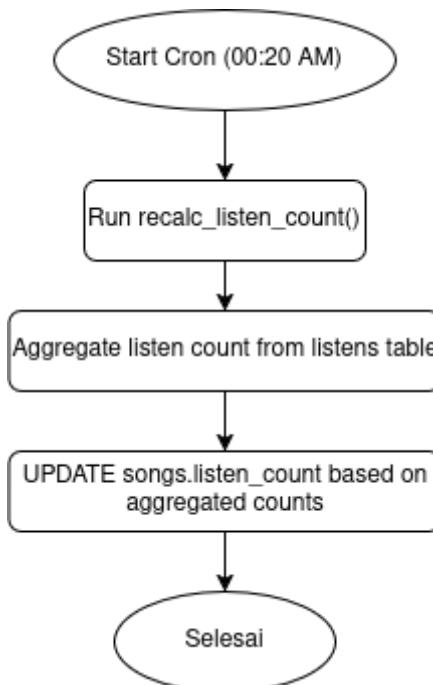
1. Menghapus tour yang sudah lewat tanggalnya.
2. Menonaktifkan status prerelease pada collection yang hari rilisnya sudah tiba.
3. Menghitung ulang metrik tertentu secara periodik seperti popularitas global, monthly listeners, atau aggregasi data lain yang tidak perlu dihitung real-time.

Cron memastikan data tetap konsisten dan bersih meskipun tidak ada aktivitas user terhadap tabel terkait.



Gambar 4. 47 Flow Chart Cron delete_expired_tours_daily

Cronjob ini dijalankan setiap hari pukul 00:05. Fungsinya adalah menghapus semua entry tour yang tanggalnya sudah lewat hari ini (`tour_date < CURRENT_DATE`). Karena trigger tidak dapat berjalan berdasarkan waktu, cronjob ini memastikan bahwa daftar tour tetap bersih dan hanya menampilkan tour yang masih relevan. Jika ada tour yang sudah lewat, sistem akan otomatis menghapusnya tanpa perlu intervensi pengguna.

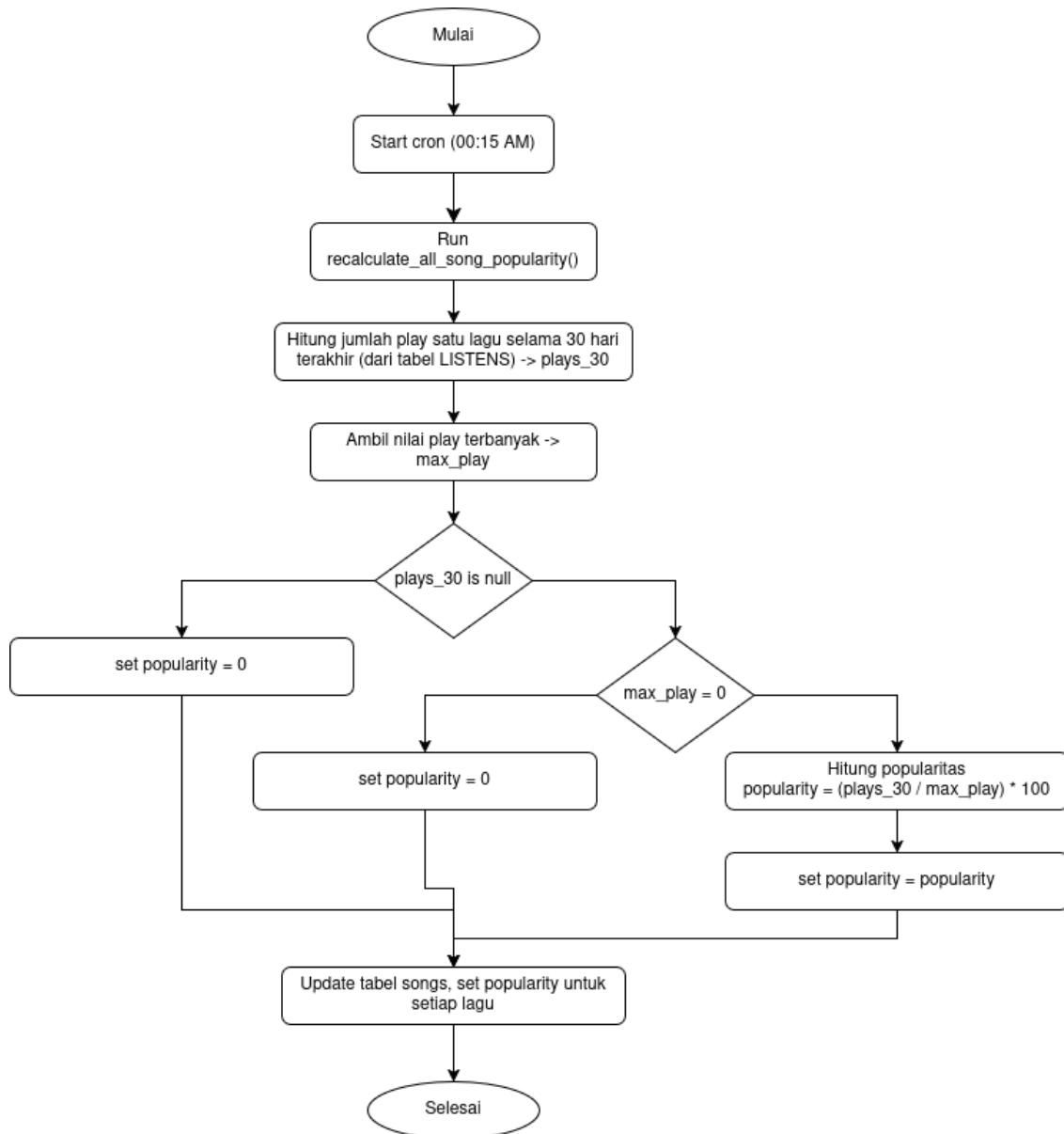


Gambar 4. 48 Flow Chart Cron recalc_listen_count_daily

Cronjob ini berjalan setiap hari pukul 00:20 untuk menghitung ulang total jumlah pemutaran (listen_count) setiap lagu berdasarkan tabel listens.

Pendekatan ini lebih efisien dibanding memperbarui listen_count melalui trigger setiap kali user memutar lagu, karena trigger akan memberikan beban tinggi pada transaksi insert.

Dengan cronjob, perhitungan dilakukan sekali sehari, tetap akurat, dan jauh lebih efisien.



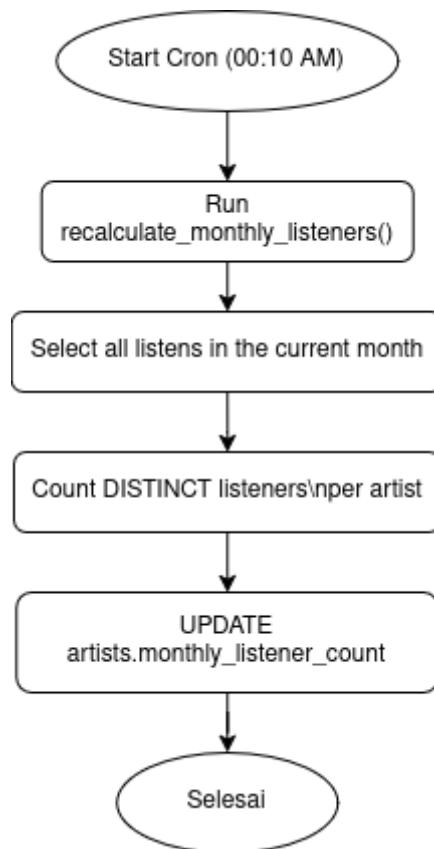
Gambar 4. 49 Flow Chart Cron recalculate_song_popularity_daily

Cronjob ini menghitung ulang popularitas seluruh lagu setiap hari pada pukul 00:15. Popularitas dihitung menggunakan formula berikut.

popularitas =

$$\text{popularitas} = \frac{\text{jumlah_play_30_hari_terakhir}}{\text{jumlah_play_lagu_terbanyak_di_sistem_30_hari_terakhir}} * 100$$

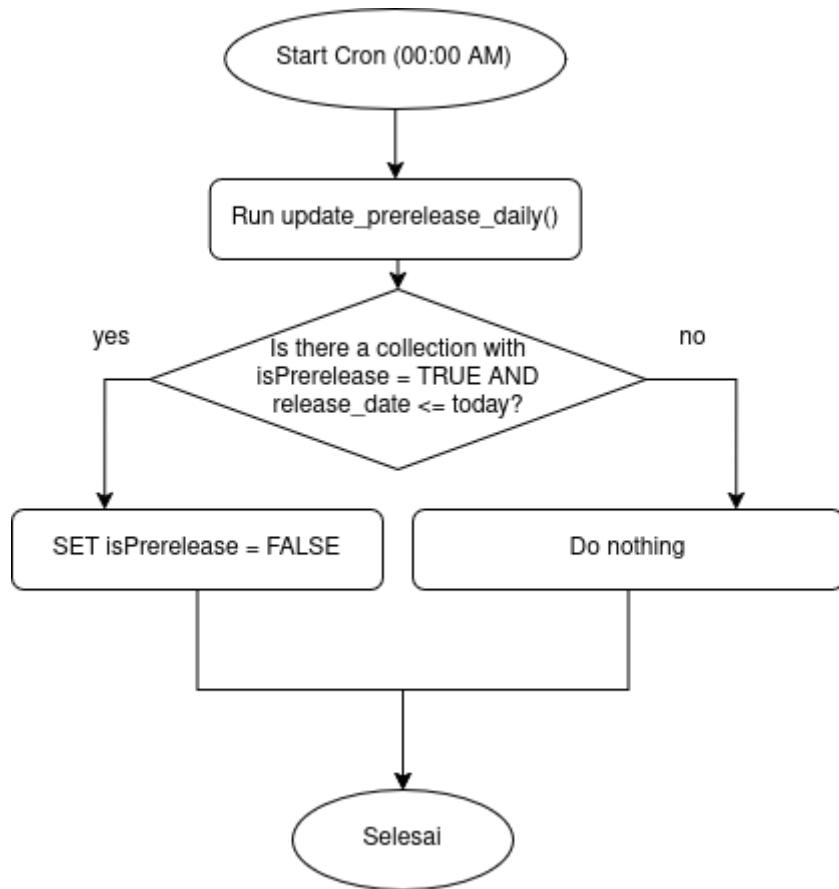
Cronjob ini menjaga agar popularitas bersifat dinamis, terbarui, dan stabil, tanpa membebani sistem saat pemutaran terjadi.



Gambar 4. 50 Flow Chart Cron recalculate_monthly_listeners_daily

Cronjob ini memperbarui jumlah monthly listeners setiap artis setiap hari pukul 00:10. Nilai dihitung berdasarkan jumlah user unik (DISTINCT user_id) yang memutar lagu artis dalam bulan berjalan.

Karena perhitungan ini memerlukan agregasi besar dan melibatkan beberapa tabel (listens dan create_songs), menjalankannya melalui trigger akan sangat tidak efisien. Cronjob menjadi solusi optimal untuk melakukan pembaruan terjadwal yang ringan bagi sistem.



Gambar 4. 51 Flow Chart Cron update_prerelease_job_daily

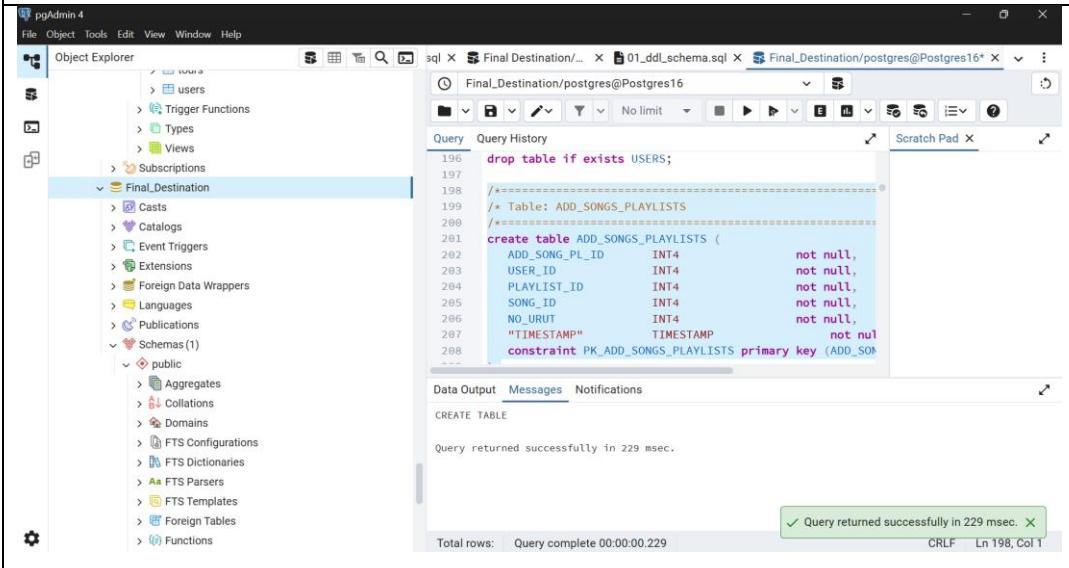
Cronjob ini dijalankan setiap hari tepat pukul 00:00, berfungsi untuk menonaktifkan status prerelease dari koleksi (album/EP/single) yang tanggal rilisnya telah tiba. Sistem secara otomatis mengecek apakah collection_release_date sudah sama atau lebih kecil dari CURRENT_DATE, kemudian mengubah isPrerelease menjadi FALSE.

Cronjob ini sangat penting untuk menghindari koleksi yang tetap tampil sebagai “belum dirilis” padahal sudah melewati hari rilisnya.

5. IMPLEMENTASI PENGOLAHAN DATA

5.1 DDL

Tabel 36 DDL – Tabel add_songs_playlist

Nama Tabel	add_songs_playlist
Deskripsi	Tabel ini berfungsi sebagai hubungan menambahkan lagu apa ke suatu playlist
Script SQL	
<pre>/*===== /* Table: ADD_SONGS_PLAYLISTS */ create table ADD_SONGS_PLAYLISTS (ADD_SONG_PL_ID INT4 not null, USER_ID INT4 not null, PLAYLIST_ID INT4 not null, SONG_ID INT4 not null, NO_URUT INT4 not null, "TIMESTAMP" DATE not null, constraint PK_ADD_SONGS_PLAYLISTS primary key (ADD_SONG_PL_ID));</pre>	
Screenshot Hasil	
 The screenshot shows the pgAdmin 4 interface. In the Object Explorer, the 'Final_Destination' schema is selected. In the main query editor window, the following SQL code is run: <pre>drop table if exists USERS; /* Table: ADD_SONGS_PLAYLISTS */ create table ADD_SONGS_PLAYLISTS (ADD_SONG_PL_ID INT4 not null, USER_ID INT4 not null, PLAYLIST_ID INT4 not null, SONG_ID INT4 not null, NO_URUT INT4 not null, "TIMESTAMP" DATE not null, constraint PK_ADD_SONGS_PLAYLISTS primary key (ADD_SONG_PL_ID));</pre> The 'Messages' tab shows the result: 'Query returned successfully in 229 msec.'.	

Tabel 37 DDL – Tabel Artists

Nama Tabel	Artists
Deskripsi	Tabel untuk menyimpan data artis, termasuk nama, foto profil, banner, email, jumlah pendengar, dan koleksi (album/EP) yang terkait.
Script SQL	<pre>/* * Table: ARTISTS */ create table ARTISTS (ARTIST_ID INT4 not null, ARTIST_NAME VARCHAR(255) not null, BIO TEXT null, MONTHLY_LISTENER_COUNT INT8 null default 0, ARTIST_PFP VARCHAR(2048) null, ARTIST_EMAIL VARCHAR(320) not null unique, BANNER VARCHAR(2048) null, FOLLOWER_COUNT INT8 null default 0, constraint PK_ARTISTS primary key (ARTIST_ID));</pre>

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface with the 'Object Explorer' on the left and the 'Query Editor' on the right. The 'Query Editor' displays the SQL code for creating the ARTISTS table:

```
/*
 * Table: ARTISTS
 */
create table ARTISTS (
    ARTIST_ID          INT4          not null,
    ARTIST_NAME        VARCHAR(255)   not null,
    BIO                TEXT          null,
    MONTHLY_LISTENER_COUNT INT8          null default 0,
    ARTIST_PFP         VARCHAR(2048)  null,
    ARTIST_EMAIL       VARCHAR(320)    not null unique,
    BANNER             VARCHAR(2048)  null,
    FOLLOWER_COUNT    INT8          null default 0,
    constraint PK_ARTISTS primary key (ARTIST_ID)
);
```

The status bar at the bottom indicates 'Query returned successfully in 97 rows.'

Tabel 38 DDL – Tabel Artists_tours

Nama Tabel	Artists_tours
Deskripsi	Tabel ini menyimpan hubungan antara tour dan artists
Script SQL	
<pre>/*===== /* Table: ARTISTS_TOURS /*===== create table ARTISTS_TOURS (TOUR_ID INT4 not null, ARTIST_ID INT4 not null, constraint PK_ARTISTS_TOURS primary key (TOUR_ID, ARTIST_ID));</pre>	

Tabel 39 DDL – Tabel Artist_Promotion

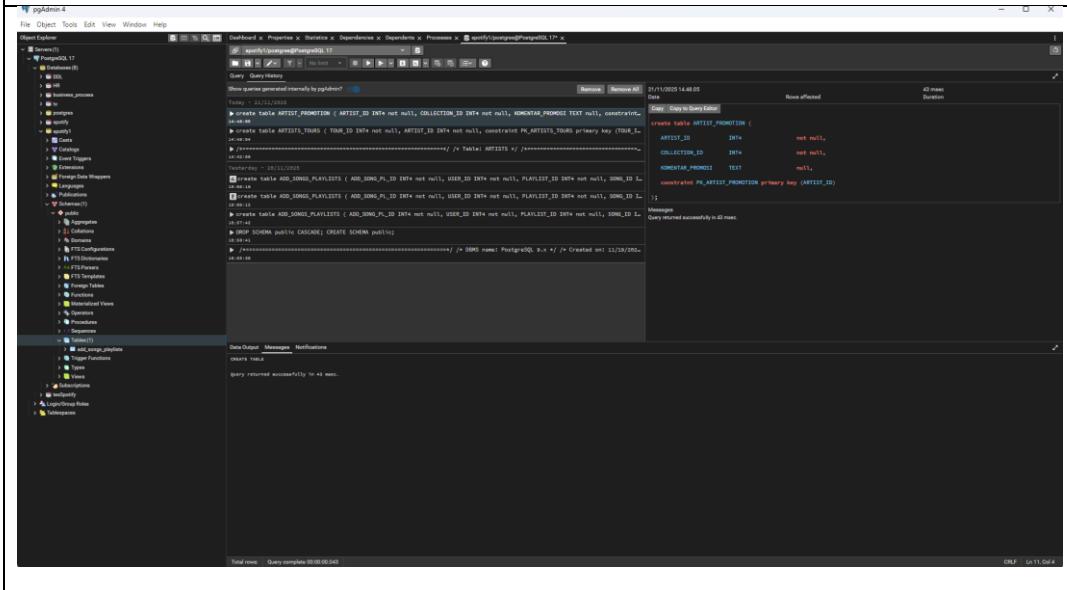
Nama Tabel	Artist_promotion
Deskripsi	Tabel ini menyimpan text promosi dan hubungan tabel artist dan collection
Script SQL	
/*=====*/	

```

/* Table: ARTIST_PROMOTION
*=====
create table ARTIST_PROMOTION (
    ARTIST_ID          INT4          not null,
    COLLECTION_ID      INT4          not null,
    KOMENTAR_PROMOSI   TEXT          null,
    constraint PK_ARTIST_PROMOTION primary key (ARTIST_ID)
);

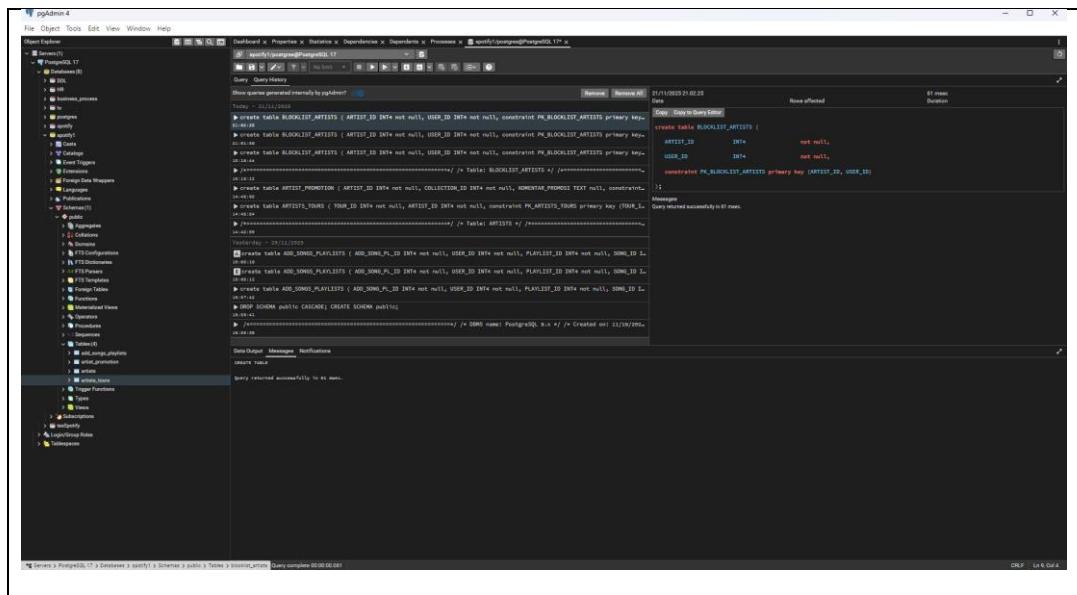
```

Screenshot Hasil



Tabel 40 DDL – Tabel Blocklist_artists

Nama Tabel	Blocklist_artists
Deskripsi	Tabel ini menyimpan hubungan antara artists dan user, untuk siapa saja artist yang di block oleh user
Script SQL	<pre> /*===== /* Table: BLOCKLIST_ARTISTS *===== create table BLOCKLIST_ARTISTS (ARTIST_ID INT4 not null, USER_ID INT4 not null, constraint PK_BLOCKLIST_ARTISTS primary key (ARTIST_ID, USER_ID)); </pre>
Screenshot Hasil	



Tabel 41 DDL – Tabel block_users

Nama Tabel	Block_users
Deskripsi	Tabel ini berfungsi untuk menyimpan data user yang diblock dan yang terblock
Script SQL	<pre>/* * Table: BLOCK_USERS */ create table BLOCK_USERS (BLOCKER_ID INT4 not null, BLOCKED_ID INT4 not null, constraint PK_BLOCK_USERS primary key (BLOCKER_ID, BLOCKED_ID));</pre>
Screenshot Hasil	

The screenshot shows the MySQL Workbench interface with the following details:

- File**, **Edit**, **Tools**, **View**, **Window**, **Help** menu.
- Server Explorer** pane on the left with a tree view of the database structure:
 - MySQL (localhost)
 - Portfolios_17
 - Tables (0)
 - Views (0)
 - Procedures (0)
 - Functions (0)
 - Triggers (0)
 - Types (0)
 - Virtual Tables (0)
 - Materialized Views (0)
 - Operations (0)
 - Procedures (0)
 - Functions (0)
 - Triggers (0)
 - Virtual Tables (0)
 - Materialized Views (0)
 - Subscriptions (0)
 - Replicability (0)
 - Tablespaces (0)
- Databases** pane at the top center showing the selected database: mysql-coverage@Portfolios_17.
- Query** pane at the bottom containing the following SQL code:

```
create table BLOCK_USERS (BLOCKER_ID INT not null, BLOCKED_ID INT not null, constraint PK_BLOCK_USERS primary key (BLOCKER_ID, BLOCKED_ID));
create table BLOCKLIST_ARTISTS (ARTIST_ID INT not null, USER_ID INT not null, constraint PK_BLOCKLIST_ARTISTS primary key (ARTIST_ID, USER_ID));
create table BLOCKLIST_ARTISTS (ARTIST_ID INT not null, USER_ID INT not null, constraint PK_BLOCKLIST_ARTISTS primary key (ARTIST_ID, USER_ID));
create table BLOCKLIST_ARTISTS (ARTIST_ID INT not null, USER_ID INT not null, constraint PK_BLOCKLIST_ARTISTS primary key (ARTIST_ID, USER_ID));
create table ARTIST_PROMOTION (ARTIST_ID INT not null, COLLECTION_ID INT not null, constraint PK_ARTIST_PROMOTION primary key (ARTIST_ID, COLLECTION_ID));
create table ARTISTS_TRACKS (TRACK_ID INT not null, ARTIST_ID INT not null, constraint PK_ARTISTS_TRACKS primary key (TRACK_ID, ARTIST_ID));
create table ARTISTS_TRACKS (TRACK_ID INT not null, ARTIST_ID INT not null, constraint PK_ARTISTS_TRACKS primary key (TRACK_ID, ARTIST_ID));
create table ARTISTS_TRACKS (TRACK_ID INT not null, ARTIST_ID INT not null, constraint PK_ARTISTS_TRACKS primary key (TRACK_ID, ARTIST_ID));
create table ADD_SONGS_PLAYLISTS (ADD_SONG_PL_ID INT not null, USER_ID INT not null, PLAYLIST_ID INT not null, SONG_ID INT not null);
create table ADD_SONGS_PLAYLISTS (ADD_SONG_PL_ID INT not null, USER_ID INT not null, PLAYLIST_ID INT not null, SONG_ID INT not null);
create table ADD_SONGS_PLAYLISTS (ADD_SONG_PL_ID INT not null, USER_ID INT not null, PLAYLIST_ID INT not null, SONG_ID INT not null);
create table SCHEMA public.CASCABEL CREATE SCHEMA public;
```
- Logs** pane at the bottom right showing a log entry: "Query returned successfully in 0 rows."

Tabel 42 DDL – Tabel collections

Nama Tabel	collections
Deskripsi	Tabel ini berfungsi untuk menyimpan collections entah itu album
Script SQL	
<pre>/*===== /* Table: COLLECTIONS /*=====*/ create table COLLECTIONS (COLLECTION_ID INT4 not null, COLLECTION_TITLE VARCHAR(255) not null, COLLECTION_TYPE VARCHAR(50) not null, COLLCETION_COVER VARCHAR(2048) null, COLLECTION_RELEASE_DATE DATE not null, COLLECTION_RATING NUMERIC(3,0) null, ISPRERELEASE BOOL null, constraint PK_COLLECTIONS primary key (COLLECTION_ID));</pre>	

The screenshot shows the pgAdmin 4 interface with a database connection to "localhost:5432" named "public". The main area displays a query history with the following content:

```
2019-02-21 10:43:25
CREATE TABLE COLLECTIONS ( COLLECTION_ID INT4 NOT NULL, COLLECTION_TITLE VARCHAR(255) NOT NULL, COLLECTION_TYPE VARCHAR(50) NOT NULL );
CREATE TABLE BLOCK_USERS ( BLOCK_ID INT4 NOT NULL, BLOCKED_ID INT4 NOT NULL, constraint PK_BLOCK_USERS primary key (BLOCK_ID, BLOCKED_ID));
CREATE TABLE BLOCKLIST_ARTISTS ( ARTIST_ID INT4 NOT NULL, USER_ID INT4 NOT NULL, constraint PK_BLOCKLIST_ARTISTS primary key (ARTIST_ID, USER_ID));
CREATE TABLE ARTISTS_PROMOTION ( ARTIST_ID INT4 NOT NULL, PROMOTER_ID INT4 NOT NULL, PROMOTER_PRIMACY TEXT NOT NULL, constraint PK_ARTISTS_PROMOTION primary key (ARTIST_ID, PROMOTER_ID));
CREATE TABLE ARTISTS_TRACKS ( TERM_ID INT4 NOT NULL, ARTIST_ID INT4 NOT NULL, constraint PK_ARTISTS_TRACKS primary key (TERM_ID, ARTIST_ID));
CREATE TABLE ARTISTS ( ARTIST_ID INT4 NOT NULL, ARTIST_NAME VARCHAR(255) NOT NULL, ARTIST_BIO TEXT NOT NULL, constraint PK_ARTISTS primary key (ARTIST_ID));
CREATE TABLE ARTISTS_HOMES ( HOME_ID INT4 NOT NULL, ARTIST_ID INT4 NOT NULL, constraint PK_ARTISTS_HOMES primary key (HOME_ID, ARTIST_ID));
CREATE TABLE ADD_SONG_PLAYLISTS ( ADD_SONG_PL_ID INT4 NOT NULL, USER_ID INT4 NOT NULL, PLAYLIST_ID INT4 NOT NULL, SONG_ID INT4 NOT NULL );
CREATE TABLE ADD_SONG_PLAYLISTS_T ( ADD_SONG_PL_ID INT4 NOT NULL, USER_ID INT4 NOT NULL, PLAYLIST_ID INT4 NOT NULL, SONG_ID INT4 NOT NULL );
CREATE TABLE ADD_SONG_PLAYLISTS_U ( ADD_SONG_PL_ID INT4 NOT NULL, USER_ID INT4 NOT NULL, PLAYLIST_ID INT4 NOT NULL, SONG_ID INT4 NOT NULL );
DROP SCHEMA public CASCADE; CREATE SCHEMA public;
-- Schema Management

Query returned successfully in 43 rows.
```

The sidebar on the left shows the database schema with the following objects:

- Schemas (1)
- Tables (17)
 - Artist
 - Blocklist
 - BlockUser
 - Collection
 - Home
 - Playlist
 - Song
 - User
- Triggers (1)
- Functions (1)
- Procedures (1)
- Sequences (1)
- Views (1)
- Types (1)
- Subscriptions (1)
- Checkpoints (1)

Tabel 43 DDL – Tabel Collections_Songs

Nama Tabel	Collections_songs
Deskripsi	Tabel ini berfungsi untuk menyimpan hubungan antara songs dan collections, beserta dengan nomor disc dan nomor track
Script SQL	
<pre>/*===== /* Table: COLLECTIONS_SONGS /*===== create table COLLECTIONS_SONGS (SONG_ID INT4 not null, COLLECTION_ID INT4 not null, NOMOR_DISC INT4 not null, NOMOR_TRACK INT4 not null, constraint PK_COLLECTIONS_SONGS primary key (SONG_ID, COLLECTION_ID));</pre>	

The screenshot shows the pgAdmin 4 application window. The left sidebar is the Schema Browser, displaying the structure of the 'public' schema. The main pane shows the results of a SQL query:

```
create table COLLECTING_SONGS (SONG_ID INT4 not null, COLLECTION_ID INT4 not null, HONOR_TRACK INT4 not null);
create table COLLECTIONS (COLLECTION_ID INT4 not null, COLLECTION_TITLE VARCHAR(255) not null, COLLECTION_TYPE VARCHAR(255) not null);
create table BLOCK_USERS (BLOCKER_ID INT4 not null, BLOCKED_ID INT4 not null, constraint PK_BLOCK_USERS primary key (BLOCKER_ID, BLOCKED_ID));
create table BLOCKLIST_ARTISTS (ARTIST_ID INT4 not null, USER_ID INT4 not null, constraint PK_BLOCKLIST_ARTISTS primary key (ARTIST_ID, USER_ID));
create table BLOCKLIST_ARTISTS (ARTIST_ID INT4 not null, USER_ID INT4 not null, constraint PK_BLOCKLIST_ARTISTS primary key (ARTIST_ID, USER_ID));
create table ARTIST_PROMOTION (ARTIST_ID INT4 not null, COLLECTION_ID INT4 not null, NONMONOTONIC_PROMO TEXT not null, constraint PK_ARTIST_PROMOTION primary key (ARTIST_ID, COLLECTION_ID));
create table ARTISTS_TOURS (TOUR_ID INT4 not null, ARTIST_ID INT4 not null, constraint PK_ARTISTS_TOURS primary key (TOUR_ID, ARTIST_ID));
create table ARTISTS (ARTIST_ID INT4 not null);
create table AD_SONG_PLAYLISTS (AD_SONG_PL_ID INT4 not null, USER_ID INT4 not null, PLAYLIST_ID INT4 not null, SONG_ID INT4 not null);
create table AD_SONG_PLAYLISTS (AD_SONG_PL_ID INT4 not null, USER_ID INT4 not null, PLAYLIST_ID INT4 not null, SONG_ID INT4 not null);
create table AD_SONG_PLAYLISTS (AD_SONG_PL_ID INT4 not null, USER_ID INT4 not null, PLAYLIST_ID INT4 not null, SOME_ID ...);
create table AD_SONG_PLAYLISTS (AD_SONG_PL_ID INT4 not null, USER_ID INT4 not null, PLAYLIST_ID INT4 not null, SOME_ID ...);
CREATE TABLE
```

Query returned successfully in 32 ms.

Tabel 44 DDL – Tabel Collection_library

Nama Tabel	Collection_library
Deskripsi	Tabel collection library berfungsi sebagai hubungan antara users dan collections, collections apa saja yang disimpan users
Script SQL	
<pre>/*===== /* Table: COLLECTION_LIBRARY /*===== create table COLLECTION_LIBRARY (USER_ID INT4 not null, COLLECTION_ID INT4 not null, constraint PK_COLLECTION_LIBRARY primary key (USER_ID, COLLECTION_ID));</pre>	

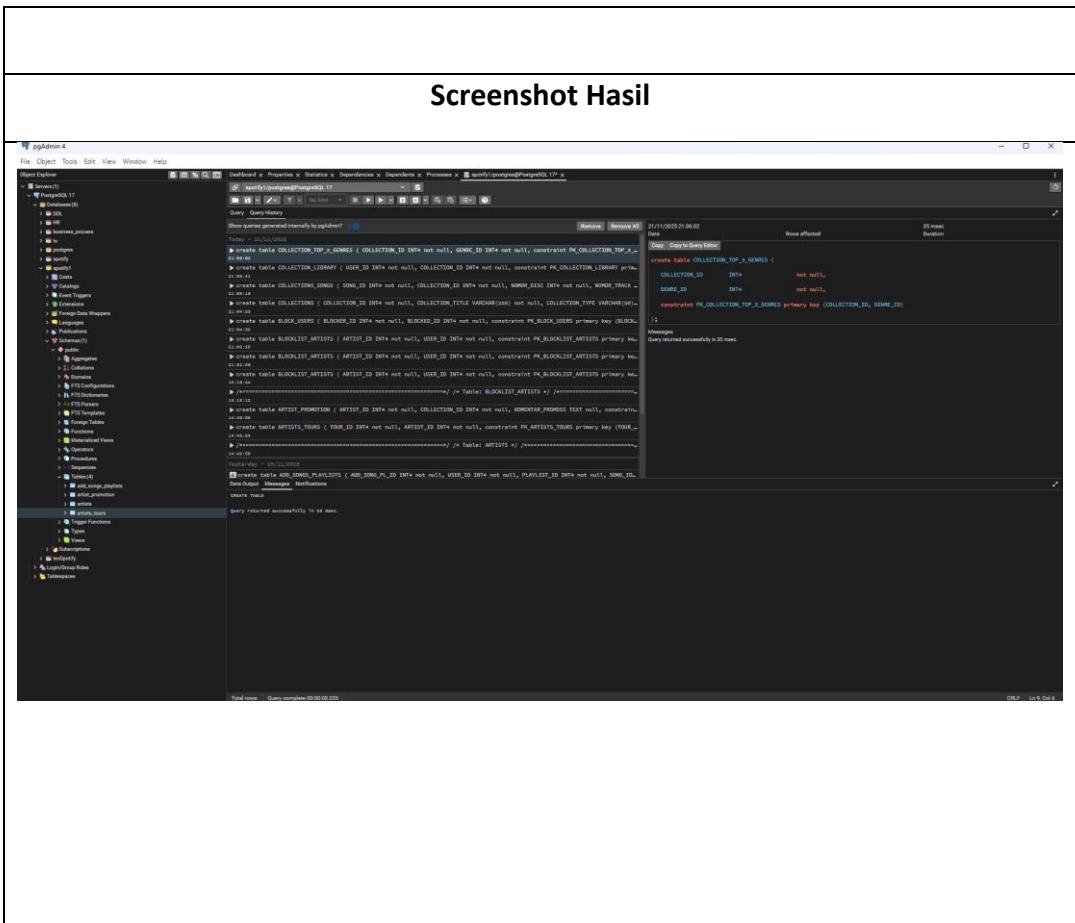
```

pgAdmin4
File Object Tools Edit View Window Help
Database > PostgreSQL > TestDB > Schemas > public > pgAdmin4.sql
File > Open > pgAdmin4.sql
Query > Query History
Show queries generated internally by pgAdmin? [ ] Remove Remove All
Date: 01/11/2020
Data 01/11/2020 21:05:41 Rows affected 36 rows Duration 36 msec
Copy > Copy to Query Editor
create table COLLECTION_LIBRARY (
    USER_ID INT4 not null,
    COLLECTION_ID INT4 not null,
    constraint PK_COLLECTION_LIBRARY primary key (USER_ID, COLLECTION_ID)
);
create table SONG (
    SONG_ID INT4 not null,
    COLLECTION_ID INT4 not null,
    NUMBER_DESC INT4 not null,
    NUMBER_TRACK INT4 not null
);
create table COLLECTION_SONGS (
    COLLECTION_ID INT4 not null,
    SONG_ID INT4 not null,
    constraint PK_COLLECTION_SONGS primary key (COLLECTION_ID, SONG_ID)
);
create table BLOCK (
    BLOCK_ID INT4 not null,
    COLLECTION_ID INT4 not null,
    COLLECTION_TYPE VARCHAR(100),
    constraint PK_BLOCK primary key (BLOCK_ID)
);
create table BLOCK_USERS (
    BLOCK_ID INT4 not null,
    USER_ID INT4 not null,
    constraint PK_BLOCK_USER primary key (BLOCK_ID, USER_ID)
);
create table BLOCKLIST_ARTISTS (
    ARTIST_ID INT4 not null,
    USER_ID INT4 not null,
    constraint PK_BLOCKLIST_ARTISTS primary key (ARTIST_ID, USER_ID)
);
create table BLOCKLIST_ARTISTS (
    ARTIST_ID INT4 not null,
    USER_ID INT4 not null,
    constraint PK_BLOCKLIST_ARTISTS primary key (ARTIST_ID, USER_ID)
);
create table ARTIST_PROMOTION (
    ARTIST_ID INT4 not null,
    COLLECTION_ID INT4 not null,
    HOMEPAGE_PROMISE TEXT null,
    constraint PK_ARTIST_PROMOTION primary key (ARTIST_ID, COLLECTION_ID)
);
create table ARTISTS (
    ARTIST_ID INT4 not null,
    TOUR_ID INT4 not null,
    constraint PK_ARTISTS primary key (ARTIST_ID, TOUR_ID)
);
create table TOURS (
    TOUR_ID INT4 not null,
    ARTIST_ID INT4 not null,
    constraint PK_ARTISTS_TOURS primary key (TOUR_ID)
);
create table ADD_SONG_PLISTS (
    ADD_SONG_PL_ID INT4 not null,
    USER_ID INT4 not null,
    PLAYLIST_ID INT4 not null,
    SONG_ID INT4 not null
);
create table ADD_SONG_PLISTS (
    ADD_SONG_PL_ID INT4 not null,
    USER_ID INT4 not null,
    PLAYLIST_ID INT4 not null,
    SONG_ID INT4 not null
);

```

Tabel 45 DDL – Tabel collection_top_3_genres

Nama Tabel	Collection_top_3_genres
Deskripsi	Tabel ini berfungsi menyimpan 3 genre utama dari suatu collection
Script SQL	
<pre> /* * Table: COLLECTION_TOP_3_GENRES */ create table COLLECTION_TOP_3_GENRES (COLLECTION_ID INT4 not null, GENRE_ID INT4 not null, constraint PK_COLLECTION_TOP_3_GENRES primary key (COLLECTION_ID, GENRE_ID)); </pre>	



Tabel 46 DDL – Tabel create_songs

Nama Tabel	Create_songs
Deskripsi	Tabel ini berfungsi pembuatan songs, hubungan antara songs dan artists
Script SQL	
<pre>/* * Table: CREATE_SONGS */ create table CREATE_SONGS (SONG_ID INT4 not null, ARTIST_ID INT4 not null, constraint PK_CREATE_SONGS primary key (SONG_ID, ARTIST_ID));</pre>	

Screenshot Hasil

The screenshot shows the pgAdmin 4 application window. The title bar reads "pgAdmin 4". The left sidebar displays the database structure under "PostgreSQL 17", including Schemas, Tables, Views, Functions, Triggers, and Sequences. A specific table named "CREATE_SONGS" is selected. The main pane shows a query history with several recent SQL statements, all of which have been successfully executed. The bottom status bar indicates "Total rows: 0" and "Query completed 00:00:00.259".

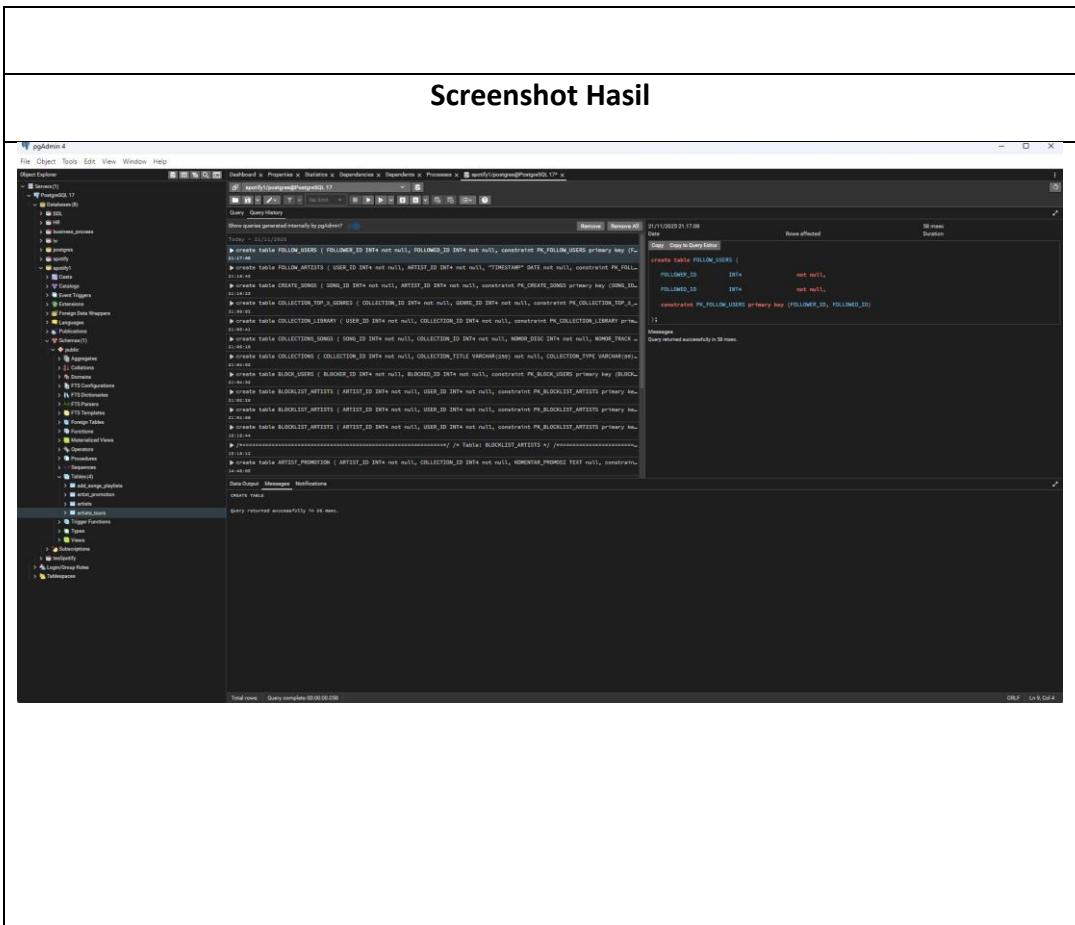
Tabel 47 DDL – Tabel follow_artists

Nama Tabel	Follow_artists
Deskripsi	Tabel ini berfungsi sebagai hubungan antara users dan artists yang difollow users
Script SQL	
<pre>/*===== /* Table: FOLLOW_ARTISTS /*===== create table FOLLOW_ARTISTS (USER_ID INT4 not null, ARTIST_ID INT4 not null, "TIMESTAMP" DATE not null, constraint PK_FOLLOW_ARTISTS primary key (USER_ID, ARTIST_ID));</pre>	

Screenshot Hasil

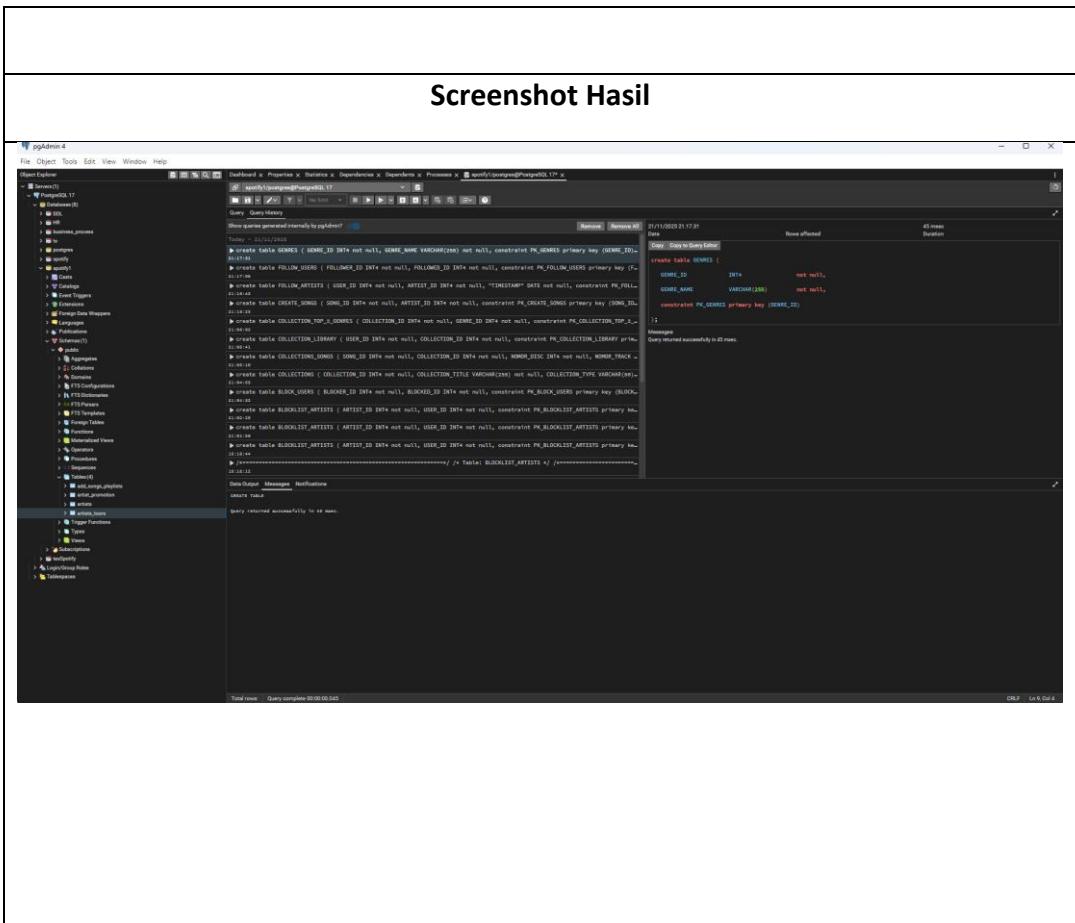
Tabel 48 DDL – Tabel follow_users

Nama Tabel	Follow_users
Deskripsi	Tabel ini sebagai hubungan antara users dan users lainnya, follow
Script SQL	
<pre>/*===== /* Table: FOLLOW_USERS /*===== create table FOLLOW_USERS (FOLLOWER_ID INT4 not null, FOLLOWED_ID INT4 not null, constraint PK_FOLLOW_USERS primary key (FOLLOWER_ID, FOLLOWED_ID));</pre>	



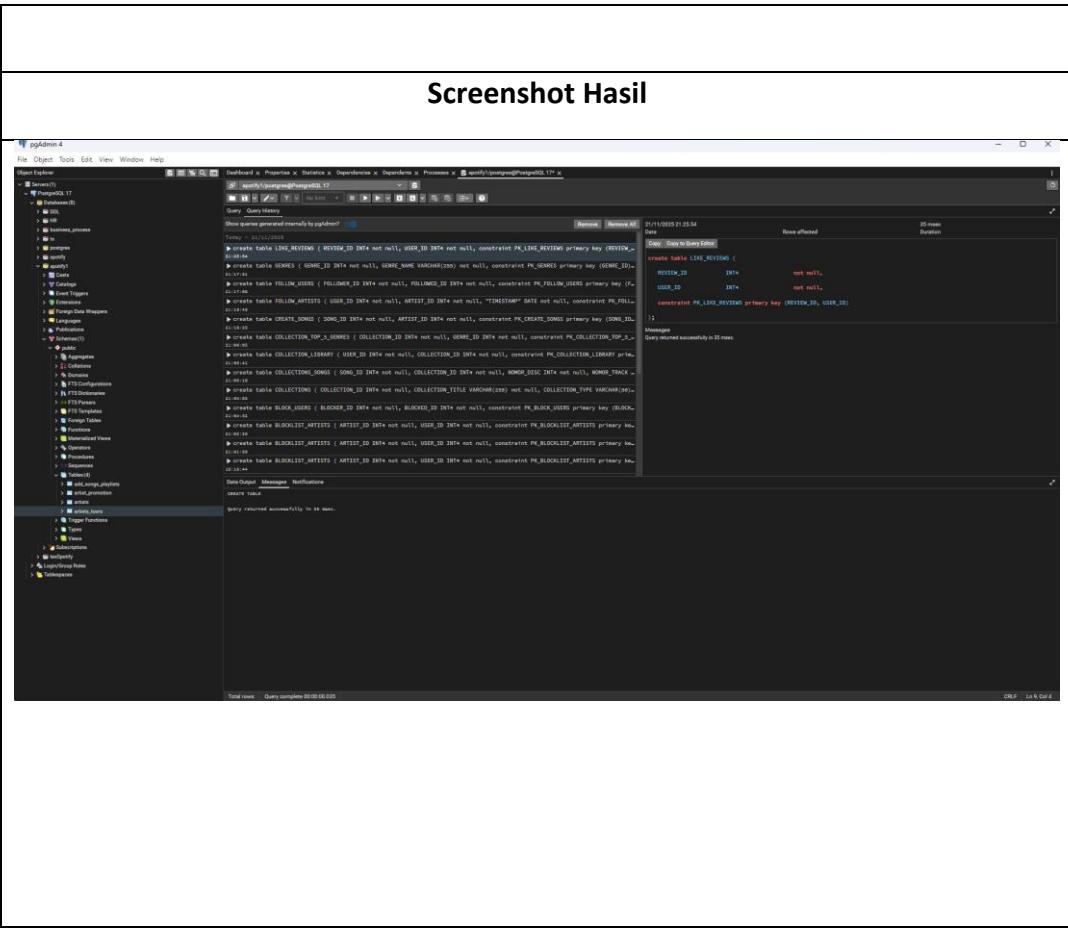
Tabel 49 DDL – Tabel genres

Nama Tabel	genres
Deskripsi	Tabel ini berfungsi menyimpan genres
Script SQL	
<pre>/* * Table: GENRES */ create table GENRES (GENRE_ID INT4 not null, GENRE_NAME VARCHAR(255) not null, constraint PK_GENRES primary key (GENRE_ID));</pre>	



Tabel 50 DDL – Tabel like_reviews

Nama Tabel	Like_reviews
Deskripsi	Tabel ini bergungsi sebagai hubungan antara reviews dan userss
Script SQL	
<pre>/* * Table: LIKE_REVIEWS */ create table LIKE_REVIEWS (REVIEW_ID INT4 not null, USER_ID INT4 not null, constraint PK_LIKE_REVIEWS primary key (REVIEW_ID, USER_ID));</pre>	



Tabel 51 DDL – Tabel like_songs

Nama Tabel	Like_songs
Deskripsi	Tabel like_songs berfungsi sebagai hubungan antara users dan songs yang di-like
Script SQL	
<pre>/*===== /* Table: LIKE_SONGS /*===== create table LIKE_SONGS (SONG_ID INT4 not null, USER_ID INT4 not null, "TIMESTAMP" DATE not null, constraint PK_LIKE_SONGS primary key (SONG_ID, USER_ID));</pre>	

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface with the SQL tab selected. The query history window displays the following DDL code:

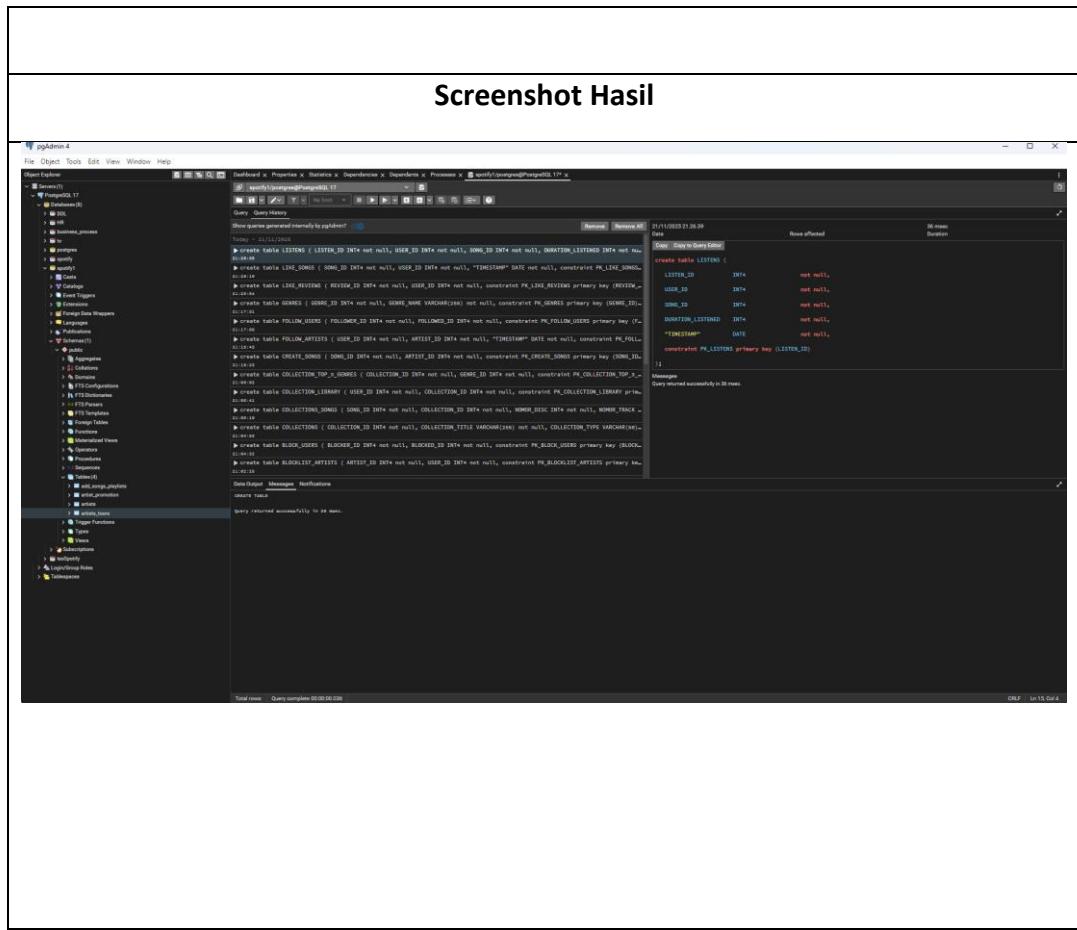
```

create table LISTENS (
    LISTEN_ID          INT4          not null,
    USER_ID            INT4          not null,
    SONG_ID             INT4          not null,
    DURATION_LISTENED INT4          not null,
    "TIMESTAMP"        DATE         not null,
    constraint PK_LISTENS primary key (LISTEN_ID)
);

```

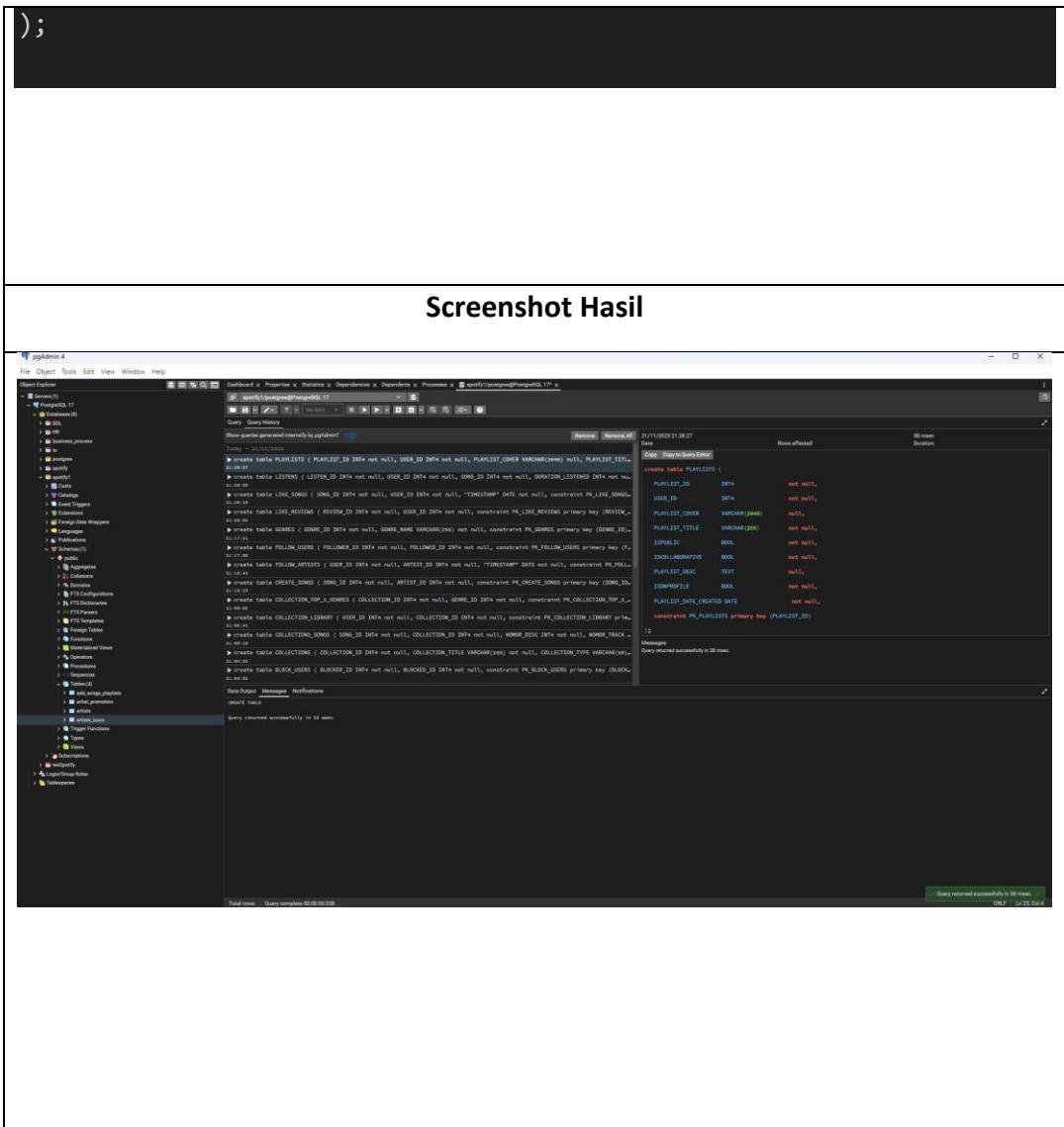
Tabel 52 DDL – Tabel listens

Nama Tabel	listens
Deskripsi	Tabel listens berfungsi sebagai penyimpanan lagu mana saja yang telah didengarkan users
Script SQL	
<pre>/* * Table: LISTENS */ create table LISTENS (LISTEN_ID INT4 not null, USER_ID INT4 not null, SONG_ID INT4 not null, DURATION_LISTENED INT4 not null, "TIMESTAMP" DATE not null, constraint PK_LISTENS primary key (LISTEN_ID));</pre>	



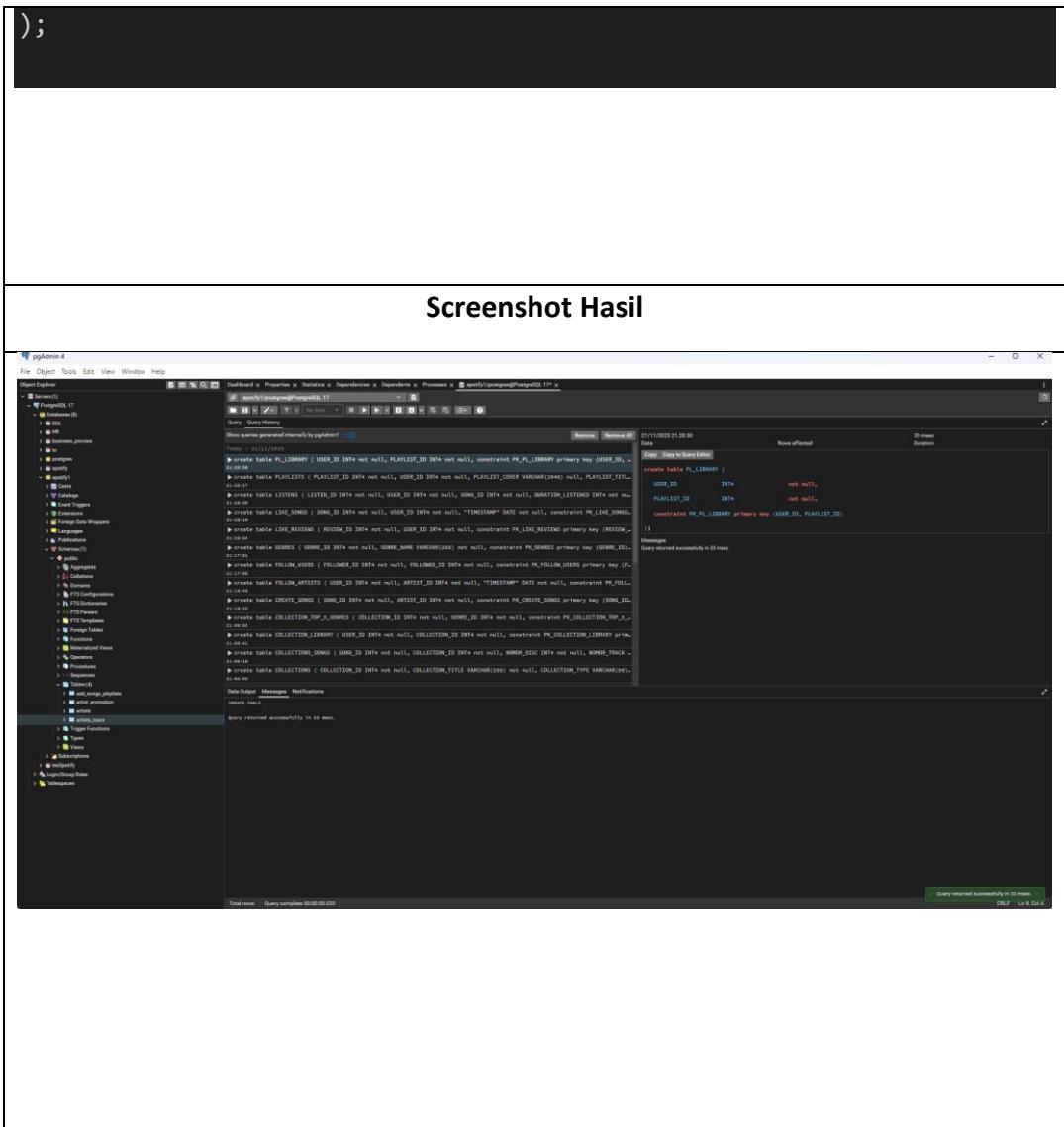
Tabel 53 DDL – Tabel playlists

Nama Tabel	playlists
Deskripsi	Tabel ini berfungsi sebagai penyimpanan sejumlah songs
Script SQL	
<pre>/* * Table: PLAYLISTS */ create table PLAYLISTS (PLAYLIST_ID INT4 not null, USER_ID INT4 not null, PLAYLIST_COVER VARCHAR(2048) null, PLAYLIST_TITLE VARCHAR(255) not null, ISPUBLIC BOOL not null, ISCOLLABORATIVE BOOL not null, PLAYLIST_DESC TEXT null, ISONPROFILE BOOL not null, PLAYLIST_DATE_CREATED DATE not null, constraint PK_PLAYLISTS primary key (PLAYLIST_ID)</pre>	



Tabel 54 DDL – Tabel PI_library

Nama Table	PI_library
Deskripsi	Tabel pl_library berfungsi sebagai penyimpanan hubungan antara playlists yang dimiliki users
Script SQL	
<pre>/* * Table: PL_LIBRARY */ create table PL_LIBRARY (USER_ID INT4 not null, PLAYLIST_ID INT4 not null, constraint PK_PL_LIBRARY primary key (USER_ID, PLAYLIST_ID)</pre>	

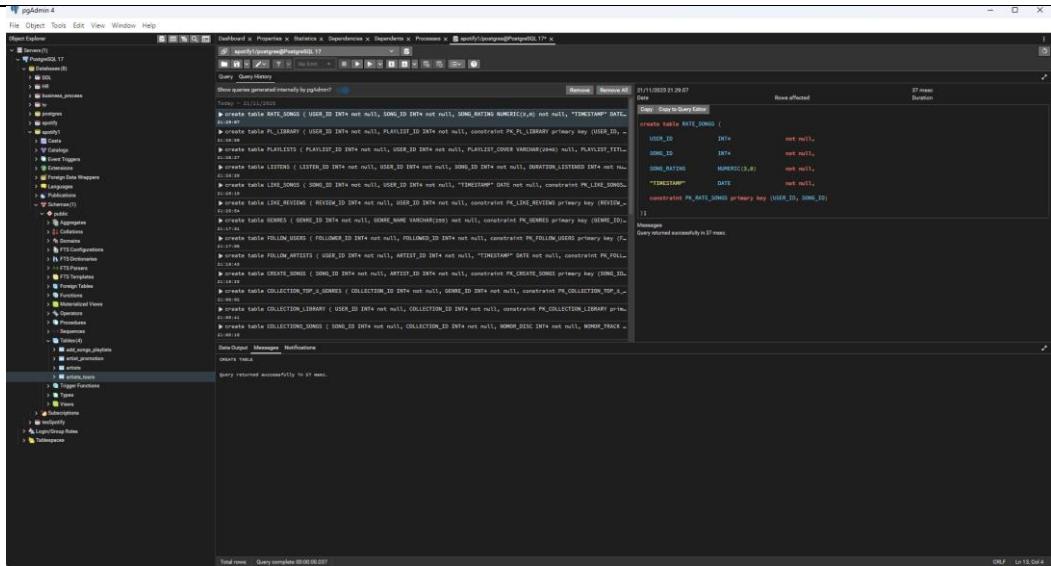


Tabel 55 DDL – Tabel Rate_songs

Nama Tabel	Rate_songs
Deskripsi	Tabel rate_songs berfungsi sebagai penyimpanan hubungan antara songs yang di-rate oleh users
Script SQL	
<pre> /* * Table: RATE_SONGS */ create table RATE_SONGS (USER_ID INT4 not null, SONG_ID INT4 not null, SONG_RATING NUMERIC(3,0) not null, </pre>	

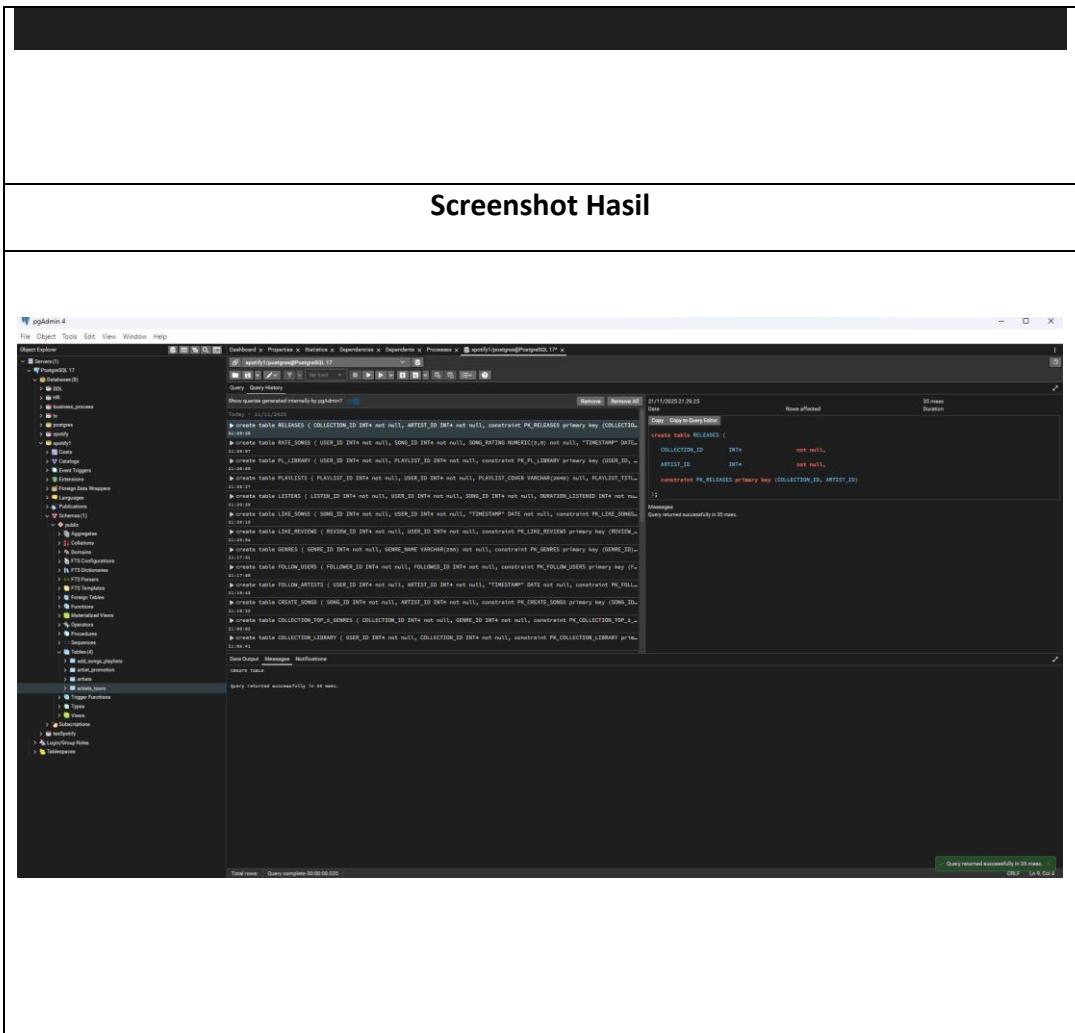
```
"TIMESTAMP"           DATE           not null,  
constraint PK_RATE_SONGS primary key (USER_ID, SONG_ID)  
);
```

Screenshot Hasil



Tabel 56 DDL – Tabel releases

Nama Tabel	releases
Deskripsi	Tabel releases berfungsi sebagai hubungan antar collection yang akan direleases oleh artist
Script SQL	
<pre>/*===== /* Table: RELEASES /*===== create table RELEASES (COLLECTION_ID INT4 not null, ARTIST_ID INT4 not null, constraint PK_RELEASES primary key (COLLECTION_ID, ARTIST_ID));</pre>	



Tabel 57 DDL – Tabel reviews

Nama Tabel	reviews
Deskripsi	Tabel ini merupakan penyimpanan review yang dibuat oleh users terhadap collections
Script SQL	<pre>/* * Table: REVIEWS */ create table REVIEWS (REVIEW TEXT null, RATING NUMERIC(3,0) not null, "TIMESTAMP" DATE not null, REVIEW_ID INT4 not null, USER_ID INT4 not null, COLLECTION_ID INT4 not null,</pre>

```
    constraint PK_REVIEWS primary key (REVIEW_ID)  
);
```

Screenshot Hasil

Tabel 58 DDL – Tabel socials

Nama Tabel	socials
Deskripsi	Tabel ini merupakan penyimpanan social media link artists
Script SQL	
<pre>/*===== /* Table: SOCIALS /*===== create table SOCIALS (SOCIAL_ID INT4 not null, ARTIST_ID INT4 not null, SOCIAL_MEDIA_LINK VARCHAR(2048) not null, constraint PK_SOCIALS primary key (SOCIAL_ID));</pre>	

Screenshot Hasil

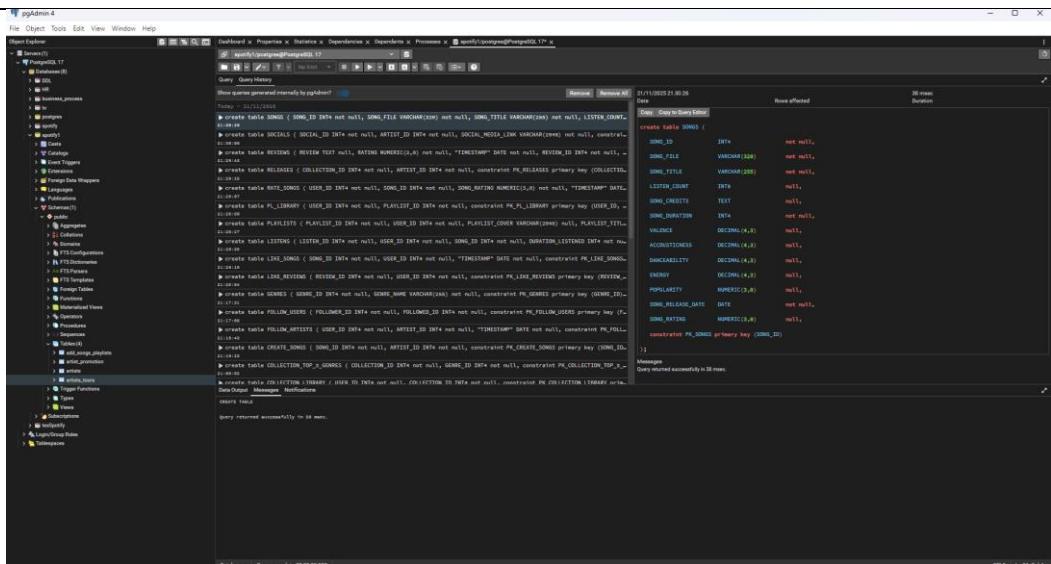
<Gambar Hasil Percobaan>

Tabel 59 DDL – Tabel songs

Nama Tabel	songs
Deskripsi	Tabel ini merupakan penyimpanan songs terdiri dari beberapa atribut
Script SQL	
<pre>/*===== /* Table: SONGS /*===== create table SONGS (SONG_ID INT4 not null, SONG_FILE VARCHAR(320) not null, SONG_TITLE VARCHAR(255) not null, LISTEN_COUNT INT8 null, SONG_CREDITS TEXT null, SONG_DURATION INT4 not null, VALENCE DECIMAL(4,3) null,</pre>	

```
ACCOUSTICNESS      DECIMAL(4,3)      null,  
DANCEABILITY       DECIMAL(4,3)      null,  
ENERGY             DECIMAL(4,3)      null,  
POPULARITY         NUMERIC(3,0)     null,  
SONG_RELEASE_DATE DATE            not null,  
SONG_RATING        NUMERIC(3,0)     null,  
constraint PK_SONGS primary key (SONG_ID)  
);
```

Screenshot Hasil

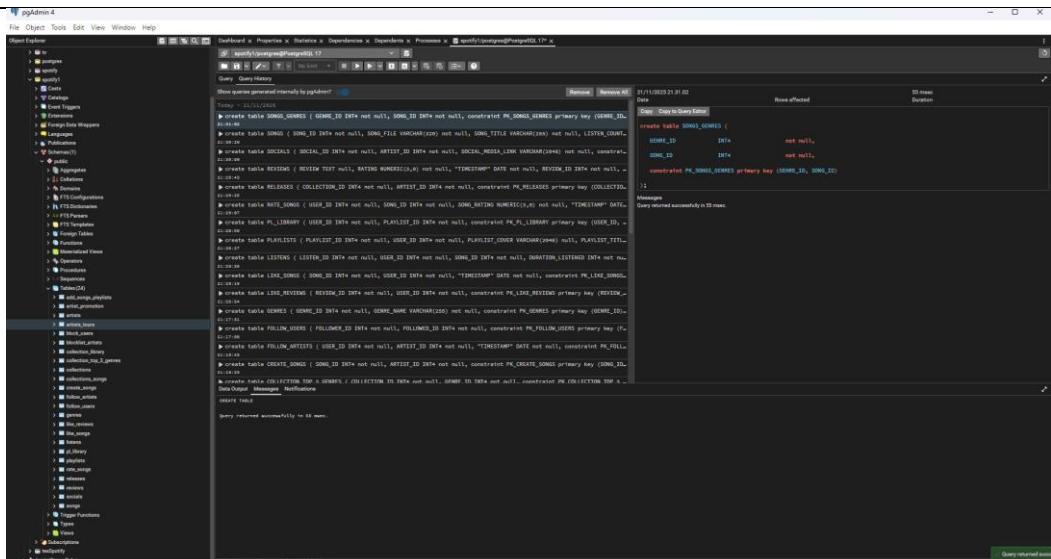


Tabel 60 DDL – Tabel song_genres

Nama Tabel	Song_genres
Deskripsi	Tabel ini berisi hubungan antara genres dan songs
Script SQL	
/*=====*/ /* Table: SONGS_GENRES */	

```
/*
create table SONGS_GENRES (
    GENRE_ID          INT4          not null,
    SONG_ID           INT4          not null,
    constraint PK_SONGS_GENRES primary key (GENRE_ID, SONG_ID)
);
```

Screenshot Hasil

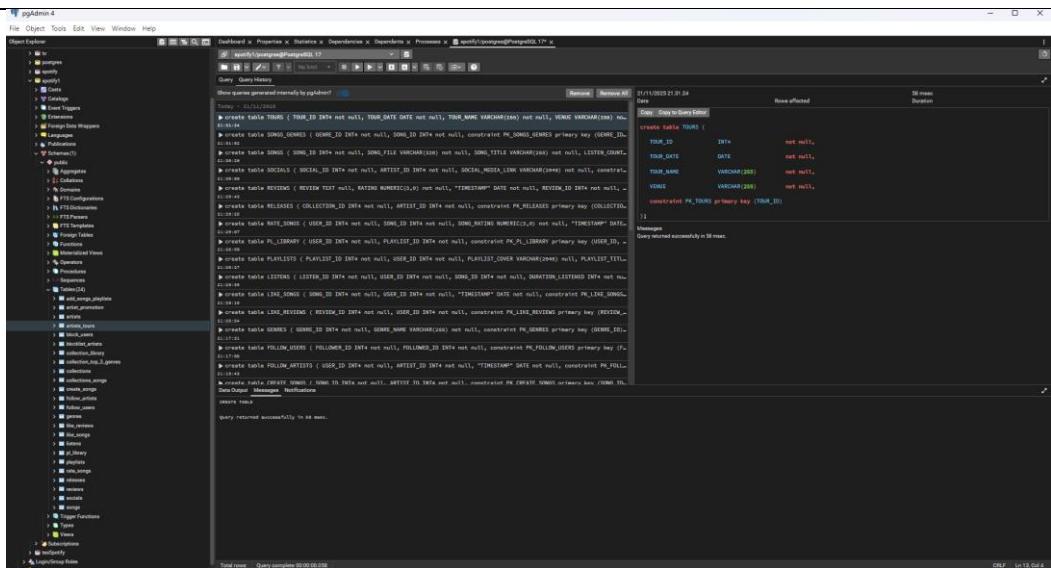


Tabel 61 DDL – Tabel tours

Nama Tabel	tours
Deskripsi	Tabel ini berfungsi menyimpan data tours dengan beberapa atributnya
Script SQL	
/* /* Table: TOURS */	

```
/*
create table TOURS (
    TOUR_ID          INT4          not null,
    TOUR_DATE        DATE          not null,
    TOUR_NAME        VARCHAR(255)  not null,
    VENUE            VARCHAR(255)  not null,
constraint PK_TOURS primary key (TOUR_ID)
);
```

Screenshot Hasil



Tabel 62 DDL – Tabel users

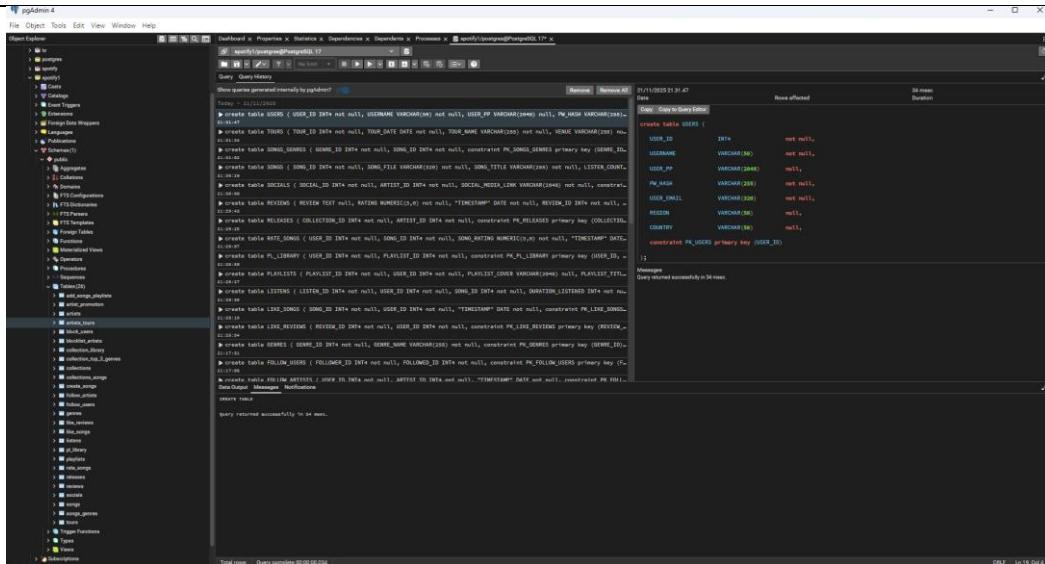
Nama Tabel	users
Deskripsi	Tabel ini berfungsi menyimpan data utama users seperti user id, username, password, email dll.
Script SQL	
	/* =====*/

```

/* Table: USERS
*/
create table USERS (
    USER_ID          INT4          not null,
    USERNAME         VARCHAR(50)    not null,
    USER_PP          VARCHAR(2048)  null,
    PW_HASH          VARCHAR(255)   not null,
    USER_EMAIL       VARCHAR(320)   not null,
    REGION           VARCHAR(50)    null,
    COUNTRY          VARCHAR(50)    null,
    constraint PK_USERS primary key (USER_ID)
);

```

Screenshot Hasil



Tabel 63 DDL - Sequence

Nama Sequence	User_id, artist_id, song_id, playlist_id, collection_id, genre_id, review_id, tour_id, social_id, listen_id, add_songs_playlist_id
----------------------	--

Deskripsi	Sequence yang digunakan untuk menghasilkan nilai ID unik pada berbagai entitas utama dalam sistem, seperti user, artist, song, playlist, collection, genre, review, tour, social, listening history, dan data penambahan lagu ke playlist. Sequence ini memastikan setiap record memiliki primary key yang konsisten, tidak duplikatif, dan tetap aman digunakan untuk operasi insert secara paralel
	<p style="text-align: center;">Script SQL</p> <pre>-- Sequence untuk artists CREATE SEQUENCE seq_artists_id START 1; ALTER TABLE ARTISTS ALTER COLUMN ARTIST_ID SET DEFAULT nextval('seq_artists_id'); ALTER SEQUENCE seq_artists_id OWNED BY ARTISTS.ARTIST_ID; -- Sequence untuk Songs CREATE SEQUENCE seq_songs_id START 1; ALTER TABLE SONGS ALTER COLUMN SONG_ID SET DEFAULT nextval('seq_songs_id'); ALTER SEQUENCE seq_songs_id OWNED BY SONGS.SONG_ID; -- Sequence untuk Users CREATE SEQUENCE seq_users_id START 1; ALTER TABLE USERS ALTER COLUMN USER_ID SET DEFAULT nextval('seq_users_id'); ALTER SEQUENCE seq_users_id OWNED BY USERS.USER_ID; -- Sequence untuk Playlists CREATE SEQUENCE seq_playlists_id START 1;</pre>

```

ALTER TABLE PLAYLISTS
ALTER COLUMN PLAYLIST_ID SET DEFAULT nextval('seq_playlists_id');
ALTER SEQUENCE seq_playlists_id OWNED BY PLAYLISTS.PLAYLIST_ID;

-- Sequence untuk Collections
CREATE SEQUENCE seq_collections_id START 1;
ALTER TABLE COLLECTIONS
ALTER COLUMN COLLECTION_ID SET DEFAULT nextval('seq_collections_id');
ALTER SEQUENCE seq_collections_id OWNED BY COLLECTIONS.COLLECTION_ID;

-- Sequence untuk Genres
CREATE SEQUENCE seq_genres_id START 1;
ALTER TABLE GENRES
ALTER COLUMN GENRE_ID SET DEFAULT nextval('seq_genres_id');
ALTER SEQUENCE seq_genres_id OWNED BY GENRES.GENRE_ID;

-- Sequence untuk Reviews
CREATE SEQUENCE seq_reviews_id START 1;
ALTER TABLE REVIEWS
ALTER COLUMN REVIEW_ID SET DEFAULT nextval('seq_reviews_id');
ALTER SEQUENCE seq_reviews_id OWNED BY REVIEWS.REVIEW_ID;

-- Sequence untuk Tour
CREATE SEQUENCE seq_tours_id START 1;
ALTER TABLE TOURS
ALTER COLUMN TOUR_ID SET DEFAULT nextval('seq_tours_id');
ALTER SEQUENCE seq_tours_id OWNED BY TOURS.TOUR_ID;

```

```
-- Sequence untuk Socials

CREATE SEQUENCE seq_socials_id START 1;

ALTER TABLE SOCIALS

ALTER COLUMN SOCIAL_ID SET DEFAULT nextval('seq_socials_id');

ALTER SEQUENCE seq_socials_id OWNED BY SOCIALS.SOCIAL_ID;

-- Sequence untuk Listens

CREATE SEQUENCE seq_listens_id START 1;

ALTER TABLE LISTENS

ALTER COLUMN LISTEN_ID SET DEFAULT nextval('seq_listens_id');

ALTER SEQUENCE seq_listens_id OWNED BY LISTENS.LISTEN_ID;

-- Sequence untuk ADD_SONGS_PLAYLISTS

CREATE SEQUENCE seq_add_songs_playlist_id START 1;

ALTER TABLE ADD_SONGS_PLAYLISTS

ALTER COLUMN ADD_SONG_PL_ID SET DEFAULT nextval('seq_add_songs_playlist_id');

ALTER SEQUENCE seq_add_songs_playlist_id OWNED BY
ADD_SONGS_PLAYLISTS.ADD_SONG_PL_ID;
```

Screenshot Hasil

```

-- Sequence untuk Reviews
CREATE SEQUENCE seq_reviews_id START 1;
ALTER TABLE REVIEWS
ALTER COLUMN REVIEW_ID SET DEFAULT nextval('seq_reviews_id');
ALTER SEQUENCE seq_reviews_id OWNED BY REVIEWS.REVIEW_ID;

-- Sequence untuk Tour
CREATE SEQUENCE seq_tours_id START 1;
ALTER TABLE TOURS
ALTER COLUMN TOUR_ID SET DEFAULT nextval('seq_tours_id');
ALTER SEQUENCE seq_tours_id OWNED BY TOURS.TOUR_ID;

```

Total rows: Query complete 00:00:00.111 CRLF Ln 1257, Col 1

Tabel 64 DDL - Constraints

Nama Constraint	CHK_SONG_METRICS, CHK_SONG_DURATION_POSITIVE, CHK_SONG_POPULARITY_RANGE, CHK_RATE_SONG_RANGE, CHK_REVIEW_RATING_RANGE, CHK_LISTENER_POSITIVE, CHK_PLAYLIST_ORDER_POSITIVE, CHK_DISC_TRACK_POSITIVE, CHK_PW_HASH_LENGTH, CHK_USERNAME_NO_WHITESPACE, CHK_ARTISTNAME_NO_WHITESPACE, CHK_NO_SELF_FOLLOW, CHK_NO_SELF_BLOCK, CHK_EMAIL_FORMAT, CHK_COLLECTION_TYPE_VALID, set defaults, FK_PROMO_OWN_RELEASE, UQ_USER_COLLECTION REVIEW
Deskripsi	Berbagai <i>check constraint</i> diterapkan untuk memastikan seluruh data yang tersimpan tetap valid dan konsisten sesuai aturan bisnis. Pada tabel SONGS, metrik audio seperti valence, danceability, energy,

dan acousticness dibatasi dalam rentang 0–1, sementara durasi lagu harus positif dan popularitas berada pada skala 0–100. Standar ini diperkuat dengan pembatasan nilai rating pada RATE_SONGS dan REVIEWS, yang harus berada dalam rentang 1–100. Selain itu, nomor urut lagu dalam playlist serta nomor disc dan track pada koleksi wajib bernilai positif, dan jumlah monthly listener artis tidak boleh negatif.

Validasi atribut identitas dan format teks juga diterapkan untuk menjaga kualitas data. Password hash diwajibkan memiliki panjang minimal, username serta nama artis tidak boleh berupa whitespace, dan email pengguna harus sesuai format email yang valid. Untuk mencegah hubungan yang tidak logis, sistem melarang pengguna mem-follow atau mem-block dirinya sendiri. Tipe koleksi pun dibatasi hanya pada kategori yang diizinkan seperti Album, EP, Single, dan Compilation. Pada sisi integritas relasional, sistem memastikan bahwa artis hanya dapat mempromosikan koleksi yang benar-benar dirilis olehnya, serta mencegah satu pengguna memberikan lebih dari satu review terhadap koleksi yang sama.

Script SQL

```
ALTER TABLE SONGS ADD CONSTRAINT CHK_SONG_METRICS
```

```
    CHECK (
        (VALENCE >= 0 AND VALENCE <= 1) AND
        (DANCEABILITY >= 0 AND DANCEABILITY <= 1) AND
        (ENERGY >= 0 AND ENERGY <= 1) AND
        (ACCOUSTICNESS >= 0 AND ACCOUSTICNESS <= 1)
    );
```

```
ALTER TABLE SONGS ADD CONSTRAINT CHK_SONG_DURATION_POSITIVE
    CHECK (SONG_DURATION > 0);
```

```
ALTER TABLE SONGS ADD CONSTRAINT CHK_SONG_POPULARITY_RANGE  
CHECK (POPULARITY >= 0 AND POPULARITY <= 100);
```

```
ALTER TABLE RATE_SONGS ADD CONSTRAINT CHK_RATE_SONG_RANGE  
CHECK (SONG_RATING >= 1 AND SONG_RATING <= 100);
```

```
ALTER TABLE REVIEWS ADD CONSTRAINT CHK_REVIEW_RATING_RANGE  
CHECK (RATING >= 1 AND RATING <= 100);
```

```
ALTER TABLE ARTISTS ADD CONSTRAINT CHK_LISTENER_POSITIVE  
CHECK (MONTHLY_LISTENER_COUNT >= 0);
```

```
ALTER TABLE ADD_SONGS_PLAYLISTS ADD CONSTRAINT  
CHK_PLAYLIST_ORDER_POSITIVE  
CHECK (NO_URUT > 0);
```

```
ALTER TABLE COLLECTIONS_SONGS ADD CONSTRAINT CHK_DISC_TRACK_POSITIVE  
CHECK (NOMOR_DISC > 0 AND NOMOR_TRACK > 0);
```

```
ALTER TABLE USERS ADD CONSTRAINT CHK_PW_HASH_LENGTH  
CHECK (LENGTH(PW_HASH) >= 8);
```

```
ALTER TABLE USERS ADD CONSTRAINT CHK_USERNAME_NO_WHITESPACE  
CHECK (LENGTH(TRIM(USERNAME)) > 0);
```

```
ALTER TABLE ARTISTS ADD CONSTRAINT CHK_ARTISTNAME_NO_WHITESPACE  
CHECK (LENGTH(TRIM(ARTIST_NAME)) > 0);
```

```

ALTER TABLE FOLLOW_USERS
ADD CONSTRAINT CHK_NO_SELF_FOLLOW
CHECK (FOLLOWER_ID <> FOLLOWED_ID);

ALTER TABLE BLOCK_USERS
ADD CONSTRAINT CHK_NO_SELF_BLOCK
CHECK (BLOCKER_ID <> BLOCKED_ID);
/* --- VALIDASI FORMAT TEKS & TIPE --- */

ALTER TABLE USERS ADD CONSTRAINT CHK_EMAIL_FORMAT
CHECK (USER_EMAIL LIKE '%_@__%.__%');

ALTER TABLE COLLECTIONS ADD CONSTRAINT CHK_COLLECTION_TYPE_VALID
CHECK (COLLECTION_TYPE IN ('Album', 'EP', 'Single', 'Compilation'));

/* --- NILAI DEFAULT --- */

-- 1. Playlist otomatis Private jika tidak diisi
ALTER TABLE PLAYLISTS ALTER COLUMN ISPUBLIC SET DEFAULT FALSE;

-- 2. Playlist otomatis Tidak Kolaboratif jika tidak diisi
ALTER TABLE PLAYLISTS ALTER COLUMN ISCOLLABORATIVE SET DEFAULT FALSE;

-- artist hanya bisa promosi rilisannya sendiri
ALTER TABLE ARTIST_PROMOTION
ADD CONSTRAINT FK_PROMO_OWN_RELEASE

```

FOREIGN KEY (ARTIST_ID, COLLECTION_ID)

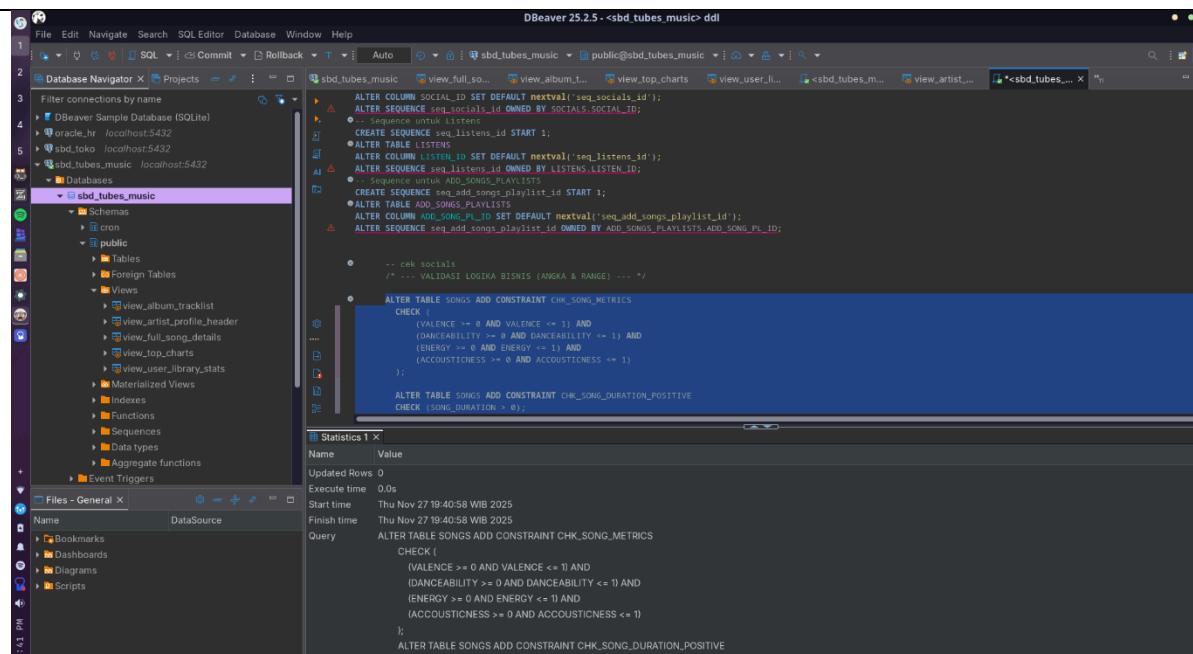
REFERENCES RELEASES (ARTIST_ID, COLLECTION_ID)

ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE REVIEWS

ADD CONSTRAINT UQ_USER_COLLECTION REVIEW UNIQUE (user_id, collection_id);

Screenshot Hasil



The screenshot shows the DBeaver interface with the SQL Editor tab selected. The code in the editor is:

```
ALTER TABLE REVIEWS ADD CONSTRAINT UQ_USER_COLLECTION UNIQUE (user_id, collection_id);
```

5.2 DML

Tabel 65 DML – Follow user

Nama Fitur	Follow User
Deskripsi	User dapat mengikuti atau berhenti mengikuti user lain.
Script SQL	

```
SELECT * FROM follow_users;
```

Screenshot Hasil

Data Output Messages Notifications		
	follower_id [PK] integer	followed_id [PK] integer
24	12	1
25	13	6
26	13	8
27	14	2
28	14	8
29	15	9
30	15	12
Total rows: 30		Query complete 00:00:00.148

Tabel 66 DML – Rilis Collection

Nama Fitur	Rilis Collection	
Deskripsi	Artis dapat merilis lagu atau collection, dicatat tanggal rilis nya. Suatu rilis bisa berupa Pre-Release, dimana wujudnya sudah dapat terlihat	
Script SQL		
<pre>select * from releases</pre>		
<h3>Screenshot Hasil</h3>		
Data Output Messages Notifications		
	collection_id [PK] integer	
24	24	12
25	25	10
26	26	9
27	27	13
28	28	12
29	29	9
30	30	9
31	31	14
Total rows: 31		
Query complete 00:00:00.208		

Tabel 67 DML – Promosi Rilis

Nama Fitur	Promosi Rilis
-------------------	---------------

Deskripsi	Artis dapat memilih satu rilisnya (lagu atau collection) untuk di rekomendasikan di profilnya.
Script SQL	
<pre>select * from artist_promotion</pre>	
Screenshot Hasil	

Data Output Messages Notifications

	artist_id [PK] integer	collection_id integer	komentar_promosi text
1	3	3	Updated promo!
2	10	20	Invalid promotion

Total rows: 2 Query complete 00:00:00.184

Tabel 68 DML – Manajemen Tur

Nama Fitur	Manajemen Tur
Deskripsi	Artist dapat melakukan tour. Seluruh data tentang tour disimpan di table tours dan relasinya di artist_tours.
Script SQL	
<pre>select * from artists_tours select * from tours --tambah tour INSERT INTO tours (tour_id, tour_name, venue, tour_date) VALUES (100, 'World Tour 2025', 'Jakarta Convention Center', '2025-12-20'); -- relasikan artis ke tour INSERT INTO artists_tours (artist_id, tour_id) VALUES (3, 100);</pre>	
Screenshot Hasil	

Data Output Messages Notifications

	tour_id [PK] integer	tour.date date	tour.name character varying (255)	venue character varying (255)
2	2	2025-03-05	Sorong Tour 2002	Gang Cihampelas No. 402
3	3	2025-05-15	Surabaya Fest 1992	Jalan Moch. Toha No. 531
4	4	2025-08-22	Kota Administrasi Jakarta Utara Live 2013	Jalan Kebonjati No. 57
5	5	2025-07-18	Ambon Concert 1998	Gang Kiaracondong No. 8
6	100	2025-12-20	World Tour 2025	Jakarta Convention Center

Total rows: 6 Query complete 00:00:00.259

Data Output			Messages	Notifications
	tour_id [PK] integer	artist_id [PK] integer		
7	4	9		
8	4	4		
9	4	2		
10	5	1		
11	100	3		

Total rows: 11 Query complete 00:00:00.157

Tabel 69 DML – Like Review

Nama Fitur	Like Review				
Deskripsi	Review untuk collection yang dibuat oleh user dapat dilike.				
Script SQL					
<pre>select * from like_reviews</pre>					
Screenshot Hasil					
 <p>Data Output Messages Notifications</p> <table border="1"> <thead> <tr> <th>review_id [PK] integer</th> <th>user_id [PK] integer</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>2</td> </tr> </tbody> </table> <p>Showing rows: 1 to 1 Page No: 1 of 1 14 << >> </p> <p>Total rows: 1 Query complete 00:00:00.273 CRLF Ln 99, Col 1</p>		review_id [PK] integer	user_id [PK] integer	1	2
review_id [PK] integer	user_id [PK] integer				
1	2				

Tabel 70 DML – playlist_collaborators

Nama Fitur	playlist_collaborators
Deskripsi	Memperbarui seluruh user_id pada lagu dalam sebuah playlist ketika playlist tersebut berubah dari <i>collaborative</i> menjadi <i>non-collaborative</i> . Semua lagu yang sebelumnya ditambahkan oleh user lain akan dialihkan menjadi milik pemilik playlist.
Script SQL	

```

SELECT * FROM USERS;

SELECT * FROM add_songs_playlists;

SELECT * FROM PLAYLISTS;

-- cek

-- Data sebelum/sesudah trigger bekerja

SELECT add_song_pl_id, user_id, playlist_id, song_id
FROM add_songs_playlists
WHERE playlist_id = 2

-- Ubah playlist collaborative --> tidak collab

UPDATE playlists
SET iscollaborative = false
WHERE playlist_id = 2;

```

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows the database schema with tables like genres, like_reviews, listens, pl_library, and playlists.
- Query Editor:** Contains the following SQL code:


```

      EXECUTE FUNCTION fix_playlist_collaborators();

      SELECT * FROM USERS; --dongorankunthara
      SELECT * FROM add_songs_playlists;
      SELECT * FROM PLAYLISTS;

      -- cek
      -- Insert playlist owner (user_id = 10 misalnya)
      INSERT INTO users (user_id, username) VALUES (10, 'owner');

      UPDATE playlists
      SET iscollaborative = false
      WHERE playlist_id = 2;
    
```
- Data Output:** Displays the results of the query. The playlists table shows the following data:

playlist_id	user_id	playlist_cover	playlist_title	ispublic	iscollaborative	playlist_desc
1	1	[null]	Playlist dignissimos Abu-abu collab	true	false	Playlist asik br.
2	2	[null]	Playlist reprehenderit Merah jambu collab	true	true	Playlist asik br.
3	3	[null]	Playlist impedit Biru muda collab	true	false	Playlist asik br.
4	4	[null]	Playlist eum Jingga collab	true	true	Playlist asik br.
5	5	[null]	Playlis			
- Messages:** Shows "Successfully run. Total query runtime: 201 msec. 30 rows affected."

```

-- Data sebelum/sesudah trigger bekerja
SELECT * FROM add_songs_playlists
WHERE playlist_id = 2

-- Ubah playlist collaborative --> tidak collab
UPDATE playlists
SET iscollaborative = false

```

Tabel 71 DML – Update follow count

Nama Fitur	Update artist follow count
Deskripsi	Menambahkan otomatis ketika user follow/unfollow
Script SQL	
<pre> INSERT INTO follow_artists (user_id, artist_id) VALUES (9, 4); </pre>	

```

INSERT INTO follow_artists (user_id, artist_id)
VALUES (10, 4);

-- unfollow

DELETE FROM follow_artists
WHERE user_id = 9 AND artist_id = 4;

DELETE FROM follow_artists
WHERE user_id = 2 AND artist_id = 1;

SELECT follower_count FROM artists WHERE artist_id = 4;

SELECT * FROM artists;

SELECT follower_count FROM artists WHERE artist_id = 1;

SELECT * FROM update_artist_follow_count();

```

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface. In the Object Explorer, there are two tables: 'arist_promosi' and 'artists'. The 'artists' table has columns: artist_id, artist_name, bio, monthly_listener_count, artist_pfp, artist_email, banner, and follower_count. The main window shows a query editor with the following SQL code:

```

VALUES (3, 4);

-- unfollow
DELETE FROM follow_artists
WHERE user_id = 5 AND artist_id = 2;
DELETE FROM follow_artists
WHERE user_id = 2 AND artist_id = 1;

SELECT follower_count FROM artists WHERE artist_id = 4;

SELECT * FROM artists;

SELECT follower_count FROM artists WHERE artist_id = 1;

SELECT * FROM update_artist_follow_count();

```

The Data Output tab displays the results of the last SELECT statement:

follower_count	bigint
1	1

Below the table, it says "Showing rows: 1 to 1" and "Page No: 1 of 1". At the bottom right of the pgAdmin window, a message says "Successfully run. Total query runtime: 161 msec. 1 rows affected."

pgAdmin 4

Object Explorer

```
-- unfollow
DELETE FROM follow_artists
WHERE user_id = 5 AND artist_id = 2;
DELETE FROM follow_artists
WHERE user_id = 2 AND artist_id = 1;

SELECT follower_count FROM artists WHERE artist_id = 4;
SELECT * FROM artists;
SELECT follower_count FROM artists WHERE artist_id = 1;
```

Data Output

follower_count	bigint
1	2

Showing rows: 1 to 1 Page No: 1 of 1 14 <> >> |

Total rows: 1 Query complete 00:00:00.139 CRLF Ln 3445, Col 1

06:42 22/11/2025

pgAdmin 4

Object Explorer

```
WHERE user_id = 2 AND artist_id = 1;
SELECT follower_count FROM artists WHERE artist_id = 4;
SELECT * FROM artists;
```

Data Output

artist_id	character varying (2048)	artist_email	character varying (320)	banner	character varying (2048)	follower_count	bigint
14	256	/uploads/artists/artist_19.jpg	Ami Uwais_19@example.com	[null]	[null]		
15	531	/uploads/artists/artist_20.jpg	Kayun Ramadan_20@example.com	[null]	[null]		
16	401	/uploads/artists/artist_3.jpg	Paiman Puspasari_3@example.com	[null]	[null]	1	
17	585	/uploads/artists/artist_4.jpg	Garang Purwanti_4@example.com	[null]	[null]	2	
18	855	/uploads/artists/artist_1.jpg	Aswani Waskita_1@example.com	[null]	[null]	0	
19	835	/uploads/artists/artist_5.jpg	drg. Jelita Sihotang_5@example.com	[null]	[null]		
20	371	/uploads/artists/artist_2.jpg	Drs. Endah Hakim, M.TI_2@example.com	[null]	[null]	1	

Total rows: 20 Query complete 00:00:00.208 CRLF Ln 3447, Col 1

06:43 22/11/2025

pgAdmin 4

Object Explorer

```
WHERE user_id = 9 AND artist_id = 4;
DELETE FROM follow_artists
WHERE user_id = 2 AND artist_id = 1;

SELECT follower_count FROM artists WHERE artist_id = 4;
SELECT * FROM artists;
SELECT follower_count FROM artists WHERE artist_id = 1;
SELECT * FROM update_artist_follow_count();
```

Data Output

follower_count	bigint
1	1

Showing rows: 1 to 1 Page No: 1 of 1 14 <> >> |

Total rows: 1 Query complete 00:00:00.138 CRLF Ln 3442, Col 1

06:44 22/11/2025

```

File Object Tools Edit View Window Help
Object Explorer
arusr_promousn
Columns(3)
artist_id
collection_id
komentar_promosi
Constraints
Indexes
RLS Policies
Rules
Triggers
artists
Columns(8)
artist_id
artist_name
bio
monthly_listener_count
artist_pfp
artist_email
banner
follower_count
Constraints
Indexes
RLS Policies
Rules
Triggers
Query Final_Destination/postgres@Postgres1
Query History
Scratch Pad
3436 -- unfollow
3437 DELETE FROM follow_artists
3438 WHERE user_id = 9 AND artist_id = 4;
3439 DELETE FROM follow_artists
3440 WHERE user_id = 2 AND artist_id = 1;
3441
3442 SELECT follower_count FROM artists WHERE artist_id = 4;
3443
3444
3445
3446
3447
Data Output Messages Notifications
Showing rows: 1 to 20 Page No: 1 of 1
19 585 /uploads/artists/artist_4.jpg Garang Purwanti_4@example.com [null] 1
20 371 /uploads/artists/artist_2.jpg Drs. Endah Hakim, M.TI_2@example.com [null] 1
Total rows: 20 Query complete 00:00:00.149 CRLF Ln 3443, Col 1

```

Tabel 72 DML – update collection rating

Nama Fitur	Update collection rating
Deskripsi	Menghitung ulang rata-rata rating pada tabel collections setiap kali ada perubahan pada tabel reviews. Nilai collection_rating diperbarui berdasarkan AVG(rating) dari semua review dengan collection_id yang sama.
Script SQL	
SELECT * FROM reviews;	
SELECT * FROM collections;	
INSERT INTO reviews (review, rating, review_id, user_id, collection_id)	
VALUES ('bagussss', 5, 1, 10, 1);	
INSERT INTO reviews (review, rating, review_id, user_id, collection_id)	
VALUES ('kerenn', 5, 2, 1, 1);	
SELECT collection_rating FROM collections WHERE collection_id = 1;	

```
-- update collection_rating nya
```

```
UPDATE reviews
```

```
SET rating = 3
```

```
WHERE review_id = 1;
```

```
-- delete review
```

```
DELETE FROM reviews WHERE review_id = 1;
```

Screenshot Hasil

```
AFTER INSERT OR UPDATE OR DELETE  
ON reviews  
FOR EACH ROW  
EXECUTE FUNCTION update_collection_rating();  
  
SELECT * FROM reviews;  
  
INSERT INTO reviews (review, rating, review_id, user_id,  
VALUES ('bagussss', 5, 1, 10, 1);  
  
SELECT collection_rating FROM collections WHERE collectic
```

The screenshot shows the pgAdmin 4 interface with the 'Final_Destination/postgres' database selected. In the 'Query' tab, a script is run that includes an AFTER trigger for the 'reviews' table, an INSERT INTO statement, and a SELECT statement. The 'Data Output' tab displays a single row from the 'reviews' table:

collection_rating	numeric (3)
1	5

A message at the bottom right indicates: "Successfully run. Total query runtime: 226 msec. 1 rows affected."

```
FOR EACH ROW  
EXECUTE FUNCTION update_collection_rating();  
  
SELECT * FROM reviews;  
  
INSERT INTO reviews (review, rating, review_id, user_id,  
VALUES ('bagussss', 5, 1, 10, 1);  
  
SELECT collection_rating FROM collections WHERE collectic  
-- update collection_rating nya  
UPDATE reviews  
SET rating = 3
```

The screenshot shows the pgAdmin 4 interface with the 'Final_Destination/postgres' database selected. In the 'Query' tab, a script is run that includes an AFTER trigger for the 'reviews' table, an INSERT INTO statement, a SELECT statement, an UPDATE statement, and another SET statement. The 'Data Output' tab displays a single row from the 'reviews' table:

review	rating	TIMESTAMP	review_id	user_id	collection_id
bagussss	5	2025-11-21 16:46:24.933663	1	10	1

A message at the bottom right indicates: "Successfully run. Total query runtime: 161 msec. 1 rows affected."

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows a tree view of database objects including collections_songs, create_songs, follow_artists, follow_users, genres, like_reviews, like_songs, listens, pl_library, playlists, rate_songs, releases, and reviews.
- Query Editor:** Contains the following SQL code:

```
3375 INSERT INTO reviews (review, rating, review_id, user_id,
3376 VALUES ('bagusss', 5, 1, 10, 1);
3377
3378 SELECT collection_rating FROM collections WHERE collective_id = 1;
3379
3380 -- update collection_rating nya
3381 UPDATE reviews
3382 SET rating = 3
3383 WHERE review_id = 1;
```
- Data Output:** Shows the result of the UPDATE query: "Query returned successfully in 253 msec."
- Status Bar:** Displays "Total rows: 1 Query complete 00:00:00.253 CRLF Ln 3380, Col 1".

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows a tree view of database objects including collections_songs, create_songs, follow_artists, follow_users, genres, like_reviews, like_songs, listens, pl_library, playlists, rate_songs, releases, and reviews.
- Query Editor:** Contains the following SQL code:

```
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
```

ON reviews
FOR EACH ROW
EXECUTE FUNCTION update_collection_rating();
SELECT * FROM reviews;
INSERT INTO reviews (review, rating, review_id, user_id,
VALUES ('bagusss', 5, 1, 10, 1);
- Data Output:** Shows a data grid with one row of results:

	review	rating	TIMESTAMP	review_id	user_id	collection_id
1	bagusss	3	2025-11-21 16:46:24.936663	1	10	1

- Status Bar:** Displays "Showing rows: 1 to 1 Page No: 1 of 1 Total rows: 1 Query complete 00:00:00.148 CRLF Ln 3373, Col 1".

--- review baru dari user berbeda

```

CREATE TRIGGER update_collection_rating
AFTER INSERT OR UPDATE OR DELETE
ON reviews
FOR EACH ROW
EXECUTE FUNCTION update_collection_rating();

SELECT * FROM reviews;
SELECT * FROM collections;

INSERT INTO reviews (review, rating, review_id, user_id,
VALUES ('bagusss', 5, 1, 10, 1);

```

review_text	rating	review_id	user_id	collection_id
bagusss	3	3368	1	10
kerenn	5	3370	2	1

Successfully run. Total query runtime: 793 msec. 2 rows affected.


```

CREATE TRIGGER update_collection_rating
AFTER INSERT OR UPDATE OR DELETE
ON reviews
FOR EACH ROW
EXECUTE FUNCTION update_collection_rating();

```

collection_id	collection_title	collection_type	collection_cover	collection_release_date	collection_rating	isprelease
31	Dolores corporis	Album	/uploads/covers/coll_32.jpg	2021-11-14	[null]	[null]
32	Corput quis asperiores veritatis	Album	/uploads/covers/coll_33.jpg	2024-01-10	[null]	[null]
33	Quidem quis ipsum	Single	/uploads/covers/coll_34.jpg	2025-07-04	[null]	[null]
34	Officia voluptates eum	Compilation	/uploads/covers/coll_35.jpg	2025-09-27	[null]	[null]
35	Corrupti architecto ipsam	Compilation	/uploads/covers/coll_36.jpg	2022-05-26	[null]	[null]
36	Nihil explicabo	Compilation	/uploads/covers/coll_37.jpg	2021-04-03	[null]	[null]
37	Iste animi rerum	EP	/uploads/covers/coll_38.jpg	2021-04-08	[null]	[null]
38	Nostrum magni	Compilation	/uploads/covers/coll_39.jpg	2023-03-14	[null]	[null]
39	Veritatis qui aliquam	Compilation	/uploads/covers/coll_40.jpg	2025-06-06	[null]	[null]
40	Inventore suscipit	Single	/uploads/covers/coll_1.jpg	2022-12-15	4	[null]

Tabel 73 DML – Update collection top3 genres

Nama Fitur	Update collection top3 genres
Deskripsi	<p>Otomatis menghitung 3 genre terbanyak dalam sebuah collection setiap kali ada perubahan pada tabel collections_songs. Mekanismenya:</p> <ol style="list-style-type: none"> 1. hapus data genre lama untuk collection tersebut, 2. hitung frekuensi tiap genre dari semua lagu melalui songs_genres,

3.ambil 3 genre dengan jumlah tertinggi dan simpan ke collection_top_3_genres.
Contoh: jika sebuah collection memiliki genre Pop (4), Jazz (2), Rock (2), dan Noise Pop (1), maka yang disimpan adalah Pop, Jazz, dan Rock sebagai top 3 genre.

Script SQL

```
-- cek  
-- Update supaya trigger jalan  
UPDATE collections_songs  
SET nomor_track = nomor_track  
WHERE collection_id = 1;  
  
UPDATE collections_songs  
SET nomor_track = nomor_track  
WHERE collection_id = 2;  
  
-- Lihat hasil top 3 genre  
SELECT * FROM collection_top_3_genres  
WHERE collection_id = 1;  
  
SELECT * FROM collection_top_3_genres  
WHERE collection_id = 2;  
  
SELECT * FROM collection_top_3_genres;
```

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows the database schema with tables: songs_genres, tours, and users.
- Query Editor:** Contains the following SQL code:

```
3761  SELECT * FROM collections_songs;
3762  -- cek
3763  -- Update supaya trigger jalan
3764  UPDATE collections_songs
3765  SET nomor_track = nomor_track
3766  WHERE collection_id = 1;
3767
3768  -- Lihat hasil top 3 genre
3769  SELECT * FROM collection_top_3_genres
3770  WHERE collection_id = 1;
3771
```
- Data Output:** Shows the result of the update query: "UPDATE 8".
Shows the result of the select query: "Query returned successfully in 193 msec."
- Messages:** Shows a green message: "Query returned successfully in 193 msec.".
- Notifications:** Shows a green message: "Query returned successfully in 193 msec.".
- System Bar:** Shows the total rows (8), query completion time (00:00:00.193), and system status (CRLF, Ln 3703, Col 1).

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows the database schema with tables: songs_genres, tours, and users.
- Query Editor:** Contains the same SQL code as the first screenshot.
- Data Output:** Shows the result of the select query:

collection_id	genre_id
1	1
2	1
3	1
- Messages:** Shows a green message: "Successfully run. Total query runtime: 675 msec. 3 rows affected.".
- Notifications:** Shows a green message: "Successfully run. Total query runtime: 675 msec. 3 rows affected.".
- System Bar:** Shows the total rows (3), query completion time (00:00:00.675), and system status (CRLF, Ln 3708, Col 1).

pgAdmin 4

Object Explorer

- > Account
- > DML_Lat
- > Final Destination
- > Final_Destination
 - > Casts
 - > Catalogs
 - > Event Triggers
 - > Extensions
 - > Foreign Data Wrappers
 - > Languages
 - > Publications
 - > Schemas(1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures

Query

```

3712 -- Lihat hasil top 3 genre
3713 SELECT * FROM collection_top_3_genres
3714 WHERE collection_id = 1;
3715
3716
3717 SELECT * FROM collection_top_3_genres
3718 WHERE collection_id = 2;
3719
3720 SELECT * FROM collection_top_3_genres;

```

Data Output

collection_id [PK] integer	genre_id [PK] integer
1	1
2	1
3	1
4	2
5	2
6	2

Total rows: 6 Query complete 00:00:00.996

13:58
27/11/2025

pgAdmin 4

Object Explorer

- > Account
- > DML_Lat
- > Final Destination
- > Final_Destination
 - > Casts
 - > Catalogs
 - > Event Triggers
 - > Extensions
 - > Foreign Data Wrappers
 - > Languages
 - > Publications
 - > Schemas(1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures

Query

```

3717 WHERE collection_id = 2;
3718
3719
3720
3721
3722
3723
3724
3725 -- Update_listen_stats
CREATE OR REPLACE FUNCTION update_listen_stats()

```

Data Output

genre_id [PK] integer	genre_name character varying (255)
1	Pop
2	Rock
3	Jazz
4	Dangdut
5	Indie
6	Hip Hop
7	R&B

Total rows: 10 Query complete 00:00:00.732

14:00
27/11/2025

Tabel 74 DML – Update review timestamp

Nama Fitur	Update review timestamp
Deskripsi	Otomatis memperbarui kolom timestamp setiap kali isi review atau rating berubah. Jika salah satu di-update, nilai timestamp diganti dengan CURRENT_TIMESTAMP, sehingga riwayat perubahan review selalu tercatat dengan akurat.

Script SQL

-- tabel review

```
SELECT * FROM reviews;
```

-- update review timestamp

```
UPDATE reviews
```

```
SET review = 'Review baru test trigger'
```

```
WHERE review_id = 1;
```

-- update rating timestamp

```
UPDATE reviews
```

```
SET rating = rating + 1
```

```
WHERE review_id = 1;
```

```
SELECT * FROM COLLECTIONS;
```

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface. In the Object Explorer, there are several schemas and tables listed under the 'releases' schema, including 'reviews' and 'collections'. The 'reviews' table has columns: review, rating, and timestamp. The 'collections' table has columns: collection_id, user_id, and review_id.

In the main window, a query is being run:

```
3524  CREATE OR REPLACE
3525  ON reviews
3526  FOR EACH ROW
3527  EXECUTE FUNCTION update_review_timestamp();
3528
3529 -- tabel review
3530
3531
3532
3533
```

The results of the query are displayed in a table:

	review	rating	TIMESTAMP	review_id	user_id	collection_id
1	bagusss	3	2025-11-21 16:46:24.933663	1	10	1
2	kerenn	5	2025-11-21 17:35:28.791737	2	1	1

A message at the bottom of the query window says: "Successfully run. Total query runtime: 158 msec. 2 rows affected."

pgAdmin 4

Object Explorer

```

releases
  Columns(2)
    collection_id
    artist_id
  Constraints
  Indexes
  RLS Policies
  Rules
  Triggers
reviews
  Columns(6)
    review
    rating
    TIMESTAMP
    review_id
    user_id
    collection_id
  Constraints
  Indexes
  RLS Policies
  Rules
  Triggers
  socials
  songs
    Columns(13)

```

Final_Destination/postgres@Postgres16

Query History

```

EXECUTE FUNCTION update_review_timestamp();

-- tabel review
SELECT * FROM reviews;

-- update review timestamp
UPDATE reviews
SET review = 'Review baru test trigger'
WHERE review_id = 1;

```

Data Output

Messages

Notifications

UPDATE 1

Query returned successfully in 138 msec.

Total rows: Query complete 00:00:00.138

✓ Query returned successfully in 138 msec. X

CRLF Ln 3531, Col 1

09:38 22/11/2025

pgAdmin 4

Object Explorer

```

releases
  Columns(2)
    collection_id
    artist_id
  Constraints
  Indexes
  RLS Policies
  Rules
  Triggers
reviews
  Columns(6)
    review
    rating
    TIMESTAMP
    review_id
    user_id
    collection_id
  Constraints
  Indexes
  RLS Policies
  Rules
  Triggers
  socials
  songs
    Columns(13)

```

Final_Destination/postgres@Postgres16

Query History

```

EXECUTE FUNCTION update_review_timestamp();

-- tabel review
SELECT * FROM reviews;

-- update review timestamp
UPDATE reviews
SET review = 'Review baru test trigger'
WHERE review_id = 1;

```

Data Output

Messages

Notifications

review_text	rating	TIMESTAMP	review_id	user_id	collection_id
kerenn	5	2025-11-21 17:35:28.791737	2	1	1
Review baru test trigger	3	2025-11-22 09:38:14.888103	1	10	1

Showing rows: 1 to 2 | Page No: 1 of 1

✓ Successfully run. Total query runtime: 414 msec. 2 rows affected. X

CRLF Ln 3529, Col 1

09:39 22/11/2025

-- cek update rating

```

-- update review timestamp
UPDATE reviews
SET review = 'Review baru test trigger'
WHERE review_id = 1;

-- update rating timestamp
UPDATE reviews
SET rating = rating + 1
WHERE review_id = 1;

```

Query returned successfully in 720 msec.

```

run each row
EXECUTE FUNCTION update_review_timestamp();

-- tabel review
SELECT * FROM reviews;

```

```

-- update review timestamp
UPDATE reviews
SET review = 'Review baru test trigger'
WHERE review_id = 1;

```

review	rating	TIMESTAMP	review_id	user_id	collection_id
kerenn	5	2025-11-21 17:35:28.791737	2	1	1
Review baru test trigger	4	2025-11-22 09:42:38.400275	1	10	1

Successfully run. Total query runtime: 700 msec. 2 rows affected.

Tabel 75 DML – Update song rating

Nama Fitur	Update song rating
Deskripsi	Ketika ada INSERT, UPDATE, atau DELETE pada RATE_SONGS. Trigger akan menghitung ulang rata-rata rating untuk setiap song_id menggunakan AVG(song_rating) dari semua rating yang ada, lalu menyimpannya ke SONGS.song_rating. Dengan begitu, nilai rating lagu selalu akurat dan terbaru.
Script SQL	
SELECT * FROM songs;	

```
-- cek  
-- Insert rating baru  
  
INSERT INTO rate_songs (user_id, song_id, song_rating)  
VALUES (2, 2, 4);
```

```
-- Update rating song  
  
UPDATE rate_songs  
SET song_rating = 5  
WHERE user_id = 2 AND song_id = 2;
```

```
-- Update rating song  
  
UPDATE rate_songs  
SET song_rating = 5  
WHERE user_id = 3 AND song_id = 2;
```

```
-- Insert baru  
  
INSERT INTO rate_songs (user_id, song_id, song_rating)  
VALUES (3, 2, 3);
```

```
-- Delete  
  
DELETE FROM rate_songs  
WHERE user_id = 2 AND song_id = 2;
```

```
SELECT song_id, song_rating FROM songs WHERE song_id = 2;
```

Screenshot Hasil

```
--- insert rating baru dengan user_id 2 rating_song 4
```

pgAdmin 4

Object Explorer

```

tours
  Columns(4)
    tour_id
    tour_date
    tour_name
    venue
  Constraints
  Indexes
  RLS Policies
  Rules
  Triggers
users
  Columns(7)
    user_id
    username
    user_pfp
    pw_hash
    user_email
    region
    country
  Constraints
  Indexes
  RLS Policies
  Rules

```

Final_Destination/postgres@Postgres16

Query Query History

```

3631   ON rate_songs
3632   FOR EACH ROW
3633     EXECUTE FUNCTION update_song_rating();
3634
3635
3636
3637
3638
3639   -- cek
3640   -- Insert rating baru
3641   INSERT INTO rate_songs (user_id, song_id, song_rating)
3642     VALUES (2, 2, 4);
3643
3644   SELECT song_id, song_rating FROM songs WHERE song_id = 2;

```

Data Output Messages Notifications

INSERT 0 1

Query returned successfully in 375 msec.

Total rows: Query complete 00:00:00.375 CRLF Ln 3639, Col 1

pgAdmin 4

Object Explorer

```

tours
  Columns(4)
    tour_id
    tour_date
    tour_name
    venue
  Constraints
  Indexes
  RLS Policies
  Rules
  Triggers
users
  Columns(7)
    user_id
    username
    user_pfp
    pw_hash
    user_email
    region
    country
  Constraints
  Indexes
  RLS Policies
  Rules

```

Final_Destination/postgres@Postgres16

Query Query History

```

3635
3636
3637
3638
3639   SELECT * FROM songs;
3640   -- cek
3641   -- Insert rating baru
3642   INSERT INTO rate_songs (user_id, song_id, song_rating)
3643     VALUES (2, 2, 4);
3644
3645
3646
3647

```

Data Output Messages Notifications

song_id	song_rating
[PK] integer	numeric (3)
1	2
	4

Showing rows: 1 to 1 Page No: 1 of 1

Total rows: 1 Query complete 00:00:00.220 CRLF Ln 3644, Col 1

-- update rating song

pgAdmin 4

Object Explorer

```

3639 INSERT INTO rate_songs (user_id, song_id, song_rating)
3640   VALUES (2, 2, 4);
3641
3642 -- Update rating song
3643 UPDATE rate_songs
3644   SET song_rating = 5
3645 WHERE user_id = 2 AND song_id = 2;
3646
3647
3648 SELECT song_id, song_rating FROM songs WHERE song_id = 2;
3649
3650
3651
  
```

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 688 msec.

Total rows: Query complete 00:00:00.690 CRLF Ln 3642; Col 1

16:25 22/11/2025

pgAdmin 4

Object Explorer

```

3639 INSERT INTO rate_songs (user_id, song_id, song_rating)
3640   VALUES (2, 2, 4);
3641
3642 -- Update rating song
3643 UPDATE rate_songs
3644   SET song_rating = 5
3645 WHERE user_id = 2 AND song_id = 2;
3646
3647
3648 SELECT song_id, song_rating FROM songs WHERE song_id = 2;
3649
3650
3651
  
```

Data Output Messages Notifications

song_id	song.rating
2	5

Showing rows: 1 to 1 Page No: 1 of 1

Successfully run. Total query runtime: 128 msec. 1 rows affected.

Total rows: 1 Query complete 00:00:00.128 CRLF Ln 3648; Col 1

16:25 22/11/2025

-- insert baru dengan rating 3 user id nya 3

pgAdmin 4

Object Explorer

```

3643 UPDATE rate_songs
3644   SET song_rating = 5
3645 WHERE user_id = 2 AND song_id = 2;
3646
3647 -- Update rating song
3648 UPDATE rate_songs
3649   SET song_rating = 5
3650 WHERE user_id = 3 AND song_id = 2;
3651
3652
3653 INSERT INTO rate_songs (user_id, song_id, song_rating)
3654   VALUES (3, 2, 3);
3655
  
```

Data Output Messages Notifications

INSERT 0 1

Query returned successfully in 462 msec.

Total rows: Query complete 00:00:00.462 CRLF Ln 3655; Col 1

16:33 22/11/2025

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows the database schema with two main tables: "tours" and "users".
- Query Editor:** Displays the following SQL code:

```
-- Update rating song
UPDATE rate_songs
SET song_rating = 5
WHERE user_id = 3 AND song_id = 2;

-- Insert baru
INSERT INTO rate_songs (user_id, song_id, song_rating)
VALUES (3, 2, 3);

SELECT song_id, song_rating FROM songs WHERE song_id = 2;
```
- Data Output:** A table showing the result of the SELECT query:

song_id	song_rating
1	2
4	4
- Messages:** A green message box indicates: "Successfully run. Total query runtime: 604 msec. 1 rows affected."
- System Bar:** Shows the total rows (1), query completion time (00:00:00.604), and the current date and time (22/11/2025 16:35).

-- Delete

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows the database schema with the same two tables: "tours" and "users".
- Query Editor:** Displays the following SQL code:

```
UPDATE rate_songs
SET song_rating = 5
WHERE user_id = 3 AND song_id = 2;

-- Insert baru
INSERT INTO rate_songs (user_id, song_id, song_rating)
VALUES (3, 2, 3);

-- Delete
DELETE FROM rate_songs
WHERE user_id = 2 AND song_id = 2;

SELECT song_id, song_rating FROM songs WHERE song_id = 2;
```
- Data Output:** A message box indicates: "Query returned successfully in 140 msec."
- Messages:** A green message box indicates: "Query returned successfully in 140 msec."
- System Bar:** Shows the total rows (1), query completion time (00:00:00.140), and the current date and time (22/11/2025 16:38).

5.3 PL/SQL

Tabel 76 PL/SQL – Function get_song_audio_features

Nama Function	get_song_audio_features
Parameter	p_song_id IN INT – ID lagu yang ingin diambil fiturnya
Deskripsi	Function ini digunakan untuk mengambil fitur audio numerik dari sebuah lagu berdasarkan song_id. Fitur audio ini mencerminkan karakteristik musik dari lagu, seperti “positif/ceria”, akustik, danceability, dan energi lagu. Function mengembalikan hasil berupa tabel dengan informasi lagu dan nilai fitur audio. Jika song_id tidak ditemukan, function mengembalikan tabel kosong.
Script SQL	
<pre>CREATE OR REPLACE FUNCTION get_song_audio_features(p_song_id INT) RETURNS TABLE (song_id INT, -- ID lagu song_title VARCHAR, -- Judul lagu</pre>	

```
valence DECIMAL(4,3),      -- Valence lagu
acousticness DECIMAL(4,3),   -- Acousticness lagu
danceability DECIMAL(4,3),   -- Danceability lagu
energy DECIMAL(4,3)         -- Energy lagu
) AS $$

BEGIN
    -- Mengambil data audio features dari tabel SONGS berdasarkan
song_id
    RETURN QUERY
    SELECT
        s.song_id,
        s.song_title,
        s.valence, -- seberapa "positif/ceria"
        s.acousticness, -- seberapa akustik
        s.danceability, -- dance
        s.energy -- seberapa enerjik
    FROM songs s
    WHERE s.song_id = p_song_id;

    -- Jika song_id tidak ada, hasilnya akan kosong
END;
$$ LANGUAGE plpgsql;
```

Screenshot Hasil

pgAdmin 4

Object Explorer

```

CREATE OR REPLACE FUNCTION get_song_audio_features(p_song_id INT)
RETURNS TABLE (
    song_id INT, -- ID lagu
    song_title VARCHAR, -- Judul lagu
    valence DECIMAL(4,3), -- Valence lagu
    acousticness DECIMAL(4,3), -- Acousticness lagu
    danceability DECIMAL(4,3), -- Danceability lagu
    energy DECIMAL(4,3) -- Energy lagu
) AS $$ 
BEGIN
    -- Mengambil data audio features dari tabel SONGS berdasarkan id
    RETURN;
END;

```

Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 578 msec.

Total rows: 0 Query complete 00:00:00.578 CRLF Ln 4379, Col 21

08:17 26/11/2025

pgAdmin 4

Object Explorer

```

$$ LANGUAGE plpgsql;
-- cek
-- coba panggil function dengan song_id
SELECT * FROM get_song_audio_features(3);

```

SELECT * FROM get_song_audio_features(3);

-- cek song_id nya yg gak ada di data

-- cek song_id nya yg gak ada di data

SELECT * FROM get_song_audio_features(500);

Data Output Messages Notifications

song_id	song_title	valence	acousticness	danceability	energy
3	Minima non facilis ullam	0.730	0.240	0.740	0.370

Successfully run. Total query runtime: 116 msec. 1 rows affected. CRLF Ln 4381, Col 1

Total rows: 1 Query complete 00:00:00.116 08:17 26/11/2025

pgAdmin 4

Object Explorer

```

$$ LANGUAGE plpgsql;
-- cek
-- coba panggil function dengan song_id
SELECT * FROM get_song_audio_features(3);

```

SELECT * FROM get_song_audio_features(3);

-- cek song_id nya yg gak ada di data

-- cek song_id nya yg gak ada di data

SELECT * FROM get_song_audio_features(500);

Data Output Messages Notifications

song_id	song_title	valence	acousticness	danceability	energy

Successfully run. Total query runtime: 567 msec. 0 rows affected. CRLF Ln 4385, Col 1

Total rows: 0 Query complete 00:00:00.567 08:17 26/11/2025

Tabel 77 PL/SQL - Function get_recently_played

Nama Function	get_recently_played
Parameter	<p>p_user_id INT --> ID user yang ingin diambil riwayat pemutarannya</p> <p>p_limit INT --> Jumlah maksimal lagu terbaru yang akan ditampilkan</p>
Deskripsi	<p>Function get_recently_played digunakan untuk mengambil daftar lagu yang paling terakhir diputar oleh seorang user. Data diambil dari tabel listens yang berisi catatan setiap kali user memutar lagu. Function ini menampilkan judul lagu, durasi pendengaran (dalam detik), dan timestamp pemutaran terakhir, kemudian mengurutkannya dari yang paling baru.</p> <p>Fitur yang disediakan:</p> <ul style="list-style-type: none"> Mendapatkan riwayat pemutaran berdasarkan user_id. Mengurutkan berdasarkan waktu pemutaran (TIMESTAMP) secara descending. Limit output menggunakan parameter p_limit. Mengembalikan data dalam bentuk TABLE.
Script SQL	<pre>CREATE OR REPLACE FUNCTION get_recently_played(p_user_id INT, -- Parameter input ID user p_limit INT DEFAULT 10 -- Limit jumlah hasil yang ditampilkan)</pre>

```

RETURNS TABLE (
    song_id INT, -- Mengambil ID lagu dari tabel songs
    song_title VARCHAR, -- Mengambil judul lagu dari tabel songs
    duration_listened INT, -- Durasi mendengarkan dalam bentuk
detik
    last_play TIMESTAMP -- Timestamp kapan lagu diputar
)
LANGUAGE plpgsql
AS $$

BEGIN
    RETURN QUERY
    SELECT
        s.song_id,
        s.song_title,
        l.duration_listened, -- dalam detik
        l."TIMESTAMP" AS last_play
    FROM listens l
    JOIN songs s ON s.song_id = l.song_id -- Join tabel listens
dengan songs berdasarkan song_id
    WHERE l.user_id = p_user_id -- Memfilter hanya record yang
sesuai user yang diminta
    ORDER BY l."TIMESTAMP" DESC -- Mengurutkan dari pemutaran
terbaru
    LIMIT p_limit; -- Membatasi jumlah record sesuai parameter
p_limit
END;
$$;

```

Screenshot Hasil

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- > rate_songs
- > releases
- > reviews
- > socials
- > songs
 - > Columns(13)
 - song_id
 - song_file
 - song_title
 - listen_count
 - song_credits
 - song_duration
 - valence
 - acousticness
 - danceability
 - energy
 - popularity
 - song_release_date
 - song_rating
 - > Constraints
 - > Indexes
 - > RLS Policies
 - > Rules
 - > Triggers
 - > songs_genres

Query Query History

```

4313 -- Function get_recently_played
4314 CREATE OR REPLACE FUNCTION get_recently_played(
4315   p_user_id INT,
4316   p_limit INT DEFAULT 10
4317 )
4318   RETURNS TABLE (
4319     song_id INT,
4320     song_title VARCHAR,
4321     duration_listened INT,
4322     last_play TIMESTAMP
4323   )
4324 LANGUAGE plpgsql
4325 AS $$
```

Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 60 msec.

Total rows: Query complete 00:00:00.060

✓ Query returned successfully in 60 msec. CRLF Ln 4314, Col 1

21:31 25/11/2025

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- > rate_songs
- > releases
- > reviews
- > socials
- > songs
 - > Columns(13)
 - song_id
 - song_file
 - song_title
 - listen_count
 - song_credits
 - song_duration
 - valence
 - acousticness
 - danceability
 - energy
 - popularity
 - song_release_date
 - song_rating
 - > Constraints
 - > Indexes
 - > RLS Policies
 - > Rules
 - > Triggers
 - > songs_genres

Query Query History

```

4339 END;
4340 $$;
4341
4342 -- cek apa user punya history
4343 SELECT * FROM listens WHERE user_id = 1;
4344
4345 -- coba jalankan function
4346 SELECT * FROM get_recently_played(1, 5);
```

Data Output Messages Notifications

listen_id	[PK] integer	user_id	integer	song_id	integer	duration_listened	TIMESTAMP
1	1	1	120	4	2025-10-23 04:52:43.384135		
2	2	1	47	212	2025-10-29 20:03:49.120694		
3	3	1	89	313	2025-11-10 08:27:07.314559		
4	4	1	51	287	2025-11-11 17:41:33.924838		
5	5	1	65	333	2025-10-25 10:28:12.785141		
6	6	1	94	338			
7	7	1	87	81			

Showing rows: 1 to 16 Page No: 1 of 1 | < << << << >> >> >>> >>>

Total rows: 16 Query complete 00:00:00.291

✓ Successfully run. Total query runtime: 291 msec. 16 rows affected. CRLF Ln 4342, Col 1

21:32 25/11/2025

```

SELECT * FROM listens WHERE user_id = 1;
-- coba jalankan function
SELECT * FROM get_recently_played(1, 5);

```

	song_id	song_title	duration_listened	last_play
1	101	Officis doloremque	300	2025-11-25 20:49:38.461843
2	125	Adipisci illum quisquam quisquam	60	2025-11-16 01:27:30.209688
3	51	Voluptatum repellat voluptas voluptates	287	2025-11-11 17:41:33.924838
4	89	Et cupiditate totam exercitationem	313	2025-11-10 08:27:07.314559
5	115	Facilis dolore vero	305	2025-11-09 15:17:57.541681

Total rows: 5 Query complete 00:00:00.092 CRLF Ln 4347, Col 1

-- jika di ganti jadi 10 riwayat lagu terakhir

```

END;
$$;
-- cek apa user punya history
SELECT * FROM listens WHERE user_id = 1;
-- coba jalankan function
SELECT * FROM get_recently_played(1, 10);

```

	song_id	song_title	duration_listened	last_play
4	89	Et cupiditate totam exercitationem	313	2025-11-10 08:27:07.314559
5	115	Facilis dolore vero	305	2025-11-09 15:17:57.541681
6	64	Eaque distinctio nemo nulla adipisci	191	2025-11-09 11:56:55.9031
7	110	Eveniet id atque voluptatem labore	129	2025-11-02 08:17:53.66961
8	100	Quidem enim eligendi	11	2025-11-01 19:30:03.92721
9	87	Magnam nostrum iusto sed	81	2025-10-31 04:45:05.921962
10	47	Ipsum asperiores consequuntur facilis		

Total rows: 10 Query complete 00:00:00.218 CRLF Ln 4346, Col 1

Tabel 78 PL/SQL – Function get_song_detail

Nama Function	get_song_detail
Parameter	p_song_id IN NUMBER
Deskripsi	Function ini digunakan untuk mengambil metadata lengkap sebuah lagu berdasarkan song_id. Data yang dikembalikan mencakup informasi lagu, artis pembuat lagu, genre, serta

album/collection tempat lagu berada. Function menggabungkan beberapa tabel: songs, create_songs, artists, songs_genres, genres, collections_songs, dan collections. Output berupa satu baris per kombinasi lagu–genre–collection.

Script SQL

```

CREATE OR REPLACE FUNCTION get_song_detail(p_song_id INT)
RETURNS TABLE (
    song_id INT,
    song_title VARCHAR,
    song_duration INT,
    song_release_date DATE,
    song_rating NUMERIC,
    artist_name VARCHAR,
    genre_name VARCHAR,
    collection_title VARCHAR,
    nomor_disc INT,
    nomor_track INT
)
AS $$ 
BEGIN
    RETURN QUERY
    SELECT
        -- Ambil ID lagu, judul lagu, durasi lagu, tanggal rilis
        lagu, rating lagu
        s.song_id,
        s.song_title,
        s.song_duration,
        s.song_release_date,
        s.song_rating,

        a.artist_name,          -- artis dari CREATE_SONGS)
        g.genre_name,           -- genre lagu dari
        SONGS_GENRES)          

        c.collection_title,     -- dari collection
        cs.nomor_disc,          -- nomor disc lagu dalam
        collection               -- nomor track lagu dalam
        collection

        FROM SONGS s

        -- ARTIST
    
```

```
LEFT JOIN CREATE_SONGS cr ON s.song_id = cr.song_id -- Join
untuk dapat artist_id dari CREATE_SONGS
LEFT JOIN ARTISTS a ON cr.artist_id = a.artist_id

-- GENRE
LEFT JOIN SONGS_GENRES sg ON s.song_id = sg.song_id
LEFT JOIN GENRES g ON sg.genre_id = g.genre_id

-- COLLECTION / ALBUM
LEFT JOIN COLLECTIONS_SONGS cs ON s.song_id = cs.song_id
LEFT JOIN COLLECTIONS c ON cs.collection_id = c.collection_id

WHERE s.song_id = p_song_id; -- Filter hanya lagu dengan ID
sesuai parameter
END;
$$ LANGUAGE plpgsql;
```

Screenshot Hasil

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

```

artist_pfp
artist_email
banner
follower_count
  > Constraints
  > Indexes
  > RLS Policies
  > Rules
  > Triggers
  > artists_tours
  > block_users
  > blocklist_artists
  > collection_library
  > collection_top_3_genres
  > collections
    > Columns (8)
      collection_id
      collection_title
      collection_type
      collection_cover
      collection_release_date
      collection_rating
      isprelease
      collection_genre

```

Query Query History

```

4044
4045
4046
4047
4048
4049
4050
4051
4052
4053
4054
4055
-- cek lihat lagu-lagu di playlist dengan ID = 1
SELECT *
FROM get_playlist_tracks(1);

-- Function get_song_detail
CREATE OR REPLACE FUNCTION get_song_detail(p_song_id IN
RETURNS TABLE (
song_id INT,
song_title VARCHAR,

```

Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 321 msec.

Total rows: 1 Query complete 00:00:00.321 CRLF Ln 4051, Col 1 19:47 24/11/2025

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

```

artist_pfp
artist_email
banner
follower_count
  > Constraints
  > Indexes
  > RLS Policies
  > Rules
  > Triggers
  > artists_tours
  > block_users
  > blocklist_artists
  > collection_library
  > collection_top_3_genres
  > collections
    > Columns (8)
      collection_id
      collection_title
      collection_type
      collection_cover
      collection_release_date
      collection_rating
      isprelease
      collection_genre

```

Query Query History

```

4092
4093
4094
4095
4096
4097
4098
4099
4100
4101
4102
4103
-- COLLECTION / ALBUM
LEFT JOIN COLLECTIONS_SONGS cs ON s.song_id = cs.song_id
LEFT JOIN COLLECTIONS c ON cs.collection_id = c.collection_id
WHERE s.song_id = p_song_id;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM get_song_detail(1);

```

Data Output Messages Notifications

song_id	song_title	song_duration	song_release_date	song_rating	artist_name	genre_name
1	Officis eum labore corrupti aperiam	163	2007-12-16	4	Drs. Endah Hakim, M.TI.	Jazz

Successfully run. Total query runtime: 112 msec. 1 rows affected.

Total rows: 1 Query complete 00:00:00.112 CRLF Ln 4100, Col 1 19:47 24/11/2025

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

```

artist_pfp
artist_email
banner
follower_count
  > Constraints
  > Indexes
  > RLS Policies
  > Rules
  > Triggers
  > artists_tours
  > block_users
  > blocklist_artists
  > collection_library
  > collection_top_3_genres
  > collections
    > Columns (8)
      collection_id
      collection_title
      collection_type
      collection_cover
      collection_release_date
      collection_rating
      isprelease
      collection_genre

```

Query Query History

```

4092
4093
4094
4095
4096
4097
4098
4099
4100
4101
4102
4103
-- COLLECTION / ALBUM
LEFT JOIN COLLECTIONS_SONGS cs ON s.song_id = cs.song_id
LEFT JOIN COLLECTIONS c ON cs.collection_id = c.collection_id
WHERE s.song_id = p_song_id;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM get_song_detail(1);

```

Data Output Messages Notifications

song_release_date	song_rating	artist_name	genre_name	collection_title	nomor_disc	nomor_track
2007-12-16	4	Drs. Endah Hakim, M.TI.	Jazz	Inventore suscipit	1	1

Total rows: 1 Query complete 00:00:00.112 CRLF Ln 4100, Col 1 19:47 24/11/2025

Tabel 79 PL/SQL – Function get_playlist_tracks

Nama Function	get_playlist_tracks
Parameter	p_playlist_id IN INT
Deskripsi	Mengambil semua lagu yang ada di suatu playlist berdasarkan playlist_id. Menampilkan detail lagu dari tabel SONGS dan informasi terkait koleksi (album) dari tabel COLLECTIONS_SONGS. Nomor urut lagu mengikuti urutan penambahan di playlist (NO_URUT dari ADD_SONGS_PLAYLISTS).
Script SQL	
<pre> CREATE OR REPLACE FUNCTION get_playlist_tracks(p_playlist_id INT) RETURNS TABLE (song_no INT, song_id INT, song_title VARCHAR, song_duration INT, collection_id INT) AS \$\$ BEGIN RETURN QUERY SELECT asp.no_urut AS song_no, -- nomor urut lagu di playlist s.song_id, s.song_title, s.song_duration, cs.collection_id -- null jika lagu tidak ada di collection FROM ADD_SONGS_PLAYLISTS asp -- Ambil tabel lagu yang ditambahkan ke playlist (alias asp) JOIN SONGS s ON asp.song_id = s.song_id LEFT JOIN COLLECTIONS_SONGS cs ON s.song_id = cs.song_id -- cek lagu di collection mana WHERE asp.playlist_id = p_playlist_id -- ambil lagu dari playlist tertentu ORDER BY asp.no_urut; -- Urut sesuai nomor urut di playlist </pre>	

```

END;
$$ LANGUAGE plpgsql;

-- cek lihat lagu-lagu di playlist dengan ID = 1
SELECT *
FROM get_playlist_tracks(1);

```

Screenshot Hasil

The screenshot displays two pgAdmin 4 sessions. The top session shows the creation of the `get_playlist_tracks` function:

```

3996 $$ LANGUAGE plpgsql;
3997
3998 -- cek detail playlist id 1
3999 SELECT * FROM get_playlist_detail(1);
4000
4001
4002 -- Function: get_playlist_tracks
4003 CREATE OR REPLACE FUNCTION get_playlist_tracks(p_playlist_id INT)
4004 RETURNS TABLE (
4005   song_no INT,
4006   song_id INT,
4007   song_title VARCHAR,
4008   song_duration INT
4009 )
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4020
4021
4022
4023 WHERE asp.playlist_id = p_playlist_id -- ambil lagu dari play-
4024 ORDER BY asp.no_urut; -- Urut sesuai nomor urut di playlist
4025
4026 END;
4027
4028 $$ LANGUAGE plpgsql;
4029
4030 -- cek lihat lagu-lagu di playlist dengan ID = 1
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4050
4051
4052
4053
4054
4055
4056
4057
4058
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168
4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4179
4180
4181
4182
4183
4184
4185
4186
4187
4188
4189
4190
4191
4192
4193
4194
4195
4196
4197
4198
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4230
4231
4232
4233
4234
4235
4236
4237
4238
4239
4240
4241
4242
4243
4244
4245
4246
4247
4248
4249
4250
4251
4252
4253
4254
4255
4256
4257
4258
4259
4260
4261
4262
4263
4264
4265
4266
4267
4268
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4290
4291
4292
4293
4294
4295
4296
4297
4298
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308
4309
4310
4311
4312
4313
4314
4315
4316
4317
4318
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329
4330
4331
4332
4333
4334
4335
4336
4337
4338
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4360
4361
4362
4363
4364
4365
4366
4367
4368
4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4420
4421
4422
4423
4424
4425
4426
4427
4428
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4460
4461
4462
4463
4464
4465
4466
4467
4468
4469
4470
4471
4472
4473
4474
4475
4476
4477
4478
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4490
4491
4492
4493
4494
4495
4496
4497
4498
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4520
4521
4522
4523
4524
4525
4526
4527
4528
4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4540
4541
4542
4543
4544
4545
4546
4547
4548
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4599
4600
4601
4602
4603
4604
4605
4606
4607
4608
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689
4690
4691
4692
4693
4694
4695
4696
4697
4698
4699
4700
4701
4702
4703
4704
4705
4706
4707
4708
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4720
4721
4722
4723
4724
4725
4726
4727
4728
4729
4730
4731
4732
4733
4734
4735
4736
4737
4738
4739
4740
4741
4742
4743
4744
4745
4746
4747
4748
4749
4750
4751
4752
4753
4754
4755
4756
4757
4758
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4829
4830
4831
4832
4833
4834
4835
4836
4837
4838
4839
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859
4859
4860
4861
4862
4863
4864
4865
4866
4867
4868
4869
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4889
4890
4891
4892
4893
4894
4895
4896
4897
4898
4899
4899
4900
4901
4902
4903
4904
4905
4906
4907
4908
4909
4909
4910
4911
4912
4913
4914
4915
4916
4917
4918
4919
4919
4920
4921
4922
4923
4924
4925
4926
4927
4928
4929
4929
4930
4931
4932
4933
4934
4935
4936
4937
4938
4939
4939
4940
4941
4942
4943
4944
4945
4946
4947
4948
4949
4949
4950
4951
4952
4953
4954
4955
4956
4957
4958
4959
4959
4960
4961
4962
4963
4964
4965
4966
4967
4968
4969
4969
4970
4971
4972
4973
4974
4975
4976
4977
4978
4979
4979
4980
4981
4982
4983
4984
4985
4986
4987
4988
4989
4989
4990
4991
4992
4993
4994
4995
4996
4997
4998
4999
4999
5000
5001
5002
5003
5004
5005
5006
5007
5008
5009
5009
5010
5011
5012
5013
5014
5015
5016
5017
5018
5019
5019
5020
5021
5022
5023
5024
5025
5026
5027
5028
5029
5029
5030
5031
5032
5033
5034
5035
5036
5037
5038
5039
5039
5040
5041
5042
5043
5044
5045
5046
5047
5048
5049
5049
5050
5051
5052
5053
5054
5055
5056
5057
5058
5059
5059
5060
5061
5062
5063
5064
5065
5066
5067
5068
5069
5069
5070
5071
5072
5073
5074
5075
5076
5077
5078
5079
5079
5080
5081
5082
5083
5084
5085
5086
5087
5088
5089
5089
5090
5091
5092
5093
5094
5095
5096
5097
5098
5098
5099
5099
5100
5101
5102
5103
5104
5105
5106
5107
5108
5109
5109
5110
5111
5112
5113
5114
5115
5116
5117
5118
5119
5119
5120
5121
5122
5123
5124
5125
5126
5127
5128
5129
5129
5130
5131
5132
5133
5134
5135
5136
5137
5138
5139
5139
5140
5141
5142
5143
5144
5145
5146
5147
5148
5149
5149
5150
5151
5152
5153
5154
5155
5156
5157
5158
5159
5159
5160
5161
5162
5163
5164
5165
5166
5167
5168
5169
5169
5170
5171
5172
5173
5174
5175
5176
5177
5178
5179
5179
5180
5181
5182
5183
5184
5185
5186
5187
5188
5189
5189
5190
5191
5192
5193
5194
5195
5196
5197
5198
5198
5199
5199
5200
5201
5202
5203
5204
5205
5206
5207
5208
5209
5209
5210
5211
5212
5213
5214
5215
5216
5217
5218
5219
5219
5220
5221
5222
5223
5224
5225
5226
5227
5228
5229
5229
5230
5231
5232
5233
5234
5235
5236
5237
5238
5239
5239
5240
5241
5242
5243
5244
5245
5246
5247
5248
5249
5249
5250
5251
5252
5253
5254
5255
5256
5257
5258
5259
5259
5260
5261
5262
5263
5264
5265
5266
5267
5268
5269
5269
5270
5271
5272
5273
5274
5275
5276
5277
5278
5278
5279
5279
5280
5281
5282
5283
5284
5285
5286
5287
5288
5289
5289
5290
5291
5292
5293
5294
5295
5296
5297
5297
5298
5299
5299
5300
5301
5302
5303
5304
5305
5306
5307
5308
5309
5309
5310
5311
5312
5313
5314
5315
5316
5317
5318
5319
5319
5320
5321
5322
5323
5324
5325
5326
5327
5328
5329
5329
5330
5331
5332
5333
5334
5335
5336
5337
5338
5339
5339
5340
5341
5342
5343
5344
5345
5346
5347
5348
5349
5349
5350
5351
5352
5353
5354
5355
5356
5357
5358
5359
5359
5360
5361
5362
5363
5364
5365
5366
5367
5368
5369
5369
5370
5371
5372
5373
5374
5375
5376
5377
5378
5378
5379
5379
5380
5381
5382
5383
5384
5385
5386
5387
5388
5389
5389
5390
5391
5392
5393
5394
5395
5396
5397
5397
5398
5399
5399
5400
5401
5402
5403
5404
5405
5406
5407
5408
5409
5409
5410
5411
5412
5413
5414
5415
5416
5417
5418
5419
5419
5420
5421
5422
5423
5424
5425
5426
5427
5428
5429
5429
5430
5431
5432
5433
5434
5435
5436
5437
5438
5439
5439
5440
5441
5442
5443
5444
5445
5446
5447
5448
5449
5449
5450
5451
5452
5453
5454
5455
5456
5457
5458
5459
5459
5460
5461
5462
5463
5464
5465
5466
5467
5468
5469
5469
5470
5471
5472
5473
5474
5475
5476
5477
5478
5478
5479
5479
5480
5481
5482
5483
5484
5485
5486
5487
5488
5489
5489
5490
5491
5492
5493
5494
5495
5496
5497
5497
5498
5499
5499
5500
5501
5502
5503
5504
5505
5506
5507
5508
5509
5509
5510
5511
5512
5513
5514
5515
5516
5517
5518
5519
5519
5520
5521
5522
5523
5524
5525
5526
5527
5528
5529
5529
5530
5531
5532
5533
5534
5535
5536
5537
5538
5539
5539
5540
5541
5542
5543
5544
5545
5546
5547
5548
5549
5549
5550
5551
5552
5553
5554
5555
5556
5557
5558
5559
5559
5560
5561
5562
5563
5564
5565
5566
5567
5568
5569
5569
5570
5571
5572
5573
5574
5575
5576
5577
5578
5578
5579
5579
5580
5581
5582
5583
5584
5585
5586
5587
5588
5589
5589
5590
5591
5592
5593
5594
5595
5596
5597
5597
5598
5599
5599
5600
5601
5602
5603
5604
5605
5606
5607
5608
5609
5609
5610
5611
5612
5613
5614
5615
5616
5617
5618
5619
5619
5620
5621
5622
5623
5624
5625
5626
5627
5628
5629
5629
5630
5631
5632
5633
5634
5635
5636
5637
5638
5639
5639
5640
5641
5642
5643
5644
5645
5646
5647
5648
5649
5649
5650
5651
5652
5653
5654
5655
5656
5657
5658
5659
5659
5660
5661
5662
5663
5664
5665
5666
5667
5668
5669
5669
5670
5671
5672
5673
5674
5675
5676
5677
5678
5678
5679
5679
5680
5681
5682
5683
5684
5685
5686
5687
5688
5689
5689
5690
5691
5692
5693
5694
5695
5696
5697
5698
5698
5699
5699
5700
5701
5702
5703
5704
5705
5706
5707
5708
5709
5709
5710
5711
5712
5713
5714
5715
5716
5717
5718
5719
5719
5720
5721
5722
5723
5724
5725
5726
5727
5728
5729
5729
5730
5731
5732
5733
5734
5735
5736
5737
5738
5739
5739
5740
5741
5742
5743
5744
5745
5746
5747
5748
5749
5749
5750
5751
5752
5753
5754
5755
5756
5757
5758
5759
5759
5760
5761
5762
5763
5764
5765
5766
5767
5768
5769
5769
5770
5771
5772
5773
5774
5775
5776
5777
5778
5778
5779
5779
5780
5781
5782
5783
5784
5785
5786
5787
5788
5789
5789
5790
5791
5792
5793
5794
5795
5796
5797
5797
5798
5799
5799
5800
5801
5802
5803
5804
5805
5806
5807
5808
5809
5809
5810
5811
5812
5813
5814
5815
5816
5817
5818
5819
5819
5820
5821
5822
5823
5824
5825
5826
5827
5828
5829
5829
5830
5831
5832
5833
5834
5835
5836
5837
5838
5839
5839
5840
5841
5842
5843
5844
5845
5846
5847
5848
5849
5849
5850
5851
5852
5853
5854
5855
5856
5857
5858
5859
5859
5860
5861
5862
5863
5864
5865
5866
5867
5868
5869
5869
5870
5871
5872
5873
5874
5875
5876
5877
5878
5878
5879
5879
5880
5881
5882
5883
5884
5885
5886
5887
5888
5889
5889
5890
5891
5892
5893
5894
5895
5896
5897
5898
5898
5899
5899
5900
5901
5902
5903
5904
5905
5906
5907
5908
5909
5909
5910
5911
5912
5913
5914
5915
5916
5917
5918
5919
5919
5920
5921
5922
5923
5924
5925
5926
5927
5928
5929
5929
5930
5931
5932
5933
5934
5935
5936
5937
5938
5939
5939
5940
5941
5942
5943
5944
5945
5946
5947
5948
5949
5949
5950
5951
5952
5953
5954
5955
5956
5957
5958
5959
5959
5960
5961
5962
5963
5964
5965
5966
5967
5968
5969
5969
5970
5971
5972
5973
5974
5975
5976
5977
5978
5978
5979
5979
5980
5981
5982
5983
5984
5985
5986
5987
5988
5989
5989
5990
5991
5992
5993
5994
5995
5996
5997
5998
5998
5999
5999
6000
6001
6002
6003
6004
6005
6006
6007
6008
6009
6009
6010
6011
6012
6013
6014
6015
6016
6017
6018
6019
6019
6020
6021
6022
6023
```

Deskripsi	Function ini mengembalikan detail dari sebuah playlist, termasuk informasi playlist itu sendiri, daftar lagu di dalamnya, nomor urut lagu, durasi lagu, dan username siapa yang menambahkan setiap lagu. Function mengembalikan data dalam bentuk tabel (set of rows).
Script SQL	
<pre> CREATE OR REPLACE FUNCTION get_playlist_detail(p_playlist_id INT) RETURNS TABLE (playlist_id INT, playlist_title VARCHAR, playlist_cover VARCHAR, playlist_desc TEXT, song_no INT, song_id INT, song_title VARCHAR, song_duration INT, added_by_username VARCHAR) AS \$\$ BEGIN -- Ambil data playlist + daftar lagu + siapa yang menambahkan lagu RETURN QUERY SELECT p.playlist_id, p.playlist_title, p.playlist_cover, p.playlist_desc, asp.no_urut AS song_no, -- nomor urut lagu di playlist s.song_id, s.song_title, s.song_duration, u.username AS added_by_username -- username yang menambahkan lagu FROM PLAYLISTS p -- Join ke tabel add_songs_playlists untuk mendapatkan daftar lagu yang ada di playlist JOIN ADD_SONGS_PLAYLISTS asp ON p.playlist_id = asp.playlist_id JOIN SONGS s ON asp.song_id = s.song_id -- Join ke tabel songs untuk mengambil detail lagu JOIN USERS u ON asp.user_id = u.user_id -- Join ke tabel users untuk mengetahui siapa yang nambahin lagu </pre>	

```
    WHERE p.playlist_id = p_playlist_id -- Filter hanya playlist  
yang sesuai dengan parameter input  
    ORDER BY asp.no_urut; -- urutkan lagu sesuai nomor urut  
END;  
$$ LANGUAGE plpgsql;  
  
-- cek detail playlist id 1  
SELECT * FROM get_playlist_detail(1);
```

Screenshot Hasil

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- > RLS Policies
- > Rules
- > Triggers
- > genres
- > like_reviews
- > like_songs
- > listens
- > pl_library
- > playlists
 - > Columns(9)
 - playlist_id
 - user_id
 - playlist_cover
 - playlist_title
 - ispublic
 - iscollaborative
 - playlist_desc
 - isonprofile
 - playlist_date_created
 - > Constraints
 - > Indexes
 - > RLS Policies
 - > Rules
 - > Triggers
 - > rate_songs

Final_Destination/postgres@Postgres16

Query Query History

```

3961
3962 -- Function: get_playlist_detail
3963 CREATE OR REPLACE FUNCTION get_playlist_detail(p_playlist_id INT)
3964 RETURNS TABLE (
3965   playlist_id INT,
3966   playlist_title VARCHAR,
3967   playlist_cover VARCHAR,
3968   playlist_desc TEXT,
3969   song_no INT,
3970   song_id INT,
3971   song_title VARCHAR,
3972   song_duration INT,
3973   added_by_username VARCHAR
3974 )
3975 
```

Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 161 msec.

Total rows: 5 Query complete 00:00:00.161 CRLF Ln 3996, Col 21

21:20 23/11/2025

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- > RLS Policies
- > Rules
- > Triggers
- > genres
- > like_reviews
- > like_songs
- > listens
- > pl_library
- > playlists
 - > Columns(9)
 - playlist_id
 - user_id
 - playlist_cover
 - playlist_title
 - ispublic
 - iscollaborative
 - playlist_desc
 - isonprofile
 - playlist_date_created
 - > Constraints
 - > Indexes
 - > RLS Policies
 - > Rules
 - > Triggers
 - > rate_songs

Final_Destination/postgres@Postgres16

Query Query History

```

3991 JOIN SONGS s ON asp.song_id = s.song_id
3992 JOIN USERS u ON asp.user_id = u.user_id
3993 WHERE p.playlist_id = p_playlist_id
3994 ORDER BY asp.no_urut; -- urutkan lagu sesuai nomor urut
3995 END;
3996 $$ LANGUAGE plpgsql;
3997
3998 -- cek detail playlist id 1
3999 SELECT * FROM get_playlist_detail(1);
4000
4001
4002 
```

Data Output Messages Notifications

playlist_id	playlist_title	playlist_cover	playlist_desc	song_no	song_id	song_title
1	Playlist dignissimos Abu-abu collab	[null]	Playlist askit buat coding	1	8	Repellat pariatur repud
2	Playlist dignissimos Abu-abu collab	[null]	Playlist askit buat coding	2	106	Eligendi voluptatibus rr
3	Playlist dignissimos Abu-abu collab	[null]	Playlist askit buat coding	3	115	Facilis dolore vero

Showing rows: 1 to 5 Page No: 1 of 1

Successfully run. Total query runtime: 128 msec. 5 rows affected.

Total rows: 5 Query complete 00:00:00.128 CRLF Ln 3998, Col 1

21:20 23/11/2025

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- > RLS Policies
- > Rules
- > Triggers
- > genres
- > like_reviews
- > like_songs
- > listens
- > pl_library
- > playlists
 - > Columns(9)
 - playlist_id
 - user_id
 - playlist_cover
 - playlist_title
 - ispublic
 - iscollaborative
 - playlist_desc
 - isonprofile
 - playlist_date_created
 - > Constraints
 - > Indexes
 - > RLS Policies
 - > Rules
 - > Triggers
 - > rate_songs

Final_Destination/postgres@Postgres16

Query Query History

```

3994 ORDER BY asp.no_urut; -- urutkan lagu sesuai nomor urut
3995 END;
3996 $$ LANGUAGE plpgsql;
3997
3998 -- cek detail playlist id 1
3999 SELECT * FROM get_playlist_detail(1);
4000
4001 
```

Data Output Messages Notifications

playlist_id	playlist_title	playlist_cover	song_no	song_id	song_title	song_duration	added_by_username
1	Playlist askit buat coding	text	1	8	Repellat pariatur repudiandae qui	190	rahayusurya
2	Playlist askit buat coding	[null]	2	106	Eligendi voluptatibus molestiae laudantium	142	wkuswoyo
3	Playlist askit buat coding	[null]	3	115	Facilis dolore vero	178	bakianto78
4	Playlist askit buat coding	[null]	4	12	Alias excepturi quisquam consectetur fugiat	277	wkuswoyo
5	Playlist askit buat coding	[null]	5	23	Alias totam iusto	132	ediwahyuni

Showing rows: 1 to 5 Page No: 1 of 1

Total rows: 5 Query complete 00:00:00.128 CRLF Ln 3998, Col 1

21:20 23/11/2025

Tabel 81 PL/SQL – Procedure log_listen

Nama Procedure	Log_listen
Parameter	<p>p_user_id – ID user yang mendengarkan lagu</p> <p>p_song_id – ID lagu yang didengarkan</p> <p>p_duration – Durasi dalam detik user mendengarkan lagu</p>
Deskripsi	<p>log_listen procedure ini yang akan digunakan untuk mencatat histori mendengarkan lagu ke tabel LISTENS.</p> <p>Fungsionalitasnya adalah:</p> <ol style="list-style-type: none"> 1. Validasi bahwa user dan lagu benar-benar ada di tabel USERS dan SONGS. 2. Validasi durasi harus lebih dari 0. 3. Mengecek apakah user sudah pernah mendengarkan lagu tersebut. <ol style="list-style-type: none"> 1. Jika belum pernah, procedure melakukan: <ul style="list-style-type: none"> - INSERT baris baru ke tabel LISTENS - Menambahkan listen_count + 1 pada tabel SONGS 2. Jika sudah pernah, procedure hanya: <ul style="list-style-type: none"> - UPDATE duration terbaru - Mengupdate timestamp - Tidak menambah listen_count lagi <p>Dengan demikian penghitungan listen_count hanya dilakukan sekali</p>

	untuk setiap pasangan user-song, dan histori durasi terakhir tetap diperbarui pada pemutaran ulang.
Script SQL	
	<pre> CREATE OR REPLACE PROCEDURE log_listen(p_user_id INT, -- Parameter input: ID user yang mendengarkan lagu, ID lagu yang didengarkan, durasi lagu yang didengarkan (dalam detik) p_song_id INT, p_duration INT) LANGUAGE plpgsql AS \$\$ DECLARE v_exists_listen INT; -- Variabel untuk menyimpan jumlah record listens yang sudah ada untuk user & song BEGIN -- VALIDASI USER IF NOT EXISTS (SELECT 1 FROM users WHERE user_id = p_user_id) THEN RAISE EXCEPTION 'User % tidak ditemukan', p_user_id; END IF; -- VALIDASI SONG IF NOT EXISTS (SELECT 1 FROM songs WHERE song_id = p_song_id) THEN RAISE EXCEPTION 'Song % tidak ditemukan', p_song_id; END IF; -- VALIDASI DURASI IF p_duration <= 0 THEN RAISE EXCEPTION 'Duration harus > 0'; END IF; -- CEK APAKAH USER SUDAH PERNAH MENDENGARKAN LAGUINI SELECT COUNT(*) -- Hitung jumlah record listens untuk user & song INTO v_exists_listen -- Simpan hasil hitungan ke variabel v_exists_listen FROM listens WHERE user_id = p_user_id AND song_id = p_song_id; -- INSERT LISTEN BARU → listen_count + 1 </pre>

```

    IF v_exists_listen = 0 THEN -- Jika user belum pernah
mendengarkan lagu ini

        INSERT INTO listens (listen_id, user_id, song_id,
duration_listened)
        VALUES (
            (SELECT COALESCE(MAX(listen_id), 0) + 1 FROM listens),
-- akan generate listen_id baru secara increment
            p_user_id,
            p_song_id,
            p_duration
        );

        -- TAMBAH COUNTER LISTEN HANYA UNTUK INSERT
        UPDATE songs
        SET listen_count = COALESCE(listen_count, 0) + 1 -- Increment listen_count lagu
        WHERE song_id = p_song_id;

        -- UPDATE LISTEN LAMA → TIDAK MENAMBAH COUNTER
        ELSE

            UPDATE listens
            SET duration_listened = p_duration, -- Update durasi terakhir mendengarkan
                "TIMESTAMP" = CURRENT_TIMESTAMP -- Update timestamp terakhir
            WHERE user_id = p_user_id
            AND song_id = p_song_id;

        END IF;

END;
$$;

```

Screenshot Hasil

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

Final_Destination/postgres@Postgres16

Query Query History

```

4211 -- PROCEDURE log_listen
4212 CREATE OR REPLACE PROCEDURE log_listen(
4213     p_user_id INT,
4214     p_song_id INT,
4215     p_duration INT
4216 )
4217 LANGUAGE plpgsql
4218 AS $$
4219 DECLARE
4220     v_exists_listen INT;
4221 BEGIN

```

Data Output Messages Notifications

CREATE PROCEDURE

Query returned successfully in 95 msec.

Total rows: 0 Query complete 00:00:00.095 CRLF Ln 4272, Col 4

2037 25/11/2025

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

Final_Destination/postgres@Postgres16

Query Query History

```

4272 ;;
4273
4274
4275 -- cek
4276 -- Cek apakah USER & SONG tersedia
4277 SELECT * FROM users WHERE user_id = 1;
4278 SELECT * FROM songs WHERE song_id = 101;
4279
4280 -- insert
4281 CALL log_listen(1, 101, 120);
4282
4283 -- cek hasil insert

```

Data Output Messages Notifications

song_id	song_file	song_title	listen_count	song_credits	song_duration	valence
101	/uploads/audio/song_101.mp3	Officis doloremque	[null]	[null]	199	0.850

Showing rows: 1 to 1 Page No: 1 of 1 |<< << >> >>|

Total rows: 1 Query complete 00:00:00.274 CRLF Ln 4277, Col 1

2042 25/11/2025

-- insert

-- cek apakan user & song tersebut

```

4277    SELECT * FROM users WHERE user_id = 1;
4278    SELECT * FROM songs WHERE song_id = 101;

4279    -- insert
4280    CALL log_listen(1, 101, 120);

4283    -- cek hasil insert
4284    SELECT * FROM listens
4285    WHERE user_id = 1 AND song_id = 101;
4286

4287    -- cek listen count

```

Data Output Messages Notifications

CALL

Query returned successfully in 78 msec.

Total rows: Query complete 00:00:00.078 CRLF Ln 4287, Col 1

2047 25/11/2025

-- cek hasil insert

-- cek apakan user & song tersebut

```

4277    SELECT * FROM users WHERE user_id = 1;
4278    SELECT * FROM songs WHERE song_id = 101;

4279    -- insert
4280    CALL log_listen(1, 101, 120);

4283    -- cek hasil insert
4284    SELECT * FROM listens
4285    WHERE user_id = 1 AND song_id = 101;
4286

4287    -- cek listen count

```

Data Output Messages Notifications

listen_id	[PK] integer	user_id	integer	song_id	integer	duration_listened	integer	TIMESTAMP	timestamp without time zone
1	751	1	101		120	2025-11-25 20:47:21.38173			

Showing rows: 1 to 1 Page No: 1 of 1

Successfully run. Total query runtime: 792 msec. 1 rows affected. CRLF Ln 4284, Col 1

Total rows: 1 Query complete 00:00:00.792 CRLF Ln 4284, Col 1

2047 25/11/2025

-- cek listen count

pgAdmin 4

```

File Object Tools Edit View Window Help
Object Explorer
postgres@... x fix.sql* x FINAL_FIX.sql x Final Destination/... x TRIGGER1.sql x coba_coba_virli.sql* x
Query Query History
CALL log_listen(1, 101, 300);
-- cek hasil insert
SELECT * FROM listens
WHERE user_id = 1 AND song_id = 101;
-- cek listen count
SELECT listen_count
FROM songs
WHERE song_id = 101;
-- update listen

```

Data Output Messages Notifications

listen_count	bigint
1	2

Total rows: 1 Query complete 00:00:00.351 CRLF Ln 4287, Col 1

-- update listen

pgAdmin 4

```

File Object Tools Edit View Window Help
Object Explorer
postgres@... x fix.sql* x FINAL_FIX.sql x Final Destination/... x TRIGGER1.sql x coba_coba_virli.sql* x
Query Query History
-- cek listen count
SELECT listen_count
FROM songs
WHERE song_id = 101;
-- update listen
CALL log_listen(1, 101, 300);
-- cek listen lagi
SELECT * FROM listens
WHERE user_id = 1 AND song_id = 101;

```

Data Output Messages Notifications

CALL

Query returned successfully in 87 msec.

Total rows: Query complete 00:00:00.087 CRLF Ln 4293, Col 1

The screenshot shows the pgAdmin 4 interface. In the Object Explorer, under the 'Trigger Functions' section, there are nine functions listed. The current function selected is 'fix_playlist_collaborators()'. The Query tab contains a SQL script with several comments and code snippets. The Data Output tab shows a single row of data from a table named 'listens'. The status bar at the bottom right indicates a successful run with a runtime of 81 msec.

```

-- update listen
CALL log_listen(1, 101, 300);

-- cek listen lagi
SELECT * FROM listens
WHERE user_id = 1 AND song_id = 101;

-- cek listen count lagi
SELECT listen_count
FROM songs
WHERE song_id = 101;

```

listen_id	user_id	song_id	duration_listened	TIMESTAMP	
1	751	1	101	300	2025-11-25 20:49:38.461843

Total rows: 1 Query complete 00:00:00.081 CRLF Ln 4295, Col 1

Tabel 82 PL/SQL – Procedure rate_song

Nama Procedure	rate_song
Parameter	p_user_id INT, p_song_id INT, p_rating NUMERIC
Deskripsi	Procedure rate_song digunakan untuk memberikan atau memperbarui rating pada sebuah lagu oleh user. Logika utamanya: Mengecek apakah user sudah pernah melakukan rating pada lagu tersebut. Jika belum, maka dilakukan INSERT rating baru ke tabel rate_songs. Jika sudah, maka dilakukan UPDATE rating beserta timestamp-nya.
Script SQL	<pre> CREATE OR REPLACE PROCEDURE rate_song(p_user_id INT, -- Parameter input: ID user yang memberi rating p_song_id INT, -- Parameter input: ID lagu yang diberi rating p_rating NUMERIC -- Parameter input: Rating yang diberikan) BEGIN IF NOT EXISTS (SELECT * FROM listens WHERE user_id = p_user_id AND song_id = p_song_id) THEN INSERT INTO listens (user_id, song_id, duration_listened, timestamp) VALUES (p_user_id, p_song_id, p_rating, now()); ELSE UPDATE listens SET duration_listened = p_rating, timestamp = now() WHERE user_id = p_user_id AND song_id = p_song_id; END IF; END; </pre>

```

    p_song_id INT, -- Parameter input: ID lagu yang dirating
    p_rating NUMERIC -- Parameter input: nilai rating lagu
)
LANGUAGE plpgsql
AS $$

BEGIN
    -- Cek apakah user sudah pernah memberi rating
    IF EXISTS (
        SELECT 1 FROM rate_songs
        WHERE user_id = p_user_id
        AND song_id = p_song_id
    ) THEN
        -- Kalo sudah ada rating → update rating lama dan timestamp
        UPDATE rate_songs
        SET song_rating = p_rating, -- Set rating baru
            "TIMESTAMP" = CURRENT_TIMESTAMP -- Update waktu rating terakhir
        WHERE user_id = p_user_id
        AND song_id = p_song_id; -- update record user & lagu
    ELSE
        -- Jika belum ada rating → insert rating baru
        INSERT INTO rate_songs (user_id, song_id, song_rating)
        VALUES (p_user_id, p_song_id, p_rating); -- Masukkan record baru
    END IF;
END;
$$;

```

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface with the 'Object Explorer' on the left and a 'Query' editor on the right. In the Object Explorer, under the 'rate_songs' table, the 'Columns(4)' section is selected, showing columns: user_id, song_id, song_rating, and TIMESTAMP. In the Query editor, the following SQL code is written:

```

-- Procedure rate_song
CREATE OR REPLACE PROCEDURE rate_song(
    p_user_id INT,
    p_song_id INT,
    p_rating NUMERIC
)
LANGUAGE plpgsql
AS $$
```

Below the code, the message 'CREATE PROCEDURE' is displayed, followed by 'Query returned successfully in 146 msec.' At the bottom of the pgAdmin window, a status bar shows 'Total rows: 0' and 'Query complete 00:00:00.146'. A green notification bar at the bottom right says '✓ Query returned successfully in 146 msec. CRLF Ln 4164, Col 1'.

-- sebelum cek

-- cek hasil

```

4228 -- cek hasil
4229 SELECT * FROM RATE_SONGS WHERE song_id = 10;
4230
4231 SELECT song_id, song_title, song_rating
4232 FROM SONGS WHERE song_id = 10;
4233
4234

```

Total rows: 2 Query complete 00:00:00.081

22:28 24/11/2025

-- cek

```

4213
4214
4215
4216
4217
4218
4219
4220
4221
4222
4223
4224

```

FROM RATE_SONGS
WHERE song_id = p_song_id
)
WHERE song_id = p_song_id;
END;
\$\$;
-- Cek insert rating
CALL rate_song(1, 10, 5);
-- Update rating

Query returned successfully in 73 msec.

Total rows: 0 Query complete 00:00:00.073

22:28 24/11/2025

```

4223
4224
4225
4226
4227
4228
4229
4230
4231
4232
4233
4234

```

-- Update rating
CALL rate_song(1, 10, 3);
-- cek hasil
SELECT * FROM RATE_SONGS WHERE song_id = 10;
SELECT song_id, song_title, song_rating
FROM SONGS WHERE song_id = 10;
SELECT * FROM RATE_SONGS;

Total rows: 3 Query complete 00:00:00.082

22:28 24/11/2025

-- update rating

```

4195
4196
4197 -- Cek insert rating
4198 CALL rate_song(1, 10, 5);
4199
4200 -- Update rating
4201 CALL rate_song(1, 10, 3);
4202
4203 -- cek hasil
4204 SELECT * FROM RATE_SONGS WHERE song_id = 10;
4205
4206 SELECT song_id, song_title, song_rating

```

CALL

Query returned successfully in 583 msec.

Total rows: Query complete 00:00:00.583 CRLF Ln 4200, Col 1


```

4195
4196
4197 -- Cek insert rating
4198 CALL rate_song(1, 10, 5);
4199
4200 -- Update rating
4201 CALL rate_song(1, 10, 3);
4202
4203 -- cek hasil
4204 SELECT * FROM RATE_SONGS WHERE song_id = 10;
4205
4206 SELECT song_id, song_title, song_rating

```

Data Output Messages Notifications

user_id	song_id	song_rating	TIMESTAMP
[PK] integer	[PK] integer	numeric (3)	timestamp without time zone
1	1	10	3 2025-11-24 22:39:33.421113

Showing rows: 1 to 1 Page No: 1 of 1 14 44 >> >>

Successfully run. Total query runtime: 161 msec. 1 rows affected. CRLF Ln 4204, Col 1

Total rows: 1 Query complete 00:00:00.161 CRLF Ln 4204, Col 1

Tabel 83 PL/SQL – Procedure toggle_like_song

Nama Procedure	Toggle_like_song
Parameter	p_user_id INT p_user_id INT
Deskripsi	Procedure toggle_like_song digunakan untuk melakukan fitur LIKE/UNLIKE lagu oleh seorang user dalam satu perintah.

	<p>Jika user belum like lagu → sistem akan INSERT ke tabel LIKE_SONGS</p> <p>Jika user sudah like lagu → sistem akan DELETE dari tabel LIKE_SONGS</p> <p>Procedure ini juga melakukan validasi apakah user dan song benar-benar ada pada tabel USERS dan SONGS.</p>
Script SQL	
	<pre> CREATE OR REPLACE PROCEDURE toggle_like_song(p_user_id INT, -- Parameter input: ID user yang ingin like/unlike lagu p_song_id INT -- Parameter input: ID lagu yang ingin di like/unlike) LANGUAGE plpgsql AS \$\$ DECLARE v_exists INT; -- Variabel untuk mengecek record like_songs sudah ada apa belum BEGIN -- 1. Cek apakah user ada IF NOT EXISTS (SELECT 1 FROM users WHERE user_id = p_user_id) THEN RAISE EXCEPTION 'User dengan ID % tidak ditemukan', p_user_id; END IF; -- 2. Cek apakah lagu ada IF NOT EXISTS (SELECT 1 FROM songs WHERE song_id = p_song_id) THEN RAISE EXCEPTION 'Song dengan ID % tidak ditemukan', p_song_id; END IF; -- 3. Cek apakah sudah like SELECT COUNT(*) INTO v_exists FROM like_songs WHERE user_id = p_user_id AND song_id = p_song_id; -- 4. Jika sudah LIKE → UNLIKE IF v_exists > 0 THEN DELETE FROM like_songs -- Hapus record like (unlike) WHERE user_id = p_user_id ELSE INSERT INTO like_songs (user_id, song_id) VALUES (p_user_id, p_song_id); END IF; END; </pre>

```

        AND song_id = p_song_id;
        RAISE NOTICE 'UNLIKE berhasil untuk song_id=% oleh
user_id=%',
        p_song_id, p_user_id;
-- 5. Jika belum LIKE → LIKE
ELSE
    INSERT INTO like_songs (song_id, user_id) -- Tambah record
like baru
        VALUES (p_song_id, p_user_id);
    RAISE NOTICE 'LIKE berhasil untuk song_id=% oleh
user_id=%',
        p_song_id, p_user_id;
END IF;
END;
$$;

-- cek like
CALL toggle_like_song(1,10);

```

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows various database objects like collections, users, and trigger functions.
- Query Editor:** Contains the SQL code for creating the 'toggle_like_song' procedure.
- Status Bar:** Shows "Query returned successfully in 1 secs 59 msec." and "Total rows: 0".
- System Tray:** Shows system icons including a battery level of 21% and a date/time of 24/11/2023 21:52.

```

-- PROCEDURE: toggle_like_song
CREATE OR REPLACE PROCEDURE toggle_like_song(
    p_user_id INT,
    p_song_id INT
)
LANGUAGE plpgsql
AS $$

DECLARE
    v_exists INT;
BEGIN
    -- 1. Cek apakah user ada
    IF NOT EXISTS (SELECT 1 FROM users WHERE user_id = p_user_id) THEN
        RAISE EXCEPTION 'User dengan ID % tidak ditemukan', p_user_id;
    END IF;

```

-- cek like

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows database schema with tables like collection_top_3_genres, collections, collections_songs, etc.
- Query Editor:** Contains the following SQL code:

```
4147
4148     END;
4149     $$;
4150
4151     -- cek like
4152     CALL toggle_like_song(1, 10);
4153
4154     -- cek hasil
4155     SELECT * FROM like_songs WHERE user_id = 1 AND song_id = 10;
4156
4157     -- cek unlike
4158     CALL toggle_like_song(1, 10);
4159
4160
```

Output pane shows:

- NOTICE: LIKE berhasil untuk song_id=10 oleh user_id=1
- CALL
- Query returned successfully in 136 msec.

Bottom status bar: Total rows: Query complete 00:00:00.136 CRLF Ln 4151, Col 1 21:52 24/11/2025

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows database schema with tables like collection_top_3_genres, collections, collections_songs, etc.
- Query Editor:** Contains the following SQL code:

```
4144
4145     RAISE NOTICE 'LIKE DERNASIL UNTUK SONG_ID=% OLEH USER_ID=%';
4146     p_song_id, p_user_id;
4147     END IF;
4148
4149     $$;
4150
4151     -- cek like
4152     CALL toggle_like_song(1, 10);
4153
4154     -- cek hasil
4155     SELECT * FROM like_songs WHERE user_id = 1 AND song_id = 10;
4156
4157     -- cek unlike
4158     CALL toggle_like_song(1, 10);
```

Output pane shows:

- RAISE NOTICE 'LIKE DERNASIL UNTUK SONG_ID=% OLEH USER_ID=%'
- p_song_id, p_user_id;
- END IF;
- \$\$;
- cek like
- CALL toggle_like_song(1, 10);
- cek hasil
- SELECT * FROM like_songs WHERE user_id = 1 AND song_id = 10;
- cek unlike
- CALL toggle_like_song(1, 10);

Data Output pane shows a table:

song_id	user_id	TIMESTAMP
10	1	2025-11-24 21:52:32.364233

Bottom status bar: Total rows: 1 Query complete 00:00:00.165 CRLF Ln 4155, Col 1 21:53 24/11/2025

```

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
Final_Destination/postgres@Postgres16
Query Query History
4147
4148
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4160
END;
$$;
-- cek like
CALL toggle_like_song(1, 10);
-- cek hasil
SELECT * FROM like_songs WHERE user_id = 1 AND song_id = 10;
-- cek unlike
CALL toggle_like_song(1, 10);
SELECT * FROM like_songs;

```

NOTICE: UNLIKE berhasil untuk song_id=10 oleh user_id=1
CALL

Query returned successfully in 113 msec.

Total rows: 0 Query complete 00:00:00.113 CRLF Ln 4157, Col 1

```

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
Final_Destination/postgres@Postgres16
Query Query History
4147
4148
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4160
END;
$$;
-- cek like
CALL toggle_like_song(1, 10);
-- cek hasil
SELECT * FROM like_songs WHERE user_id = 1 AND song_id = 10;
-- cek unlike
CALL toggle_like_song(1, 10);
SELECT * FROM like_songs;

```

song_id user_id TIMESTAMP

Successfully run. Total query runtime: 140 msec. 0 rows affected.

Total rows: 0 Query complete 00:00:00.140 CRLF Ln 4155, Col 61

Tabel 84 PL/SQL – Procedure remove_song_from_playlist

Nama Procedure	Remove_song_from_playlist
Parameter	<p>p_playlist_id (INT) --> ID playlist tempat lagu akan dihapus</p> <p>p_song_id (INT) --> ID lagu yang ingin dihapus dari playlist</p>
Deskripsi	Procedure remove_song_from_playlist ini prosedur yang digunakan untuk menghapus lagu tertentu dari sebuah playlist sekaligus

memastikan struktur playlist tetap rapi dan valid. Prosedur ini terlebih dahulu memeriksa apakah lagu yang akan dihapus benar-benar ada pada playlist; jika tidak ditemukan, sistem akan memberikan error menggunakan RAISE EXCEPTION. Jika lagu ditemukan, prosedur akan menghapusnya dari tabel add_songs_playlists, lalu melakukan proses penataan ulang (re-ordering) nomor urut lagu agar tidak ada nomor yang hilang atau loncat setelah penghapusan—misalnya dari 1,2,4 menjadi 1,2,3. Dengan demikian, prosedur ini menjaga konsistensi data dan memastikan playlist tetap terstruktur secara benar.

Script SQL

```
CREATE OR REPLACE PROCEDURE remove_song_from_playlist(
    p_playlist_id INT, -- Parameter input: ID playlist yang ingin
    dihapus lagunya
    p_song_id INT -- Parameter input: ID lagu yang ingin dihapus
)
LANGUAGE plpgsql
AS $$

DECLARE
    v_exists INT; -- Variabel untuk mengecek apakah lagu ada di
    playlist
BEGIN
    -- Cek apakah lagu ada di playlist
    SELECT COUNT(*)
    INTO v_exists
    FROM add_songs_playlists
    WHERE playlist_id = p_playlist_id
        AND song_id = p_song_id;

    IF v_exists = 0 THEN
        -- Kalo lagu tidak ditemukan di playlist,
        RAISE EXCEPTION 'Lagu % tidak ditemukan di playlist %.', 
        p_song_id, p_playlist_id;
    END IF;

    -- Hapus lagu dari playlist
    DELETE FROM add_songs_playlists
    WHERE playlist_id = p_playlist_id
        AND song_id = p_song_id;
```

```
-- Perbaiki nomor urut (re-order) supaya urutan lagu tetap berurutan
WITH ordered AS (
    SELECT
        add_song_pl_id, -- Ambil ID record
        ROW_NUMBER() OVER (ORDER BY no_urut) AS new_no -- Hitung
nomor urut baru
    FROM add_songs_playlists
    WHERE playlist_id = p_playlist_id -- Hanya untuk playlist yang
sama
)
UPDATE add_songs_playlists asp
SET no_urut = o.new_no -- Update no_urut sesuai urutan baru
FROM ordered o
WHERE asp.add_song_pl_id = o.add_song_pl_id; -- Cocokin dengan ID
record nya

END;
$$;

-- hapus lagu (song_id 30) dari playlist 10
CALL remove_song_from_playlist(10, 30);
```

Screenshot Hasil

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- Sequences (11)
 - seq_add_songs_playlist_id
 - seq_artists_id
 - seq_collections_id
 - seq_genres_id
 - seq_listens_id
 - seq_playlists_id
 - seq_reviews_id
 - seq_socials_id
 - seq_songs_id
 - seq_tours_id
 - seq_users_id
- Tables (27)
 - add_songs_playlists
 - artist_promotion
 - artists
 - Columns (8)
 - artist_id
 - artist_name
 - bio
 - monthly_listener_count
 - artist_pfp

Final_Destination/postgres@Postgres16

Query Query History

```

-- PROCEDURE remove_song_from_playlist
CREATE OR REPLACE PROCEDURE remove_song_from_playlist(
    p_playlist_id INT,
    p_song_id INT
)
LANGUAGE plpgsql
AS $$

DECLARE
    v_exists INT;
BEGIN
    IF v_exists = 0 THEN
        RETURN;
    END IF;
    DELETE FROM add_songs_playlists
    WHERE playlist_id = p_playlist_id
    AND song_id = p_song_id;
    IF rowcount > 0 THEN
        RAISE NOTICE 'Song removed from playlist';
    ELSE
        RAISE NOTICE 'Song not found in playlist';
    END IF;
END;

```

Data Output Messages Notifications

CREATE PROCEDURE

Query returned successfully in 284 msec.

Total rows: 0 Query complete 00:00:00.204 CRLF Ln 3919, Col 1 2058 23/11/2025

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- Sequences (11)
 - seq_add_songs_playlist_id
 - seq_artists_id
 - seq_collections_id
 - seq_genres_id
 - seq_listens_id
 - seq_playlists_id
 - seq_reviews_id
 - seq_socials_id
 - seq_songs_id
 - seq_tours_id
 - seq_users_id
- Tables (27)
 - add_songs_playlists
 - artist_promotion
 - artists
 - Columns (8)
 - artist_id
 - artist_name
 - bio
 - monthly_listener_count
 - artist_pfp

Final_Destination/postgres@Postgres16

Query Query History

```

-- hapus lagu (song_id 5) dari playlist 10
CALL remove_song_from_playlist(10, 5);

-- cek hasil
SELECT *
FROM add_songs_playlists
WHERE playlist_id = 10
ORDER BY no_urut;

```

Data Output Messages Notifications

ERROR: Lagu 5 tidak ditemukan di playlist 10.
CONTEXT: PL/pgSQL function remove_song_from_playlist(integer,integer) line 13 at RAISE
SQL state: P0001

Total rows: 0 Query complete 00:00:00.125 CRLF Ln 3952, Col 1 2059 23/11/2025

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- Sequences (11)
 - seq_add_songs_playlist_id
 - seq_artists_id
 - seq_collections_id
 - seq_genres_id
 - seq_listens_id
 - seq_playlists_id
 - seq_reviews_id
 - seq_socials_id
 - seq_songs_id
 - seq_tours_id
 - seq_users_id
- Tables (27)
 - add_songs_playlists
 - artist_promotion
 - artists
 - Columns (8)
 - artist_id
 - artist_name
 - bio
 - monthly_listener_count
 - artist_pfp

Final_Destination/postgres@Postgres16

Query Query History

```

END;
$$;

-- cek (user_id, playlist_id, song_id)
CALL add_song_to_playlist(2, 10, 5);

SELECT *
FROM add_songs_playlists
WHERE playlist_id = 10
ORDER BY no_urut;

```

Data Output Messages Notifications

	add_song_p_id [PK] integer	user_id integer	playlist_id integer	song_id integer	no_urut integer	TIMESTAMP timestamp without time zone
2	47	29	10	32	2	2025-11-08 01:46:08.262688
3	48	48	10	37	3	2025-11-01 22:01:08.247335
4	49	39	10	109	4	2025-09-25 18:01:56.118343
5	50	34	10	30	5	2025-11-05 08:00:04.068935

Total rows: 5 Query complete 00:00:00.112 CRLF Ln 3904, Col 1 2059 23/11/2025

```

-- hapus lagu (song_id 30) dari playlist 10
CALL remove_song_from_playlist(10, 30);

-- cek hasil
SELECT *
FROM add_songs_playlists
WHERE playlist_id = 10
ORDER BY no_urut;

```

Total rows: Query complete 00:00:00.276 CRLF Ln 3952, Col 1


```

-- hapus lagu (song_id 30) dari playlist 10
CALL remove_song_from_playlist(10, 30);

-- cek hasil
SELECT *
FROM add_songs_playlists
WHERE playlist_id = 10
ORDER BY no_urut;

```

add_song_p_id	user_id	playlist_id	song_id	no_urut	TIMESTAMP
[PK] integer	integer	integer	integer	integer	timestamp without time zone
1	46	2	10	96	1 2025-11-04 23:39:41.711578
2	47	29	10	32	2 2025-11-08 01:46:08.2625888
3	48	48	10	37	3 2025-11-01 22:01:08.247335
4	49	39	10	109	

Total rows: 4 Query complete 00:00:00.163 CRLF Ln 3955, Col 1

Tabel 85 PL/SQL – Procedure add_song_to_playlist

Nama Procedure	Add_song_to_playlist
Parameter	<p>p_user_id INT --> ID user yang menambahkan</p> <p>p_playlist_id INT --> ID playlist tujuan</p> <p>p_song_id INT --> ID lagu yang ingin ditambahkan</p>
Deskripsi	Procedure ini digunakan untuk menambahkan lagu ke dalam playlist

	dengan mekanisme pengecekan dan penomoran otomatis. Fungsi utamanya:
	<ul style="list-style-type: none"> - Memastikan playlist ada - Mencegah duplikasi lagu - Memastikan hak akses user - Menghitung nomor urut lagu otomatis <ul style="list-style-type: none"> - Mengambil MAX(no_urut) - Lagu baru akan berada pada urutan terakhir (seperti Spotify) - Melakukan insert data ke tabel add_songs_playlists

Script SQL

```

CREATE OR REPLACE PROCEDURE add_song_to_playlist(
    p_user_id INT, -- Parameter input: ID user yang ingin menambahkan
lagu
    p_playlist_id INT, -- Parameter input: ID playlist tujuan
    p_song_id INT -- Parameter input: ID lagu yang ingin ditambahkan
)
LANGUAGE plpgsql
AS $$

DECLARE
    v_owner_id INT; -- Variabel untuk menyimpan ID pemilik playlist
    v_is_collab BOOLEAN;
    v_exists INT; -- Variabel untuk mengecek apakah lagu sudah ada di
playlist
    v_last_no_urut INT; -- Variabel untuk menentukan nomor urut lagu
terakhir
BEGIN
    -- Cek apakah playlist ada
    SELECT user_id, isCollaborative
    INTO v_owner_id, v_is_collab
    FROM playlists
    WHERE playlist_id = p_playlist_id;

    IF NOT FOUND THEN
        -- Kalo playlist tidak ditemukan,
        RAISE EXCEPTION 'Playlist % tidak ditemukan.', p_playlist_id;
    END IF;

    -- Cek duplikasi lagu
    SELECT COUNT(*)
    INTO v_exists
    FROM add_songs_playlists

```

```

    WHERE playlist_id = p_playlist_id
        AND song_id = p_song_id;

    IF v_exists > 0 THEN
        -- Kalo lagu sudah ada di playlist,
        RAISE EXCEPTION 'Lagu % sudah ada di playlist .', p_song_id,
p_playlist_id;
    END IF;

    -- Cek kepemilikan playlist (non-collaborative)
    IF v_is_collab = FALSE AND p_user_id <> v_owner_id THEN
        RAISE EXCEPTION
            'Playlist ini non-collaborative. Hanya pemilik (user_id = %)
yang dapat menambahkan lagu.',

            v_owner_id;
    END IF;

    -- Hitung nomor urut otomatis
    SELECT COALESCE(MAX(no_urut), 0)
    INTO v_last_no_urut
    FROM add_songs_playlists
    WHERE playlist_id = p_playlist_id;

    v_last_no_urut := v_last_no_urut + 1; -- Tambah 1 untuk nomor urut
lagu baru

    -- Insert lagu ke playlist
    INSERT INTO add_songs_playlists (
        user_id,
        playlist_id,
        song_id,
        no_urut
    ) VALUES (
        p_user_id,
        p_playlist_id,
        p_song_id,
        v_last_no_urut
    );

END;
$$;

-- cek (user_id, playlist_id, song_id)
CALL add_song_to_playlist(2, 10, 5);

```

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows sequences (11) and tables (27). The sequence `seq_add_songs_playlist_id` is selected.
- Query Editor:** Contains the SQL code for creating the `add_song_to_playlist` procedure. The code includes parameters `p_user_id`, `p_playlist_id`, and `p_song_id`, and a variable `v_owner_id`. It also includes a `DECLARE` section for `v_is_collab`.
- Data Output:** Shows the message "Query returned successfully in 179 msec."
- Messages:** Shows a green success message: "Query returned successfully in 179 msec."
- Notifications:** None.

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows sequences (11) and tables (27). The sequence `seq_add_songs_playlist_id` is selected.
- Query Editor:** Continues the SQL code for the `add_song_to_playlist` procedure. It includes a `SELECT * FROM add_songs_playlists` statement at the end.
- Data Output:** Shows the message "Query returned successfully in 113 msec."
- Messages:** Shows an error message: "Playlist ini non-collaborative. Hanya pemilik (user_id = 15) yang dapat menambahkan lagu." and "CONTEXT: PL/pgSQL function add_song_to_playlist(integer,integer,integer) line 31 at RAISE".
- Notifications:** None.

no urut

```

-- cek (user_id, playlist_id, song_id)
CALL add_song_to_playlist(2, 10, 5);

SELECT *
FROM add_songs_playlists
WHERE playlist_id = 10
ORDER BY no_urut;

```

Successfully run. Total query runtime: 188 msec. 5 rows affected.

Tabel 86 PL/SQL – Procedure create_playlist

Nama Procedure	create_playlist
Parameter	<p>p_user_id (INT)</p> <p>p_title (VARCHAR)</p> <p>p_ispublic (BOOLEAN)</p> <p>p_iscollaborative (BOOLEAN)</p> <p>p_description (TEXT)</p> <p>p_cover (VARCHAR)</p> <p>p_isonprofile (BOOLEAN)</p>
Deskripsi	<p>Procedure create_playlist digunakan untuk menambahkan playlist baru ke dalam tabel playlists.</p> <p>Ketika procedure ini dipanggil, sistem akan memasukkan data playlist baru berdasarkan parameter yang diberikan, seperti user pembuat playlist, judul playlist, status publik atau privat, status kolaboratif,</p>

deskripsi, cover, dan apakah playlist ingin ditampilkan di profil pengguna.

Procedure ini juga mencatat tanggal pembuatan playlist secara otomatis menggunakan CURRENT_DATE, serta menampilkan pesan konfirmasi menggunakan RAISE NOTICE jika proses berhasil.

Script SQL

```
CREATE OR REPLACE PROCEDURE create_playlist(
    p_user_id INT,
    p_title VARCHAR,
    p_ispublic BOOLEAN,
    p_iscollaborative BOOLEAN,
    p_description TEXT,
    p_cover VARCHAR,
    p_isonprofile BOOLEAN
)
LANGUAGE plpgsql
AS $$

BEGIN
    INSERT INTO playlists (
        user_id,
        playlist_cover,
        playlist_title,
        ispublic,
        iscollaborative,
        playlist_desc,
        isonprofile,
        playlist_date_created
    )
    VALUES (
        p_user_id,
        p_cover,
        p_title,
        p_ispublic,
        p_iscollaborative,
        p_description,
        p_isonprofile,
        CURRENT_DATE
    );

    RAISE NOTICE 'Playlist created successfully.';
END;
```

```
$$;

-- CALL user_id, playlist_title, ispublic, iscollaborative,
playlist_desc, playlist_cover, isonprofile
CALL create_playlist(5, 'cek procedure', true, false, 'Santai
sore', 'cover.png', true);
```

Screenshot Hasil

pgAdmin 4

Object Explorer

```

-- Procedure Create Playlist
CREATE OR REPLACE PROCEDURE create_playlist(
    p_user_id INT,
    p_title VARCHAR,
    p_ispublic BOOLEAN,
    p_iscollaborative BOOLEAN,
    p_description TEXT,
    p_cover VARCHAR,
    p_isonprofile BOOLEAN
)
LANGUAGE plpgsql
AS $$
BEGIN

```

CREATE PROCEDURE

Query returned successfully in 253 msec.

Total rows: Query complete 00:00:00.253 CRLF Ln 3790, Col 1

19:38 23/11/2025

pgAdmin 4

Object Explorer

```

        p_iscollaborative,
        p_description,
        p_isonprofile,
        CURRENT_DATE
    );
$$;
RAISE NOTICE 'Playlist created successfully.';
END;
$$;

CALL create_playlist($1, 'cek procedure', true, false, 'SantaI sore');

SELECT *
FROM playlists;

```

NOTICE: Playlist created successfully.

Query returned successfully in 129 msec.

Total rows: Query complete 00:00:00.129 CRLF Ln 3828, Col 1

19:45 23/11/2025

pgAdmin 4

Object Explorer

```

CALL create_playlist($1, 'cek procedure', true, false, 'SantaI sore');

SELECT *
FROM playlists;

```

Showing rows: 1 to 31 Page No: 1 of 1

playlist_id	user_id	playlist_cover	playlist_title	ispublic	iscollaborative	playlist_desc
25	26	[null]	Playlist qui Biru muda collab	true	false	Playlist asik bt
26	27	[null]	Playlist sit Merah collab	true	true	Playlist asik bt
27	28	[null]	Playlist libero Hijau collab	true	false	Playlist asik bt
28	29	[null]	Playlist beatave Hitam collab	true	false	Playlist asik bt
29	30	[null]	Playlist officia Merah jambu collab	true	false	Playlist asik bt
30	2	[null]	Playlist reprehenderit Merah jambu collab	true	false	Playlist asik bt
31	32	cover.png	cek procedure	true	false	SantaI sore

Total rows: 31 Query complete 00:00:00.809 CRLF Ln 3831, Col 1

19:47 23/11/2025

Tabel 87 PL/SQL – Function Register User

Nama Function	register_user
Parameter	p_username VARCHAR, p_raw_pw VARCHAR, p_user_email VARCHAR
Deskripsi	Mendaftarkan user baru dengan memvalidasi email & username, meng-hash password, lalu menghasilkan user_id baru dari sequence.
Script SQL	
<pre> CREATE OR REPLACE FUNCTION register_user(p_username VARCHAR, p_raw_pw VARCHAR, p_user_email VARCHAR) RETURNS TABLE (user_id INT4, message TEXT) AS \$\$ DECLARE v_user_id INT4; v_pw_hash TEXT; BEGIN --cek apakah email sudah dipakai atau belum IF EXISTS (SELECT 1 FROM users WHERE user_email = p_user_email) THEN RETURN QUERY SELECT NULL::INT, 'Email already registered'::TEXT; RETURN; END IF; --cek apakah username sudah dipakai atau belum IF EXISTS (SELECT 1 FROM users WHERE username = p_username) THEN RETURN QUERY SELECT NULL::INT, 'Username already taken'::TEXT; RETURN; END IF; --hash password menggunakan bcrypt v_pw_hash := crypt(p_raw_pw, gen_salt('bf')); --generate user_id dari sequence SELECT nextval('seq_users_id') INTO v_user_id; --insert user baru, user_id dari sequence INSERT INTO users (user_id, username, pw_hash, user_email) VALUES (v_user_id, p_username, v_pw_hash, p_user_email); RETURN QUERY SELECT v_user_id, 'Registration successful'::TEXT; RETURN; END; \$\$ LANGUAGE plpgsql; </pre>	
Screenshot Hasil	

```

CREATE OR REPLACE FUNCTION register_user(p_username VARCHAR, p_password VARCHAR)
RETURNS TEXT AS $$
DECLARE
    v_user_id INT;
    v_pw_hash VARCHAR(255);
BEGIN
    --insert user baru, user_id dari sequence
    INSERT INTO users (user_id, username, pw_hash, user_email)
    VALUES (nextval('seq_users_id'), p_username, crypt(p_password, gen_salt('bf')), p_email);
    RETURN QUERY SELECT v_user_id, 'Registration successful'::TEXT;
END;
$$ LANGUAGE plpgsql;

```

Tabel 88 PL/SQL – Function Login User

Nama Function	login_user
Parameter	p_user_email VARCHAR, p_raw_pw VARCHAR
Deskripsi	Melakukan verifikasi login user dengan cek email dan mencocokkan password yang di-hash.
Script SQL	
<pre> CREATE OR REPLACE FUNCTION login_user(p_user_email VARCHAR, p_raw_pw VARCHAR) RETURNS TABLE (user_id INT4, message TEXT) AS \$\$ DECLARE v_user_id INT4; v_pw_hash VARCHAR(255); BEGIN --cek apakah username sudah terdaftar atau belum SELECT u.user_id, u.pw_hash INTO v_user_id, v_pw_hash FROM users u WHERE u.user_email = p_user_email; --jika email belum terdaftar IF v_user_id IS NULL THEN RETURN QUERY SELECT NULL::INT4, 'Email not registered'::TEXT; RETURN; END IF; --cek password IF crypt(p_raw_pw, v_pw_hash) <> v_pw_hash THEN RETURN QUERY SELECT NULL::INT4, 'Invalid password'::TEXT; RETURN; END IF; --else, berhasil login RETURN QUERY SELECT v_user_id, 'Login successful'::TEXT; RETURN; END; </pre>	

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface with a database browser on the left and a query editor on the right. The query editor displays a PostgreSQL script for logging in a user and creating a procedure to toggle follow status between users. The status bar at the bottom indicates the query was completed successfully in 72 msec.

```
--else, berhasil login
RETURN QUERY SELECT v_user_id, 'Login successful'::TEXT;
RETURN;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM login_user('nurahma@gmail.com', 'password1234');

/*
Procedure: TOGGLE_FOLLOW_USER
toggle follow dan unfollow sesama user/
*/
CREATE OR REPLACE PROCEDURE toggle_follow(user_id INT, follower_id INT)
AS $$
```

Total rows: Query complete 00:00:00.072 CRLF Ln 90, Col 21

Tabel 89 PL/SQL – Procedure Toggle Follow User

Nama Function	toggle_follow_user
Parameter	p_follower_id INT, p_followed_id INT
Deskripsi	Mengikuti atau berhenti mengikuti user lain secara otomatis berdasarkan apakah relationship follow sudah ada atau belum.
Script SQL	
<pre> CREATE OR REPLACE PROCEDURE toggle_follow_user(p_follower_id INT, p_followed_id INT) LANGUAGE PLPGSQL AS \$\$ DECLARE v_exists BOOLEAN; BEGIN --tidak boleh follow/unfollow diri sendiri IF p_follower_id = p_followed_id THEN RAISE EXCEPTION 'User cannot follow/unfollow themselves'; END IF; --cek apakah kedua user valid IF NOT EXISTS (SELECT 1 FROM users WHERE user_id = p_follower_id) THEN RAISE EXCEPTION 'Follower user_id % not found', p_follower_id; END IF; IF NOT EXISTS (SELECT 1 FROM users WHERE user_id = p_followed_id) THEN RAISE EXCEPTION 'Followed user_id % not found', p_followed_id; END IF; --cek apakah sudah follow, jika sudah -> unfollow SELECT EXISTS (SELECT 1 FROM follow_users WHERE follower_id = p_follower_id AND followed_id = p_followed_id) INTO v_exists; IF v_exists THEN DELETE FROM follow_users WHERE follower_id = p_follower_id AND followed_id = p_followed_id; ELSE INSERT INTO follow_users(follower_id, followed_id) VALUES(p_follower_id, p_followed_id); END IF; END; </pre>	

```

--delete relationship from follow_users
DELETE from follow_users
    WHERE follower_id = p_follower_id AND followed_id = p_followed_id;
RAISE NOTICE 'Unfollow successful';
RETURN;
END IF;

--jika belum follow -> follow
--insert ke table follow_users
INSERT INTO follow_users VALUES (p_follower_id, p_followed_id);
RAISE NOTICE 'Follow successful';

END;$$;

```

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface. On the left is a tree view of database objects. The main area contains a query editor with the following PL/pgSQL code:

```

--jika belum follow -> follow
--insert ke table follow_users
INSERT INTO follow_users VALUES (p_follower_id, p_followed_id);
RAISE NOTICE 'Follow successful';

END;$$;

CALL toggle_follow_user(12, 20);
SELECT * FROM follow_users;
CALL unfollow_user(11, 11);
CALL unfollow_user(100, 2);

```

Below the query editor, the status bar displays "Query returned successfully in 57 msec." and "CRLF Ln 136, Col 8".

Tabel 90 PL/SQL – Function Get Followers

Nama Function	get_followers
Parameter	p_user_id INT
Deskripsi	Mengambil daftar user yang mengikuti user tertentu.
Script SQL	

```

CREATE OR REPLACE FUNCTION get_followers(p_user_id INT)
RETURNS TABLE (
    follower_id INT4,
    follower_username VARCHAR(50),
    follower_email VARCHAR(320)
)
AS $$
BEGIN
    RETURN QUERY
    SELECT u.user_id, u.username, u.user_email
    FROM follow_users
    JOIN users u ON fu.follower_id = u.user_id
    WHERE fu.followed_id = p_user_id
    ORDER BY u.username;
END;

```

```
$$ LANGUAGE plpgsql;
```

Screenshot Hasil

```

CREATE OR REPLACE FUNCTION get_followers(p_user_id INT)
RETURNS TABLE (
    follower_id INT4,
    follower_username VARCHAR(50),
    follower_email VARCHAR(320)
)
AS $$
BEGIN
    RETURN QUERY
    SELECT u.user_id, u.username, u. user_email
    FROM follow_users
    JOIN users u ON fu.follower_id = u.user_id
    WHERE fu.followed_id = p_user_id
    ORDER BY u.username;
END;
$$ LANGUAGE plpgsql;

Data Output Messages Notifications
CREATE FUNCTION

Query returned successfully in 107 msec.

Total rows: Query complete 00:00:00.107
✓ Query returned successfully in 107 msec. CRLF Ln 165, Col 21

```

Tabel 91 PL/SQL – Function Get Following

Nama Function	get_following
Parameter	p_user_id INT
Deskripsi	Mengambil daftar user yang diikuti oleh user tertentu.
Script SQL	
<pre> CREATE OR REPLACE FUNCTION get_following(p_user_id INT) RETURNS TABLE (following_id INT4, following_username VARCHAR(50), following_email VARCHAR(320)) AS \$\$ BEGIN RETURN QUERY SELECT u.user_id, u.username, u. user_email FROM follow_users JOIN users u ON fu.followed_id = u.user_id WHERE fu.follower_id = p_user_id ORDER BY u.username; END; \$\$ LANGUAGE plpgsql; </pre>	
Screenshot Hasil	

```

180 RETURNS TABLE (
181     following_id INT4,
182     following_username VARCHAR(50),
183     following_email VARCHAR(320)
184 )
185 AS $$ BEGIN
186     RETURN QUERY
187     SELECT u.user_id, u.username, u.user_email
188     FROM follow_users
189     JOIN users u ON fu.followed_id = u.user_id
190     WHERE fu.follower_id = p_user_id
191     ORDER BY u.username;
192 END;
193 $$ LANGUAGE plpgsql;
194
195 CALL follow_user(12, 40);
196 CALL follow_user(12, 30);
197 CALL follow_user(12, 53);
198
199
200 SELECT * FROM get_following(12);

```

Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 78 msec.

Total rows: Query complete 00:00:00.078 CRLF Ln 194, Col 21

Tabel 92 PL/SQL – Procedure Create Review

Nama Function	create_review
Parameter	p_review_text TEXT, p_rating NUMERIC (3,0), p_user_id INT4, p_collection_id INT4
Deskripsi	Membuat review baru untuk sebuah koleksi setelah memvalidasi user, koleksi, isi review, dan memastikan user belum pernah memberi review yang sama.

Script SQL

```

CREATE OR REPLACE PROCEDURE create_review(
    p_review_text TEXT,
    p_rating NUMERIC (3,0),
    p_user_id INT4,
    p_collection_id INT4
)
LANGUAGE plpgsql
AS $$

DECLARE
    v_review_id INT;
BEGIN
    -- cek user
    IF NOT EXISTS (SELECT 1 FROM users WHERE user_id = p_user_id) THEN
        RAISE EXCEPTION 'User_id % not found', p_user_id;
    END IF;

    -- cek collection
    IF NOT EXISTS (SELECT 1 FROM collections WHERE collection_id = p_collection_id) THEN
        RAISE EXCEPTION 'Collection_id % not found', p_collection_id;
    END IF;

    -- cek review kosong
    IF p_review_text IS NULL OR LENGTH(TRIM(p_review_text)) = 0 THEN
        RAISE EXCEPTION 'Review text cannot be empty';
    END IF;

    -- cek apakah user sudah review koleksi ini
    IF EXISTS (SELECT 1 FROM reviews WHERE user_id = p_user_id

```

```

        AND collection_id = p_collection_id
    ) THEN
        RAISE EXCEPTION 'User has already reviewed this collection';
END IF;

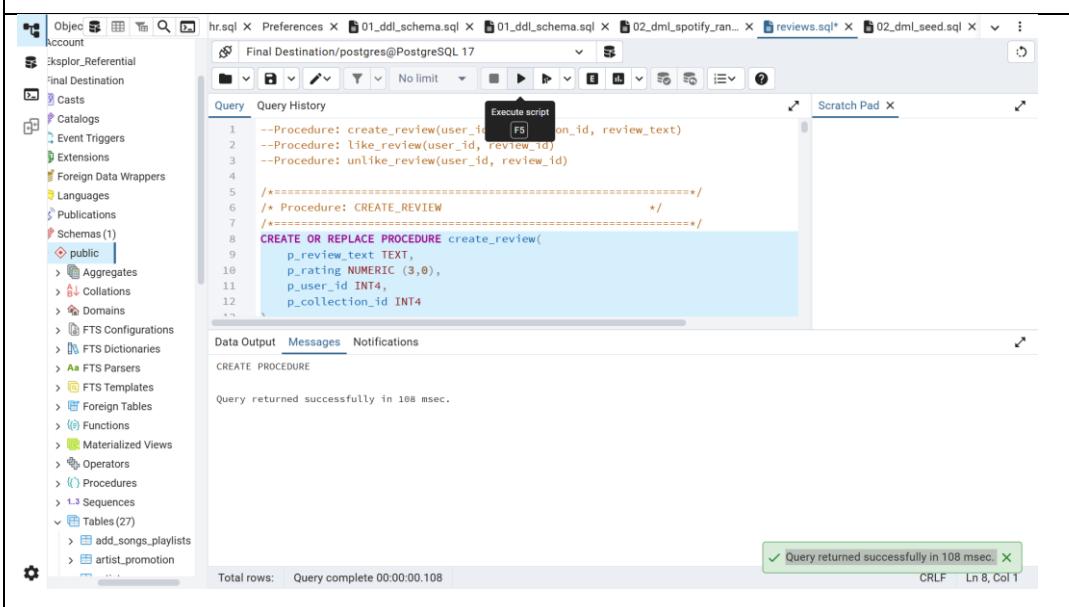
-- ambil review_id dari sequence
SELECT NEXTVAL('seq_reviews_id')
INTO v_review_id;

-- insert review (trigger akan update timestamp jika review diganti)
INSERT INTO reviews(review, rating, "TIMESTAMP", review_id, user_id, collection_id)
VALUES (p_review_text, p_rating, CURRENT_TIMESTAMP, v_review_id, p_user_id,
p_collection_id);

RAISE NOTICE 'Review created with ID %', v_review_id;
END;
$$;

```

Screenshot Hasil



Tabel 93 PL/SQL – Procedure Toggle Like Review

Nama Function	toggle_like_review
Parameter	p_user_id INT, p_review_id INT
Deskripsi	Melakukan like/unlike pada review tergantung apakah user sudah pernah menyukai review tersebut.
Script SQL	

```

CREATE OR REPLACE PROCEDURE toggle_like_review(p_user_id INT, p_review_id INT)
LANGUAGE plpgsql
AS $$$
DECLARE
    v_exists BOOLEAN;
BEGIN
    -- cek user
    IF NOT EXISTS (SELECT 1 FROM users WHERE user_id = p_user_id) THEN

```

```

        RAISE EXCEPTION 'User_id % not found', p_user_id;
    END IF;

    -- cek review
    IF NOT EXISTS (SELECT 1 FROM reviews WHERE review_id = p_review_id) THEN
        RAISE EXCEPTION 'Review_id % not found', p_review_id;
    END IF;

    --cek toggle
    SELECT EXISTS (SELECT 1 FROM like_reviews WHERE review_id = p_review_id AND user_id =
    p_user_id)
    INTO v_exists;

    --jika sudah like -> unlike
    IF v_exists THEN
        DELETE FROM like_reviews WHERE review_id = p_review_id AND user_id = p_user_id;
        RAISE NOTICE 'Review % unliked by user %', p_review_id, p_user_id;
        RETURN;
    END IF;

    -- jika belum like -> insert like
    INSERT INTO like_reviews VALUES (p_review_id, p_user_id);
    RAISE NOTICE 'Review % liked by user %', p_review_id, p_user_id;
END;
$$;

```

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface with the 'reviews.sql' file open. In the center pane, a query window displays the creation of a PL/pgSQL procedure:

```

95  $$;
96
97  CALL toggle_like_review(2, 4);
98  CALL toggle_like_review(9999, 15);
99  CALL toggle_like_review(2, 9999);
100  CALL toggle_like_review(1,4);
101  CALL toggle_like_review(2, 4);
102
103 /*=====
104 /* Procedure: UNLIKE REVIEW
105 /*=====
106 CREATE OR REPLACE PROCEDURE unlike_review(p_user_id INT, p_review_id INT)
107 LANGUAGE plpgsql

```

The procedure is named `unlike_review` and takes two parameters: `p_user_id` and `p_review_id`. It uses the `CALL` statement to invoke the `toggle_like_review` function twice with specific arguments. The code is annotated with comments indicating the start and end of the procedure definition and the function call.

Tabel 94 PL/SQL – Function Search

Nama Function	search
Parameter	keyword TEXT, type TEXT
Deskripsi	Mencari lagu, album, playlist, artis, atau user berdasarkan keyword dan type pencarian, lalu mengembalikan hasil dalam format unified.

Script SQL

```

CREATE OR REPLACE FUNCTION search(keyword TEXT, type TEXT)
RETURNS TABLE (
    result_type TEXT,
    id INT,
    title TEXT,
    info TEXT,
    extra_info TEXT
)
AS $$ 
BEGIN
    -- =====
    -- TYPE: SONG
    -- =====
    IF type = 'song' THEN
        RETURN QUERY
        SELECT
            'song' AS result_type,
            v.song_id AS id,
            v.song_title ::TEXT AS title,
            'Artist: ' || COALESCE(v.artists_name, '-') AS info,
            'Album: ' || COALESCE(v.album_name, '-') ||
            ' | Popularity: ' || COALESCE(v.popularity, 0) AS extra_info
        FROM view_full_song_details v
        WHERE v.song_title ILIKE '%' || keyword || '%'
            OR v.artists_name ILIKE '%' || keyword || '%'
            OR v.album_name ILIKE '%' || keyword || '%'
        ORDER BY v.popularity DESC NULLS LAST;

    -- =====
    -- TYPE: COLLECTION (ALBUM)
    -- =====
    ELSIF type = 'collection' THEN
        RETURN QUERY
        SELECT
            'collection',
            c.collection_id,
            c.collection_title ::TEXT,
            'Artist: ' || COALESCE(aa.album_artists,'-') AS info,
            'Release: ' || COALESCE(c.collection_release_date::TEXT, '-') AS extra_info
        FROM collections c
        JOIN (
            SELECT collection_id,
                STRING_AGG(artist_name, ', ') AS album_artists
            FROM releases r
            JOIN artists a ON a.artist_id = r.artist_id
            GROUP BY collection_id
        ) aa ON aa.collection_id = c.collection_id
        WHERE c.collection_title ILIKE '%' || keyword || '%'
            OR aa.album_artists ILIKE '%' || keyword || '%';

    -- =====
    -- TYPE: PLAYLIST
    -- =====
    ELSIF type = 'playlist' THEN
        RETURN QUERY
        SELECT
            'playlist',
            p.playlist_id,
            p.playlist_title ::TEXT,
            COALESCE(p.playlist_desc, ''),
            COUNT(asp.song_id)::TEXT AS extra
        FROM playlists p
        LEFT JOIN add_songs_playlists asp ON asp.playlist_id = p.playlist_id
        LEFT JOIN view_full_song_details v ON v.song_id = asp.song_id
        WHERE p.playlist_title ILIKE '%' || keyword || '%'
            OR COALESCE(p.playlist_desc, '') ILIKE '%' || keyword || '%'
            OR v.song_title ILIKE '%' || keyword || '%'
            OR v.artists_name ILIKE '%' || keyword || '%'
            OR v.album_name ILIKE '%' || keyword || '%'
        GROUP BY p.playlist_id, p.playlist_title, p.playlist_desc;

```

```

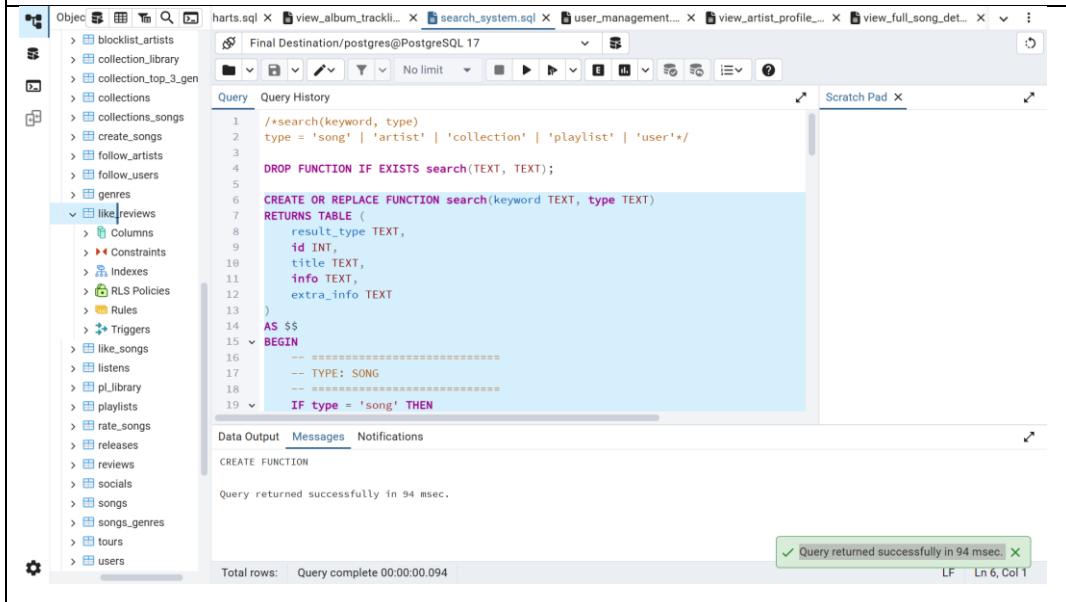
-- =====
-- TYPE: ARTIST
-- =====
ELSIF type = 'artist' THEN
    RETURN QUERY
    SELECT
        'artist',
        v.artist_id,
        v.artist_name ::TEXT,
        v.follower_count::TEXT || ' followers',
        'Albums: ' || total_albums || ' | Tracks: ' || total_tracks AS extra
    FROM view_artist_profile_header v
    WHERE v.artist_name ILIKE '%' || keyword || '%';

-- =====
-- TYPE: USER
-- =====
ELSIF type = 'user' THEN
    RETURN QUERY
    SELECT
        'user',
        u.user_id,
        u.username ::TEXT,
        u.followers_count::TEXT || ' followers',
        'Following: ' || u.following_count ||
        ' | Public playlists: ' || u.public_playlists AS extra
    FROM view_user_library_stats u
    WHERE u.username ILIKE '%' || keyword || '%';

ELSE
    RAISE EXCEPTION 'Invalid type: %, valid types = song | artist | collection |
playlist | user', type;
END IF;
END;
$$ LANGUAGE plpgsql STABLE;

```

Screenshot Hasil



Tabel 95 PL/SQL – Function Get Artist Detail

Nama Function	get_artist_detail
Parameter	p_artist_id INT

Deskripsi	Mengambil informasi lengkap profil artis dari view_artist_profile_header berdasarkan artist_id.
Script SQL	
	<pre>CREATE OR REPLACE FUNCTION get_artist_detail(p_artist_id INT) RETURNS TABLE (artist_id INT, artist_name VARCHAR(255), bio TEXT, artist_pfp VARCHAR(2048), banner VARCHAR(2048), artist_email VARCHAR(320), monthly_listener_count BIGINT, follower_count BIGINT, total_albums BIGINT, total_tracks BIGINT) AS \$\$ BEGIN RETURN QUERY SELECT * FROM view_artist_profile_header v WHERE v.artist_id = p_artist_id; END; \$\$ LANGUAGE plpgsql STABLE;</pre>

Screenshot Hasil

The screenshot shows the pgAdmin interface with the 'Query' tab selected. The query editor contains the SQL code for creating the function. The code is as follows:

```

1  /*****
2  /* Function: GET_ARTIST_DETAIL
3  *****/
4  DROP FUNCTION IF EXISTS get_artist_detail(INT);
5
6  CREATE OR REPLACE FUNCTION get_artist_detail(p_artist_id INT)
7  RETURNS TABLE (
8      artist_id INT,
9      artist_name VARCHAR(255),
10     bio TEXT,
11     artist_pfp VARCHAR(2048),
12     banner VARCHAR(2048),
13     artist_email VARCHAR(320),
14     monthly_listener_count BIGINT,
15     follower_count BIGINT,
16     total_albums BIGINT,
17     total_tracks BIGINT
)
```

Below the code, the message "Query returned successfully in 64 msec." is displayed. The left sidebar shows the database schema with various tables like seq_genres_id, seq_listens_id, seq_playlists_id, seq_reviews_id, seq_socials_id, seq_songs_id, seq_tours_id, seq_users_id, and many tables under the 'Tables (27)' category. The 'artists' table is specifically highlighted.

```

AS $$ AS $$ BEGIN
    RETURN QUERY
        SELECT * FROM view_artist_profile_header v WHERE v.artist_id = p_artist_id;
END;
$$ LANGUAGE plpgsql STABLE;

SELECT * FROM VIEW_ARTIST_PROFILE_HEADER
SELECT * FROM artists
SELECT * FROM get_artist_detail(3)
/*
Function: GET_ARTIST_CONTENT
*/
DROP FUNCTION IF EXISTS get_artist_content(INT, TEXT);
CREATE OR REPLACE FUNCTION get_artist_content(p_artist_id INT, p_type TEXT)
RETURNS TABLE (
    content_type TEXT,
    content_id INT,
    title TEXT,
    extra_info TEXT
) AS $$ LANGUAGE plpgsql STABLE;

```

Showing rows: 1 to 1 Page No: 1 of 1 |<|>|<<|>>|<<<<|>>>>

artist_id	artist_name	bio
3	Paiman Puspasari	Eligendi distinctio exercitationem porro sequi. Laborum temporibus sapiente at incidenti eius a.

Total rows: 1 Query complete 00:00:00.079 ✓ Successfully run. Total query runtime: 79 msec. 1 rows affected. CRLF Ln 28, Col 1

Tabel 96 PL/SQL – Function Get Artist Content

Nama Function	get_artist_content
Parameter	p_artist_id INT, p_type TEXT
Deskripsi	Mengambil daftar konten artis (lagu, koleksi, tur, atau promosi) sesuai kategori yang diminta.
Script SQL	
<pre> CREATE OR REPLACE FUNCTION get_artist_content(p_artist_id INT, p_type TEXT) RETURNS TABLE (content_type TEXT, content_id INT, title TEXT, extra_info TEXT) AS \$\$ LANGUAGE plpgsql STABLE; BEGIN -- ===== SONGS ===== IF p_type = 'songs' THEN RETURN QUERY SELECT 'song', v.song_id, v.song_title ::TEXT, 'Album: ' COALESCE(v.album_name, '-') ' • Popularity: ' v.popularity FROM view_full_song_details v WHERE v.artist_name ILIKE (SELECT '%' artist_name '%' FROM artists WHERE artist_id = p_artist_id) ORDER BY v.popularity DESC; -- ===== COLLECTIONS ===== ELSIF p_type = 'collections' THEN RETURN QUERY SELECT 'collection', c.collection_id, c.collection_title ::TEXT, c.collection_type ::TEXT END IF; END; \$\$ </pre>	

```

        FROM releases r
        JOIN collections c ON c.collection_id = r.collection_id
        WHERE r.artist_id = p_artist_id;

-- ===== TOURS =====
ELSIF p_type = 'tours' THEN
    RETURN QUERY
SELECT
    'tour',
    t.tour_id,
    t.tour_name ::TEXT,
    t.venue || ' • ' || t.tour_date
FROM artists_tours at
JOIN tours t ON t.tour_id = at.tour_id
WHERE at.artist_id = p_artist_id;

-- ===== PROMOTIONS =====
ELSIF p_type = 'promotions' THEN
    RETURN QUERY
SELECT
    'promotion',
    ap.collection_id,
    c.collection_title ::TEXT,
    ap.komentar_promosi
FROM artist_promotion ap
LEFT JOIN collections c ON c.collection_id = ap.collection_id
WHERE ap.artist_id = p_artist_id;

ELSE
    RAISE EXCEPTION 'Invalid type: %, valid types: songs, collections, tours,
promotions', p_type;
END IF;

END;
$$ LANGUAGE plpgsql STABLE;

```

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Navigator:** Shows various database objects like seq_genres_id, seq_listens_id, seq_playlists_id, seq_reviews_id, seq_socials_id, seq_songs_id, seq_tours_id, seq_users_id, and Tables (27) which include add_songs_playlists, artist_promotion, artists, artists_tours, block_users, blocklist_artists, collection_library, collection_top_3_gen, collections, collections_songs, create_songs, follow_artists, follow_users, genres, like_reviews, like_songs, listens, p_library, and playlists.
- Query Editor:** Displays the SQL code for the function get_artist_content. The code includes a comment /* Function: GET_ARTIST_CONTENT */ and logic for handling different content types based on the parameter p_type.
- Status Bar:** Shows "Query returned successfully in 66 msec." and "Total rows: Query complete 00:00:00.066".

```
SELECT * FROM get_artist_content(2, 'songs')
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Browser:** Shows a tree view of database objects including tables like `seq_tours_id`, `seq_users_id`, and various artist-related tables such as `artists`, `artists_tours`, `blocklist_artists`, etc.
- Query Editor:** Displays the PL/pgSQL script `artists.sql` with the following code:

```

95
96     END;
97     $$ LANGUAGE plpgsql STABLE;
98
99     select * from artists
100    SELECT * FROM get_artist_content(2, 'songs')
101    select * from songs
102    select * from view_artist_profile_header
103    select * from view_full_song_details
104
105   /* Procedure: TOGGLE_FOLLOW_ARTIST
106   */
107  CREATE OR REPLACE PROCEDURE toggle_follow_artist(p_user_id INT, p_artist_id INT)
108  LANGUAGE plpgsql
109  AS $$
110  BEGIN
111      -- cek user valid

```
- Data Output:** Shows a table with 11 rows of song data, each with columns: content_type, content_id, title, and extra_info.
- Status Bar:** Shows "Successfully run. Total query runtime: 198 msec. 11 rows affected."

The second screenshot shows the same environment after running a specific query:

Query: `SELECT * FROM get_artist_content(2, 'song')`

Query History: Contains the same PL/pgSQL code as the first screenshot, with the addition of the following line at the top:

```

WHERE p_artist_id = p_artist_list_id;

```

Data Output: Shows an error message:

```

ERROR: Invalid type: song, valid types: songs, collections, tours, promotions
CONTEXT: PL/pgSQL function get_artist_content(integer,text) line 55 at RAISE

```

Tabel 97 PL/SQL – Procedure Toggle Follow Artist

Nama Function	toggle_follow_artist
Parameter	p_user_id INT, p_artist_id INT
Deskripsi	Mengaktifkan fitur follow/unfollow artis dalam satu prosedur berdasarkan status relasi sebelumnya.
Script SQL	
<pre> CREATE OR REPLACE PROCEDURE toggle_follow_artist(p_user_id INT, p_artist_id INT) LANGUAGE plpgsql AS \$\$ BEGIN -- cek user valid IF NOT EXISTS (SELECT 1 FROM users WHERE user_id = p_user_id) THEN RAISE EXCEPTION 'User_id % not found', p_user_id; END IF; WHERE p_artist_id = p_artist_list_id; </pre>	

```

        END IF;

        -- cek artist valid
        IF NOT EXISTS (SELECT 1 FROM artists WHERE artist_id = p_artist_id) THEN
            RAISE EXCEPTION 'Artist_id % not found', p_artist_id;
        END IF;

        -- jika sudah follow -> UNFOLLOW
        IF EXISTS (
            SELECT 1 FROM follow_artists
            WHERE user_id = p_user_id AND artist_id = p_artist_id
        ) THEN
            DELETE FROM follow_artists
            WHERE user_id = p_user_id AND artist_id = p_artist_id;

            RAISE NOTICE 'Unfollowed artist %', p_artist_id;
            RETURN;
        END IF;

        -- jika belum follow -> FOLLOW
        INSERT INTO follow_artists(user_id, artist_id)
        VALUES (p_user_id, p_artist_id);

        RAISE NOTICE 'Followed artist %', p_artist_id;

    END;
$$;

```

Screenshot Hasil

```

132
133     -- jika belum follow -> FOLLOW
134     INSERT INTO follow_artists(user_id, artist_id)
135     VALUES (p_user_id, p_artist_id);
136
137     RAISE NOTICE 'Followed artist %', p_artist_id;
138
139     END;
140     $$;
141
142     call toggle_follow_artist(1, 12)
143     select * from follow_artists
144     call toggle_follow_artist(9999, 12)
145     call toggle_follow_artist(12, 9999)
146     call toggle_follow_artist(1, 12)
147     select * from follow_artists
148

```

Data Output Messages Notifications Scratch Pad X

CREATE PROCEDURE

Query returned successfully in 68 msec.

Total rows: Query complete 00:00:00.068 ✓ Query returned successfully in 68 msec. X

Tabel 98 PL/SQL – Function Get Collection Detail

Nama Function	get_collection_detail
Parameter	p_collection_id INT
Deskripsi	Mengambil detail lengkap sebuah koleksi (album/EP/single) beserta jumlah lagu dan jumlah artis yang terlibat.
Script SQL	

```

CREATE OR REPLACE FUNCTION get_collection_detail(p_collection_id INT)
RETURNS TABLE (
    collection_id INT,
    collection_title TEXT,
    collection_type TEXT,
    collection_cover TEXT,
    release_date DATE,
    rating NUMERIC(3,0),
    is_prerelease BOOL,
    total_tracks BIGINT,
    total_artists BIGINT
)
AS $$ 
BEGIN
    RETURN QUERY
    SELECT
        c.collection_id,
        c.collection_title ::TEXT,
        c.collection_type ::TEXT,
        c.collection_cover ::TEXT,
        c.collection_release_date,
        c.collection_rating,
        c.isprerelease,
        (SELECT COUNT(*) FROM collections_songs cs
         WHERE cs.collection_id = c.collection_id) AS total_tracks,
        (SELECT COUNT(DISTINCT r.artist_id)
         FROM releases r
         WHERE r.collection_id = c.collection_id) AS total_artists
    FROM collections c
    WHERE c.collection_id = p_collection_id;
END;
$$ LANGUAGE plpgsql STABLE;

```

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface with the 'Procedures' node selected in the left sidebar. A query window is open with the following SQL code:

```

1 /*10. get_collection_detail(collection_id) - FUNCTION
2 11. get_collection_tracks(collection_id) - FUNCTION
3 12. get_new_releases(limit_n) - FUNCTION
4      Mirip Spotify Browse > "New Releases".
5      > get_artist_collections sudah ditutup fungsinya di modul Artist.*/
6 DROP FUNCTION IF EXISTS get_collection_detail(INT);
7 CREATE OR REPLACE FUNCTION get_collection_detail(p_collection_id INT)
8 RETURNS TABLE (
9     collection_id INT,
10    collection_title TEXT,
11    collection_type TEXT,
12    collection_cover TEXT,
13    release_date DATE,
14    rating NUMERIC(3,0),
15    is_prerelease BOOL,
16    total_tracks BIGINT,
17    total_artists BIGINT
18 )
19 AS $$
20  v BEGIN
21    RETURN QUERY

```

The status bar at the bottom right indicates: 'Query returned successfully in 110 msec.' and 'CRLF Ln 7, Col 1'.

Tabel 99 PL/SQL – Function Get Collection Tracks

Nama Function	get_collection_tracks
Parameter	p_collection_id INT

Deskripsi	Mengembalikan daftar track dalam sebuah koleksi lengkap dengan artis, durasi, file audio, dan urutan track.
Script SQL	
<pre> CREATE OR REPLACE FUNCTION get_collection_tracks(p_collection_id INT) RETURNS TABLE (track_number INT, song_id INT, song_title TEXT, artist_name TEXT, duration INT, popularity NUMERIC(3,0), song_file TEXT) AS \$\$ BEGIN RETURN QUERY SELECT cs.nomor_track, v.song_id, v.song_title ::TEXT, v.artist_name ::TEXT, v.song_duration, v.popularity, v.song_file ::TEXT FROM collections_songs cs JOIN view_full_song_details v ON v.song_id = cs.song_id WHERE cs.collection_id = p_collection_id ORDER BY cs.nomor_disc, cs.nomor_track; END; \$\$ LANGUAGE plpgsql STABLE; </pre>	

Screenshot Hasil

The screenshot shows the pgAdmin interface with the 'Objects' tab selected. A script editor window is open, showing the PL/pgSQL code for the 'get_collection_tracks' function. The code defines a function that takes a parameter 'p_collection_id' and returns a table with columns: track_number, song_id, song_title, artist_name, duration, popularity, and song_file. It performs a JOIN between 'collections_songs' and 'view_full_song_details' tables, filtering by 'collection_id'. The function is created using the 'plpgsql' language and is marked as 'STABLE'. The status bar at the bottom indicates the query was completed successfully in 142 msec.

Tabel 100 PL/SQL – Function Get New Releases

Nama Function	get_new_realeses
Parameter	p_limit INT

Deskripsi	Mengambil daftar rilis terbaru (dibatasi p_limit) beserta artis dan total lagu dalam setiap koleksi
Script SQL	
<pre> CREATE OR REPLACE FUNCTION get_new_releases(p_limit INT) RETURNS TABLE (collection_id INT, title TEXT, artist_name TEXT, release_date DATE, total_tracks BIGINT) AS \$\$ BEGIN RETURN QUERY SELECT c.collection_id, c.collection_title ::TEXT, STRING_AGG(DISTINCT a.artist_name, ', ') AS artist_name, c.collection_release_date, COUNT(cs.song_id) AS total_tracks FROM collections c LEFT JOIN releases r ON r.collection_id = c.collection_id LEFT JOIN artists a ON a.artist_id = r.artist_id LEFT JOIN collections_songs cs ON cs.collection_id = c.collection_id GROUP BY c.collection_id, c.collection_title, c.collection_release_date ORDER BY c.collection_release_date DESC LIMIT p_limit; END; \$\$ LANGUAGE plpgsql STABLE; </pre>	

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface with the 'Final Destination/postgres@PostgreSQL 17' connection selected. In the left sidebar, under 'Procedures', there is a new entry for 'get_new_releases'. The main pane displays the function's source code. Below the code, the message 'Query returned successfully in 153 msec.' is visible. The bottom status bar indicates 'CRLF' and 'Ln 103, Col 28'.

Tabel 101 PL/SQL – Procedure Add Artist Promotion

Nama Function	add_artist_promotion
Parameter	p_artist_id INT, p_collection_id INT, p_comment TEXT

Deskripsi	Menambahkan atau memperbarui komentar promosi seorang artis untuk sebuah koleksi; hanya diizinkan jika koleksi tersebut memang dirilis oleh artis tersebut.
Script SQL	
<pre> CREATE OR REPLACE PROCEDURE add_artist_promotion(p_artist_id INT, p_collection_id INT, p_comment TEXT) LANGUAGE plpgsql AS \$\$ BEGIN -- cek artist valid IF NOT EXISTS (SELECT 1 FROM artists WHERE artist_id = p_artist_id) THEN RAISE EXCEPTION 'Artist_id % not found', p_artist_id; END IF; -- cek collection ada IF NOT EXISTS (SELECT 1 FROM collections WHERE collection_id = p_collection_id) THEN RAISE EXCEPTION 'Collection_id % not found', p_collection_id; END IF; --validasi, apakah koleksi ini dirilis oleh artis tersebut? IF NOT EXISTS (SELECT 1 FROM releases WHERE artist_id = p_artist_id AND collection_id = p_collection_id) THEN RAISE EXCEPTION 'Artist % cannot promote collection %, because it does not belong to them', p_artist_id, p_collection_id; END IF; --jika sudah pernah promosi -> update komentar IF EXISTS (SELECT 1 FROM artist_promotion WHERE artist_id = p_artist_id AND collection_id = p_collection_id) THEN UPDATE artist_promotion SET komentar_promosi = p_comment WHERE artist_id = p_artist_id AND collection_id = p_collection_id; RAISE NOTICE 'Promotion updated successfully'; RETURN; END IF; --jika belum -> insert baru INSERT INTO artist_promotion(artist_id, collection_id, komentar_promosi) VALUES (p_artist_id, p_collection_id, p_comment); RAISE NOTICE 'Promotion added successfully'; END; \$\$; </pre>	

Screenshot Hasil

```

--jika belum -> insert baru
INSERT INTO artist_promotion(artist_id, collection_id, komentar_promosi)
VALUES (p.artist_id, p.collection_id, p.comment);
RAISE NOTICE 'Promotion added successfully';

END;
$$;

select * from releases
CALL add_artist_promotion(3, 3, 'Check out my new album!');
select * from artist_promotion
CALL add_artist_promotion(3, 3, 'Updated promo!');
select * from artist_promotion

```

Query returned successfully in 128 msec.

Total rows: Query complete 00:00:00.128 CRLF Ln 48, Col 4

Tabel 102 PL/SQL – Function Get Artist Tours

Nama Function	get_artists_tours
Parameter	p_artist_id INT
Deskripsi	Mengambil daftar tur dari artis berdasarkan tabel artists_tours dan tours, diurutkan berdasarkan tanggal tur.

Script SQL

```

CREATE OR REPLACE FUNCTION get_artist_tours(p_artist_id INT)
RETURNS TABLE (
    tour_id INT,
    tour_name TEXT,
    venue TEXT,
    tour_date DATE
)
AS $$$
BEGIN
    RETURN QUERY
    SELECT
        t.tour_id,
        t.tour_name ::TEXT,
        t.venue ::TEXT,
        t.tour_date
    FROM artists_tours at
    JOIN tours t ON t.tour_id = at.tour_id
    WHERE at.artist_id = p_artist_id
    ORDER BY t.tour_date ASC;
END;
$$ LANGUAGE plpgsql STABLE;

```

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface. On the left is a tree view of database objects under the schema 'Final Destination'. The 'like_reviews' table is selected. The main window contains a query editor with the following SQL code:

```
62 /*****  
63 CREATE OR REPLACE FUNCTION get_artist_tours(p_artist_id INT)  
64 RETURNS TABLE (  
65   tour_id INT,  
66   tour_name TEXT,  
67   venue TEXT,  
68   tour_date DATE  
69 )  
70 AS $$  
71 BEGIN  
72   RETURN QUERY  
73   SELECT  
74     t.tour_id,  
75     t.tour_name ::TEXT,  
76     t.venue ::TEXT,  
77     t.tour_date  
78   FROM artists_tours at  
79   JOIN tours t ON t.tour_id = at.tour_id  
80   WHERE at.artist_id = p_artist_id  
81   ORDER BY t.tour_date ASC;  
82 END;  
83 $$ LANGUAGE plpgsql STABLE;
```

The 'Data Output' tab shows the message: 'CREATE FUNCTION'. Below it, 'Query returned successfully in 373 msec.' and 'Total rows: Query complete 00:00:00.379'.

Tabel 103 PL/SQL – Function delete_expired_tours

Nama Function	delete_expired_tours
Parameter	
Deskripsi	Menghapus semua entry tour yang tanggalnya sudah lewat hari ini (tour_date < CURRENT_DATE). Dalam bentuk function karena merupakan sebuah cron.

Script SQL

```
CREATE OR REPLACE FUNCTION delete_expired_tours()
RETURNS VOID AS $$ 
BEGIN
    DELETE FROM tours
    WHERE tour_date < CURRENT_DATE;
END;
$$ LANGUAGE plpgsql;
```

Screenshot Hasil

Before

Grid	tour_id	tour_date	tour_name	venue
1	1	2025-11-25	tur satu	sebelah mall cihampelas, Jakarta, ID
2	2	2025-12-18	tur dua	green garden, Chicago, US

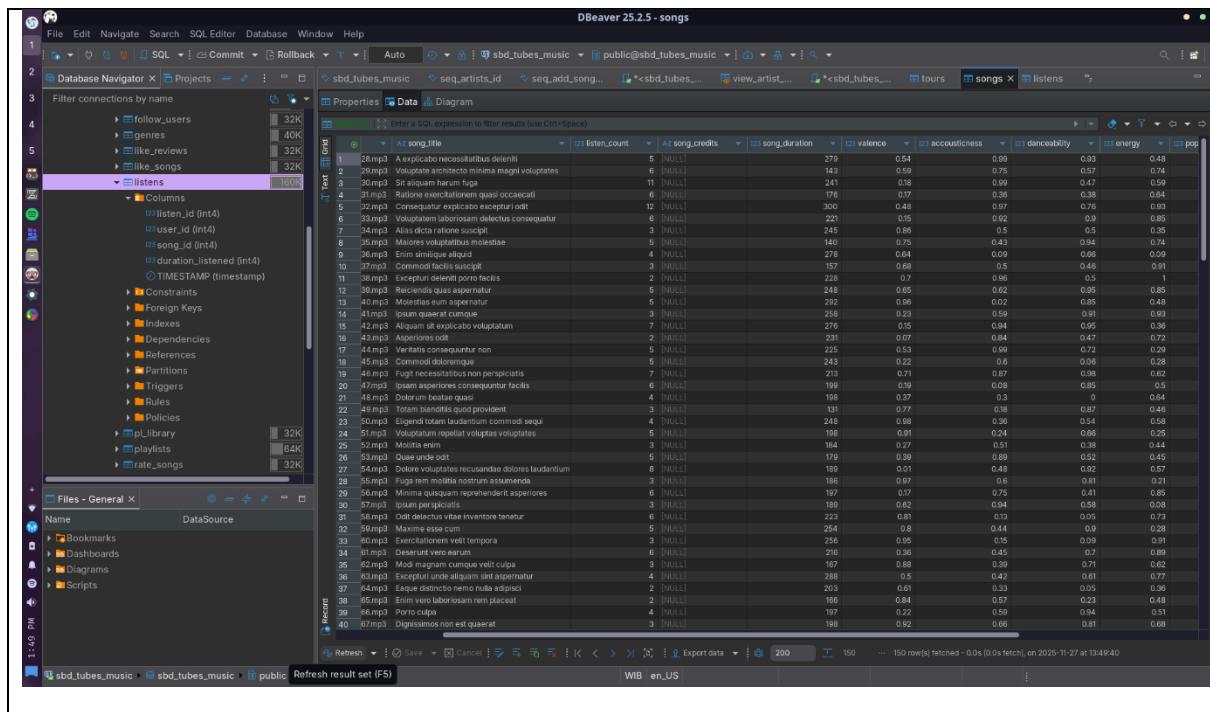
*current_date = 2025-11-27

After

Grid	tour_id	tour_date	tour_name	venue
1	2	2025-12-18	tur dua	green garden, Chicago, US

Tabel 104 PL/SQL – Function recalc_listen_count

Nama Function	recalc_listen_count																																																																																																																																																																																																																																																																																																																																																																																																																									
Parameter																																																																																																																																																																																																																																																																																																																																																																																																																										
Deskripsi	Menghitung listen_count dan mengupdate nya untuk setiap lagu. Dalam bentuk function karena merupakan sebuah cron.																																																																																																																																																																																																																																																																																																																																																																																																																									
Script SQL																																																																																																																																																																																																																																																																																																																																																																																																																										
<pre>CREATE OR REPLACE FUNCTION recalc_listen_count() RETURNS void AS \$\$ BEGIN UPDATE songs s SET listen_count = sub.cnt FROM (SELECT song_id, COUNT(*) AS cnt FROM listens GROUP BY song_id) sub WHERE s.song_id = sub.song_id; END; \$\$ LANGUAGE plpgsql;</pre>																																																																																																																																																																																																																																																																																																																																																																																																																										
Screenshot Hasil																																																																																																																																																																																																																																																																																																																																																																																																																										
<p>Before</p> <table border="1"> <thead> <tr> <th>song_id</th> <th>song_title</th> <th>listen_count</th> <th>song_credits</th> <th>song_duration</th> <th>valence</th> <th>acousticness</th> <th>danceability</th> <th>energy</th> <th>popularity</th> </tr> </thead> <tbody> <tr><td>1</td><td>1.mp3</td><td>163</td><td>0.87</td><td>0.79</td><td>0.15</td><td>0.88</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>2</td><td>2.mp3</td><td>195</td><td>0.93</td><td>0.44</td><td>0.49</td><td>0.88</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>3</td><td>3.mp3</td><td>189</td><td>0.73</td><td>0.24</td><td>0.74</td><td>0.37</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>4</td><td>4.mp3</td><td>230</td><td>0.11</td><td>0.83</td><td>0.4</td><td>0.16</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>5</td><td>5.mp3</td><td>149</td><td>0.84</td><td>0.04</td><td>0.54</td><td>0.09</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>6</td><td>6.mp3</td><td>205</td><td>0.82</td><td>0.3</td><td>0.61</td><td>0.15</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>7</td><td>7.mp3</td><td>165</td><td>0.47</td><td>0.73</td><td>0.41</td><td>0.61</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>8</td><td>8.mp3</td><td>190</td><td>0.3</td><td>0.28</td><td>0.14</td><td>0.5</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>9</td><td>9.mp3</td><td>165</td><td>0.09</td><td>0.92</td><td>0.6</td><td>0.01</td><td>0.73</td><td>0.05</td><td>0.01</td></tr> <tr><td>10</td><td>10.mp3</td><td>145</td><td>0.75</td><td>0.27</td><td>0.89</td><td>0.4</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>11</td><td>11.mp3</td><td>224</td><td>0.8</td><td>0.78</td><td>0.08</td><td>0.73</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>12</td><td>12.mp3</td><td>277</td><td>0.19</td><td>0.28</td><td>0.79</td><td>0.05</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>13</td><td>13.mp3</td><td>288</td><td>0.09</td><td>0.63</td><td>0.73</td><td>0.03</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>14</td><td>14.mp3</td><td>214</td><td>0.7</td><td>0.08</td><td>0.98</td><td>0.48</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>15</td><td>15.mp3</td><td>276</td><td>0.06</td><td>1</td><td>0.14</td><td>0.89</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>16</td><td>16.mp3</td><td>180</td><td>0.8</td><td>0.28</td><td>0.01</td><td>0.01</td><td>0.73</td><td>0.05</td><td>0.01</td></tr> <tr><td>17</td><td>17.mp3</td><td>270</td><td>0.41</td><td>0.68</td><td>0.91</td><td>0.28</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>18</td><td>18.mp3</td><td>217</td><td>0.43</td><td>0.19</td><td>0.64</td><td>0.51</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>19</td><td>19.mp3</td><td>153</td><td>0.81</td><td>0.78</td><td>0.79</td><td>0.5</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>20</td><td>20.mp3</td><td>294</td><td>0.05</td><td>0.08</td><td>0.8</td><td>0.23</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>21</td><td>21.mp3</td><td>279</td><td>0.58</td><td>0.9</td><td>0.23</td><td>1</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>22</td><td>22.mp3</td><td>207</td><td>0.61</td><td>0.63</td><td>0.83</td><td>0.01</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>23</td><td>23.mp3</td><td>132</td><td>0.08</td><td>0.25</td><td>0.15</td><td>0.23</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>24</td><td>24.mp3</td><td>174</td><td>0.09</td><td>0.57</td><td>0.45</td><td>0.88</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>25</td><td>25.mp3</td><td>170</td><td>0.01</td><td>0.77</td><td>0.41</td><td>0.01</td><td>0.73</td><td>0.05</td><td>0.01</td></tr> <tr><td>26</td><td>26.mp3</td><td>293</td><td>0.58</td><td>0.04</td><td>0.35</td><td>0.77</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>27</td><td>27.mp3</td><td>258</td><td>0.53</td><td>0.03</td><td>0.8</td><td>0.05</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>28</td><td>28.mp3</td><td>279</td><td>0.54</td><td>0.99</td><td>0.93</td><td>0.48</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>29</td><td>29.mp3</td><td>143</td><td>0.59</td><td>0.75</td><td>0.57</td><td>0.74</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>30</td><td>30.mp3</td><td>241</td><td>0.18</td><td>0.99</td><td>0.47</td><td>0.59</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>31</td><td>31.mp3</td><td>171</td><td>0.17</td><td>0.38</td><td>0.64</td><td>0.01</td><td>0.73</td><td>0.05</td><td>0.01</td></tr> <tr><td>32</td><td>32.mp3</td><td>300</td><td>0.48</td><td>0.97</td><td>0.76</td><td>0.93</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>33</td><td>33.mp3</td><td>211</td><td>0.13</td><td>0.02</td><td>0.33</td><td>0.55</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>34</td><td>34.mp3</td><td>245</td><td>0.86</td><td>0.5</td><td>0.5</td><td>0.33</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>35</td><td>35.mp3</td><td>140</td><td>0.75</td><td>0.43</td><td>0.94</td><td>0.74</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>36</td><td>36.mp3</td><td>278</td><td>0.64</td><td>0.09</td><td>0.68</td><td>0.09</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>37</td><td>37.mp3</td><td>157</td><td>0.68</td><td>0.5</td><td>0.48</td><td>0.91</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>38</td><td>38.mp3</td><td>228</td><td>0.7</td><td>0.98</td><td>0.5</td><td>1</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>39</td><td>39.mp3</td><td>248</td><td>0.85</td><td>0.82</td><td>0.95</td><td>0.95</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> <tr><td>40</td><td>40.mp3</td><td>292</td><td>0.98</td><td>0.92</td><td>0.85</td><td>0.48</td><td>0.41</td><td>0.05</td><td>0.01</td></tr> </tbody> </table>	song_id	song_title	listen_count	song_credits	song_duration	valence	acousticness	danceability	energy	popularity	1	1.mp3	163	0.87	0.79	0.15	0.88	0.41	0.05	0.01	2	2.mp3	195	0.93	0.44	0.49	0.88	0.41	0.05	0.01	3	3.mp3	189	0.73	0.24	0.74	0.37	0.41	0.05	0.01	4	4.mp3	230	0.11	0.83	0.4	0.16	0.41	0.05	0.01	5	5.mp3	149	0.84	0.04	0.54	0.09	0.41	0.05	0.01	6	6.mp3	205	0.82	0.3	0.61	0.15	0.41	0.05	0.01	7	7.mp3	165	0.47	0.73	0.41	0.61	0.41	0.05	0.01	8	8.mp3	190	0.3	0.28	0.14	0.5	0.41	0.05	0.01	9	9.mp3	165	0.09	0.92	0.6	0.01	0.73	0.05	0.01	10	10.mp3	145	0.75	0.27	0.89	0.4	0.41	0.05	0.01	11	11.mp3	224	0.8	0.78	0.08	0.73	0.41	0.05	0.01	12	12.mp3	277	0.19	0.28	0.79	0.05	0.41	0.05	0.01	13	13.mp3	288	0.09	0.63	0.73	0.03	0.41	0.05	0.01	14	14.mp3	214	0.7	0.08	0.98	0.48	0.41	0.05	0.01	15	15.mp3	276	0.06	1	0.14	0.89	0.41	0.05	0.01	16	16.mp3	180	0.8	0.28	0.01	0.01	0.73	0.05	0.01	17	17.mp3	270	0.41	0.68	0.91	0.28	0.41	0.05	0.01	18	18.mp3	217	0.43	0.19	0.64	0.51	0.41	0.05	0.01	19	19.mp3	153	0.81	0.78	0.79	0.5	0.41	0.05	0.01	20	20.mp3	294	0.05	0.08	0.8	0.23	0.41	0.05	0.01	21	21.mp3	279	0.58	0.9	0.23	1	0.41	0.05	0.01	22	22.mp3	207	0.61	0.63	0.83	0.01	0.41	0.05	0.01	23	23.mp3	132	0.08	0.25	0.15	0.23	0.41	0.05	0.01	24	24.mp3	174	0.09	0.57	0.45	0.88	0.41	0.05	0.01	25	25.mp3	170	0.01	0.77	0.41	0.01	0.73	0.05	0.01	26	26.mp3	293	0.58	0.04	0.35	0.77	0.41	0.05	0.01	27	27.mp3	258	0.53	0.03	0.8	0.05	0.41	0.05	0.01	28	28.mp3	279	0.54	0.99	0.93	0.48	0.41	0.05	0.01	29	29.mp3	143	0.59	0.75	0.57	0.74	0.41	0.05	0.01	30	30.mp3	241	0.18	0.99	0.47	0.59	0.41	0.05	0.01	31	31.mp3	171	0.17	0.38	0.64	0.01	0.73	0.05	0.01	32	32.mp3	300	0.48	0.97	0.76	0.93	0.41	0.05	0.01	33	33.mp3	211	0.13	0.02	0.33	0.55	0.41	0.05	0.01	34	34.mp3	245	0.86	0.5	0.5	0.33	0.41	0.05	0.01	35	35.mp3	140	0.75	0.43	0.94	0.74	0.41	0.05	0.01	36	36.mp3	278	0.64	0.09	0.68	0.09	0.41	0.05	0.01	37	37.mp3	157	0.68	0.5	0.48	0.91	0.41	0.05	0.01	38	38.mp3	228	0.7	0.98	0.5	1	0.41	0.05	0.01	39	39.mp3	248	0.85	0.82	0.95	0.95	0.41	0.05	0.01	40	40.mp3	292	0.98	0.92	0.85	0.48	0.41	0.05	0.01
song_id	song_title	listen_count	song_credits	song_duration	valence	acousticness	danceability	energy	popularity																																																																																																																																																																																																																																																																																																																																																																																																																	
1	1.mp3	163	0.87	0.79	0.15	0.88	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
2	2.mp3	195	0.93	0.44	0.49	0.88	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
3	3.mp3	189	0.73	0.24	0.74	0.37	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
4	4.mp3	230	0.11	0.83	0.4	0.16	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
5	5.mp3	149	0.84	0.04	0.54	0.09	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
6	6.mp3	205	0.82	0.3	0.61	0.15	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
7	7.mp3	165	0.47	0.73	0.41	0.61	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
8	8.mp3	190	0.3	0.28	0.14	0.5	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
9	9.mp3	165	0.09	0.92	0.6	0.01	0.73	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
10	10.mp3	145	0.75	0.27	0.89	0.4	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
11	11.mp3	224	0.8	0.78	0.08	0.73	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
12	12.mp3	277	0.19	0.28	0.79	0.05	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
13	13.mp3	288	0.09	0.63	0.73	0.03	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
14	14.mp3	214	0.7	0.08	0.98	0.48	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
15	15.mp3	276	0.06	1	0.14	0.89	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
16	16.mp3	180	0.8	0.28	0.01	0.01	0.73	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
17	17.mp3	270	0.41	0.68	0.91	0.28	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
18	18.mp3	217	0.43	0.19	0.64	0.51	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
19	19.mp3	153	0.81	0.78	0.79	0.5	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
20	20.mp3	294	0.05	0.08	0.8	0.23	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
21	21.mp3	279	0.58	0.9	0.23	1	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
22	22.mp3	207	0.61	0.63	0.83	0.01	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
23	23.mp3	132	0.08	0.25	0.15	0.23	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
24	24.mp3	174	0.09	0.57	0.45	0.88	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
25	25.mp3	170	0.01	0.77	0.41	0.01	0.73	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
26	26.mp3	293	0.58	0.04	0.35	0.77	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
27	27.mp3	258	0.53	0.03	0.8	0.05	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
28	28.mp3	279	0.54	0.99	0.93	0.48	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
29	29.mp3	143	0.59	0.75	0.57	0.74	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
30	30.mp3	241	0.18	0.99	0.47	0.59	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
31	31.mp3	171	0.17	0.38	0.64	0.01	0.73	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
32	32.mp3	300	0.48	0.97	0.76	0.93	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
33	33.mp3	211	0.13	0.02	0.33	0.55	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
34	34.mp3	245	0.86	0.5	0.5	0.33	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
35	35.mp3	140	0.75	0.43	0.94	0.74	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
36	36.mp3	278	0.64	0.09	0.68	0.09	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
37	37.mp3	157	0.68	0.5	0.48	0.91	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
38	38.mp3	228	0.7	0.98	0.5	1	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
39	39.mp3	248	0.85	0.82	0.95	0.95	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	
40	40.mp3	292	0.98	0.92	0.85	0.48	0.41	0.05	0.01																																																																																																																																																																																																																																																																																																																																																																																																																	



Tabel 105 PL/SQL – Function recalculate_all_song_popularity

Nama Function	recalculate_all_song_popularity
Parameter	
Deskripsi	Menghitung popularity suatu lagu dengan rumus tertentu dan mengupdate nya untuk setiap lagu. Dalam bentuk function karena merupakan sebuah cron.
Script SQL	
<pre> CREATE OR REPLACE FUNCTION recalculate_all_song_popularity() RETURNS VOID AS \$\$ BEGIN -- hitung play maksimum dalam 30 hari terakhir WITH play_counts AS (SELECT song_id, COUNT(*) AS plays_30 FROM listens WHERE "TIMESTAMP" >= CURRENT_DATE - INTERVAL '30 days' GROUP BY song_id), max_play AS (SELECT MAX(plays_30) AS max_plays FROM play_counts) UPDATE songs s SET popularity = CASE WHEN pc.plays_30 IS NULL THEN 0 -- tidak ada play sama sekali -> 0 WHEN mp.max_plays = 0 THEN 0 -- kalau kosong ELSE ROUND((pc.plays_30::numeric / mp.max_plays::numeric) * 100) END FROM play_counts pc, max_play mp WHERE s.song_id = pc.song_id; END; \$\$ LANGUAGE plpgsql; </pre>	

Screenshot Hasil

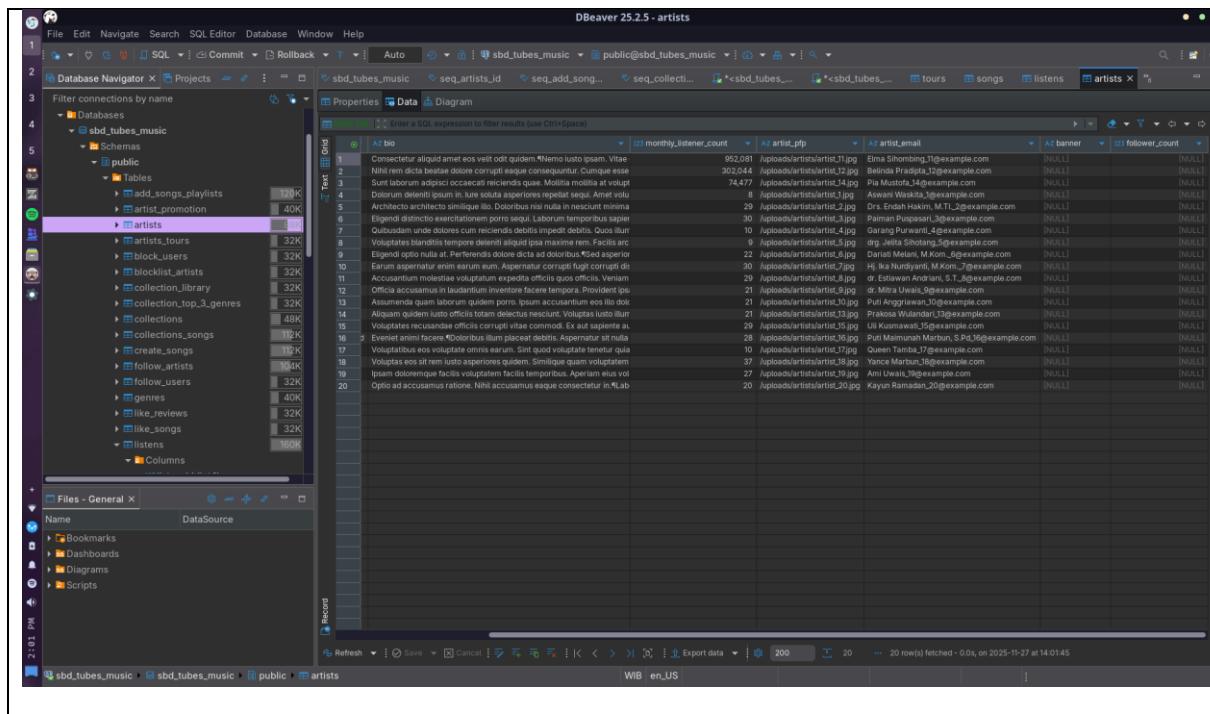
Before

		int	song_credits	float	song_duration	float	valence	float	acousticness	float	danceability	float	energy	float	popularity	float	song_release_date	date	song_rating	float
1	2	5	[NULL]	279	0.54	0.99	0.93	0.48	0.48	0.72	0.98	0.42	0.20	0.19	0.99	2019-04-02	[NULL]	[NULL]	[NULL]	
2	6	[NULL]	143	0.59	0.75	0.57	0.74	0.42	0.57	0.54	0.36	0.64	0.40	0.59	0.36	2020-10-31	[NULL]	[NULL]	[NULL]	
3	11	[NULL]	241	0.18	0.99	0.47	0.59	0.36	0.38	0.64	0.0	0.27	0.45	0.53	0.37	2021-04-25	[NULL]	[NULL]	[NULL]	
4	4	[NULL]	176	0.57	0.97	0.47	0.55	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2021-06-29	[NULL]	[NULL]	[NULL]	
5	6	[NULL]	300	0.40	0.97	0.47	0.55	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2020-10-16	[NULL]	[NULL]	[NULL]	
6	6	[NULL]	221	0.15	0.92	0.49	0.59	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2020-03-13	[NULL]	[NULL]	[NULL]	
7	3	[NULL]	245	0.86	0.5	0.5	0.5	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2019-12-14	[NULL]	[NULL]	[NULL]	
8	5	[NULL]	140	0.75	0.43	0.64	0.64	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2019-01-01	[NULL]	[NULL]	[NULL]	
9	4	[NULL]	278	0.64	0.09	0.66	0.09	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2020-10-01	[NULL]	[NULL]	[NULL]	
10	3	[NULL]	157	0.68	0.5	0.46	0.41	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2021-07-06	[NULL]	[NULL]	[NULL]	
11	2	[NULL]	228	0.7	0.99	0.5	0.5	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2024-11-11	[NULL]	[NULL]	[NULL]	
12	5	[NULL]	249	0.85	0.82	0.55	0.55	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2019-03-09	[NULL]	[NULL]	[NULL]	
13	5	[NULL]	292	0.96	0.02	0.85	0.48	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2019-02-08	[NULL]	[NULL]	[NULL]	
14	3	[NULL]	258	0.23	0.59	0.01	0.93	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2014-10-06	[NULL]	[NULL]	[NULL]	
15	7	[NULL]	276	0.15	0.94	0.95	0.36	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2011-05-18	[NULL]	[NULL]	[NULL]	
16	2	[NULL]	231	0.07	0.84	0.47	0.72	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2013-02-24	[NULL]	[NULL]	[NULL]	
17	5	[NULL]	225	0.53	0.99	0.72	0.29	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2021-07-06	[NULL]	[NULL]	[NULL]	
18	5	[NULL]	243	0.81	0.59	0.55	0.55	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2019-01-29	[NULL]	[NULL]	[NULL]	
19	7	[NULL]	213	0.71	0.87	0.98	0.62	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2015-10-13	[NULL]	[NULL]	[NULL]	
20	6	[NULL]	199	0.19	0.08	0.85	0.5	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2010-03-06	[NULL]	[NULL]	[NULL]	
21	4	[NULL]	198	0.37	0.3	0.64	0.64	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2021-10-04	[NULL]	[NULL]	[NULL]	
22	3	[NULL]	131	0.77	0.18	0.87	0.44	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2019-02-07	[NULL]	[NULL]	[NULL]	
23	5	[NULL]	248	0.86	0.62	0.95	0.85	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2019-01-27	[NULL]	[NULL]	[NULL]	
24	5	[NULL]	197	0.17	0.75	0.41	0.85	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2025-05-02	[NULL]	[NULL]	[NULL]	
25	6	[NULL]	199	0.62	0.94	0.58	0.86	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2015-10-10	[NULL]	[NULL]	[NULL]	
26	3	[NULL]	223	0.81	0.13	0.05	0.75	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2009-12-29	[NULL]	[NULL]	[NULL]	
27	3	[NULL]	254	0.93	0.15	0.05	0.75	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2022-09-21	[NULL]	[NULL]	[NULL]	
28	6	[NULL]	256	0.95	0.15	0.09	0.91	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2010-05-21	[NULL]	[NULL]	[NULL]	
29	6	[NULL]	216	0.36	0.45	0.7	0.89	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2019-05-21	[NULL]	[NULL]	[NULL]	
30	5	[NULL]	167	0.88	0.39	0.52	0.45	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2019-01-23	[NULL]	[NULL]	[NULL]	
31	5	[NULL]	157	0.68	0.5	0.48	0.91	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2024-11-09	[NULL]	[NULL]	[NULL]	
32	2	[NULL]	228	0.7	0.96	0.5	1	0	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2021-10-31	[NULL]	[NULL]	[NULL]
33	5	[NULL]	248	0.65	0.62	0.95	0.85	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2017-12-07	[NULL]	[NULL]	[NULL]	
34	5	[NULL]	292	0.96	0.02	0.85	0.48	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2019-02-08	[NULL]	[NULL]	[NULL]	
35	3	[NULL]	258	0.23	0.59	0.81	0.93	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2014-10-06	[NULL]	[NULL]	[NULL]	
36	7	[NULL]	276	0.15	0.94	0.47	0.59	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2011-05-19	[NULL]	[NULL]	[NULL]	
37	5	[NULL]	221	0.53	0.99	0.72	0.29	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2021-07-06	[NULL]	[NULL]	[NULL]	
38	6	[NULL]	243	0.22	0.6	0.06	0.28	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2019-10-15	[NULL]	[NULL]	[NULL]	
39	4	[NULL]	197	0.22	0.59	0.84	0.51	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2023-09-01	[NULL]	[NULL]	[NULL]	
40	3	[NULL]	198	0.92	0.66	0.81	0.81	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2020-07-27	[NULL]	[NULL]	[NULL]	

After

		int	song_credits	float	song_duration	float	valence	float	acousticness	float	danceability	float	energy	float	popularity	float	song_release_date	date	song_rating	float
1	2	5	[NULL]	279	0.54	0.99	0.93	0.48	0.48	0.72	0.98	0.42	0.20	0.19	0.99	2019-04-02	[NULL]	[NULL]	[NULL]	
2	6	[NULL]	143	0.59	0.75	0.57	0.74	0.42	0.57	0.54	0.36	0.64	0.40	0.59	0.36	2020-10-31	[NULL]	[NULL]	[NULL]	
3	11	[NULL]	241	0.18	0.99	0.47	0.59	0.36	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2021-04-25	[NULL]	[NULL]	[NULL]	
4	4	[NULL]	176	0.57	0.97	0.47	0.55	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2022-06-29	[NULL]	[NULL]	[NULL]	
5	6	[NULL]	300	0.40	0.97	0.47	0.55	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2015-10-13	[NULL]	[NULL]	[NULL]	
6	6	[NULL]	221	0.15	0.92	0.49	0.59	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2010-03-13	[NULL]	[NULL]	[NULL]	
7	3	[NULL]	245	0.86	0.5	0.5	0.5	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2019-12-14	[NULL]	[NULL]	[NULL]	
8	5	[NULL]	140	0.75	0.43	0.64	0.64	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2019-01-14	[NULL]	[NULL]	[NULL]	
9	4	[NULL]	278	0.64	0.09	0.66	0.09	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2020-07-27	[NULL]	[NULL]	[NULL]	
10	3	[NULL]	157	0.68	0.5	0.46	0.41	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2021-07-06	[NULL]	[NULL]	[NULL]	
11	2	[NULL]	228	0.7	0.99	0.5	1	0	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2024-11-11	[NULL]	[NULL]	[NULL]
12	5	[NULL]	249	0.85	0.82	0.55	0.55	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2019-03-09	[NULL]	[NULL]	[NULL]	
13	5	[NULL]	292	0.96	0.02	0.85	0.48	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2019-02-08	[NULL]	[NULL]	[NULL]	
14	3	[NULL]	258	0.23	0.59	0.81	0.93	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2014-10-06	[NULL]	[NULL]	[NULL]	
15	7	[NULL]	276	0.15	0.94	0.47	0.59	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2010-12-14	[NULL]	[NULL]	[NULL]	
16	2	[NULL]	221	0.53	0.99	0.72	0.29	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2021-07-06	[NULL]	[NULL]	[NULL]	
17	5	[NULL]	225	0.53	0.99	0.72	0.29	0.35	0.38	0.64	0.0	0.20	0.45	0.53	0.37	2021-07-06	[NULL]	[NULL]	[NULL]	
18	5	[NULL]	243	0.22	0.6	0.06	0.28	0.35	0.38	0.64	0.0</									

Parameter																																																																																																																																																																
Deskripsi	Menghitung monthly_listeners suatu artis. Dalam bentuk function karena merupakan sebuah cron.																																																																																																																																																															
Script SQL																																																																																																																																																																
<pre>CREATE OR REPLACE FUNCTION recalculate_monthly_listeners() RETURNS VOID AS \$\$\$ BEGIN UPDATE artists a SET monthly_listener_count = sub.cnt FROM (SELECT cs.artist_id, COUNT(DISTINCT l.user_id) AS cnt FROM listens l JOIN create_songs cs ON cs.song_id = l.song_id WHERE DATE_TRUNC('month', l."TIMESTAMP") = DATE_TRUNC('month', CURRENT_DATE) GROUP BY cs.artist_id) sub WHERE a.artist_id = sub.artist_id; END; \$\$ LANGUAGE plpgsql;</pre>																																																																																																																																																																
Screenshot Hasil																																																																																																																																																																
<p>Before</p> <table border="1"> <thead> <tr> <th>Index</th> <th>Artist ID</th> <th>Bio</th> <th>Monthly Listener Count</th> <th>Artist PF</th> <th>Artist Email</th> <th>Banner</th> <th>Follower Count</th> </tr> </thead> <tbody> <tr><td>1</td><td>Dolorum deleniti</td><td>ipsum in, iure sunt asperiores repellat sequi. Amet volu-</td><td>509,855</td><td>/uploads/artists/artist1.jpg</td><td>Aswani Waskita_1@example.com</td><td>[NULL]</td><td>[NULL]</td></tr> <tr><td>2</td><td>Architecto architecto</td><td>similique illis. Doloribus nisi nulla in reincident minimu</td><td>475,371</td><td>/uploads/artists/artist2.jpg</td><td>Drs. Endah Hakim, M.TI_2@example.com</td><td>[NULL]</td><td>[NULL]</td></tr> <tr><td>3</td><td>Elegiend distinctio</td><td>exercitatione porro sequi. Laborum temporibus sapier</td><td>6,401</td><td>/uploads/artists/artist3.jpg</td><td>Paiman Puspasari_3@example.com</td><td>[NULL]</td><td>[NULL]</td></tr> <tr><td>4</td><td>Quiibusdam unde</td><td>dolorum cum reincident debitis impedit debitis. Quos ille</td><td>363,585</td><td>/uploads/artists/artist4.jpg</td><td>Garang Purwanti_4@example.com</td><td>[NULL]</td><td>[NULL]</td></tr> <tr><td>5</td><td>Voluptatum beatae</td><td>delectus delectus delectus delectus delectus rem. Fugiat</td><td>108,835</td><td>/uploads/artists/artist5.jpg</td><td>dr. Agustina Sari_5@example.com</td><td>[NULL]</td><td>[NULL]</td></tr> <tr><td>6</td><td>Eiusmod et nulla at. Perferendis dolore dicta et ad</td><td>adipisci. Ut enim asperiores</td><td>238,630</td><td>/uploads/artists/artist6.jpg</td><td>dr. Eddy Muliawan_6@example.com</td><td>[NULL]</td><td>[NULL]</td></tr> <tr><td>7</td><td>Eiarum asperatur</td><td>enim earum eum. Aspernatur corrupti fugit corrupti di</td><td>688,206</td><td>/uploads/artists/artist7.jpg</td><td>Hj. Ika Haryanti, M.Kom_7@example.com</td><td>[NULL]</td><td>[NULL]</td></tr> <tr><td>8</td><td>Accusantium molestias</td><td>voluptatum expedita officiis quis officiis. Veniam</td><td>378,046</td><td>/uploads/artists/artist8.jpg</td><td>dr. Estiwan Andrians, S.Ti_8@example.com</td><td>[NULL]</td><td>[NULL]</td></tr> <tr><td>9</td><td>Officia accusamus</td><td>in autem inventore facere tempora. Provident ips</td><td>186,963</td><td>/uploads/artists/artist9.jpg</td><td>dr. Mira Uwais_9@example.com</td><td>[NULL]</td><td>[NULL]</td></tr> <tr><td>10</td><td>Assumenda quam</td><td>laborum quidem porro. Ipsam accusantium eos illo dol</td><td>834,772</td><td>/uploads/artists/artist10.jpg</td><td>Puti Anggrilwan_10@example.com</td><td>[NULL]</td><td>[NULL]</td></tr> <tr><td>11</td><td>Nihil dicta beatae</td><td>dicta corrupti equal consequuntur. Cumque esse</td><td>302,030</td><td>/uploads/artists/artist11.jpg</td><td>Ema Simorang, Tg@example.com</td><td>[NULL]</td><td>[NULL]</td></tr> <tr><td>12</td><td>Aliquid quodammodo</td><td>consequuntur. Voluptate necessitatibus</td><td>606,079</td><td>/uploads/artists/artist12.jpg</td><td>Belinda Pradipta_12@example.com</td><td>[NULL]</td><td>[NULL]</td></tr> <tr><td>13</td><td>Sunt laborum adipisci</td><td>occaccied recidivis quia. Molitia mollitia et volupt</td><td>74,477</td><td>/uploads/artists/artist13.jpg</td><td>Pia Miftah_13@example.com</td><td>[NULL]</td><td>[NULL]</td></tr> <tr><td>14</td><td>Voluptates recusandae</td><td>officis corrupti vitae commod. Ex aut sapiente ai</td><td>543,141</td><td>/uploads/artists/artist14.jpg</td><td>Uki Kurniawati_14@example.com</td><td>[NULL]</td><td>[NULL]</td></tr> <tr><td>15</td><td>Eveniet animi facere.</td><td>*Doloribus illum placeat debitis. Aspernatur sit nulla</td><td>657,175</td><td>/uploads/artists/artist15.jpg</td><td>Puti Maimunah Martun, S.Pd_15@example.com</td><td>[NULL]</td><td>[NULL]</td></tr> <tr><td>16</td><td>Voluptatibus eos</td><td>voluptate omnis earum. Sint quod voluptate tenerit quia</td><td>728,793</td><td>/uploads/artists/artist16.jpg</td><td>Queen Tamila_17@example.com</td><td>[NULL]</td><td>[NULL]</td></tr> <tr><td>17</td><td>Voluptatis eos sit</td><td>rem iusto asperiores quidem. Similique quam voluptatem</td><td>711,460</td><td>/uploads/artists/artist17.jpg</td><td>Yance Martun_18@example.com</td><td>[NULL]</td><td>[NULL]</td></tr> <tr><td>18</td><td>Ipsam dolorencere</td><td>facilis voluptatum facilis temporibus. Aperiam eius vol</td><td>62,756</td><td>/uploads/artists/artist18.jpg</td><td>Ami Uwais_19@example.com</td><td>[NULL]</td><td>[NULL]</td></tr> <tr><td>19</td><td>Optio ad accusamus</td><td>ratione. Nihil accusamus exque conetur it;lab</td><td>467,551</td><td>/uploads/artists/artist19.jpg</td><td>Kayun Ramadhan_20@example.com</td><td>[NULL]</td><td>[NULL]</td></tr> </tbody> </table>	Index	Artist ID	Bio	Monthly Listener Count	Artist PF	Artist Email	Banner	Follower Count	1	Dolorum deleniti	ipsum in, iure sunt asperiores repellat sequi. Amet volu-	509,855	/uploads/artists/artist1.jpg	Aswani Waskita_1@example.com	[NULL]	[NULL]	2	Architecto architecto	similique illis. Doloribus nisi nulla in reincident minimu	475,371	/uploads/artists/artist2.jpg	Drs. Endah Hakim, M.TI_2@example.com	[NULL]	[NULL]	3	Elegiend distinctio	exercitatione porro sequi. Laborum temporibus sapier	6,401	/uploads/artists/artist3.jpg	Paiman Puspasari_3@example.com	[NULL]	[NULL]	4	Quiibusdam unde	dolorum cum reincident debitis impedit debitis. Quos ille	363,585	/uploads/artists/artist4.jpg	Garang Purwanti_4@example.com	[NULL]	[NULL]	5	Voluptatum beatae	delectus delectus delectus delectus delectus rem. Fugiat	108,835	/uploads/artists/artist5.jpg	dr. Agustina Sari_5@example.com	[NULL]	[NULL]	6	Eiusmod et nulla at. Perferendis dolore dicta et ad	adipisci. Ut enim asperiores	238,630	/uploads/artists/artist6.jpg	dr. Eddy Muliawan_6@example.com	[NULL]	[NULL]	7	Eiarum asperatur	enim earum eum. Aspernatur corrupti fugit corrupti di	688,206	/uploads/artists/artist7.jpg	Hj. Ika Haryanti, M.Kom_7@example.com	[NULL]	[NULL]	8	Accusantium molestias	voluptatum expedita officiis quis officiis. Veniam	378,046	/uploads/artists/artist8.jpg	dr. Estiwan Andrians, S.Ti_8@example.com	[NULL]	[NULL]	9	Officia accusamus	in autem inventore facere tempora. Provident ips	186,963	/uploads/artists/artist9.jpg	dr. Mira Uwais_9@example.com	[NULL]	[NULL]	10	Assumenda quam	laborum quidem porro. Ipsam accusantium eos illo dol	834,772	/uploads/artists/artist10.jpg	Puti Anggrilwan_10@example.com	[NULL]	[NULL]	11	Nihil dicta beatae	dicta corrupti equal consequuntur. Cumque esse	302,030	/uploads/artists/artist11.jpg	Ema Simorang, Tg@example.com	[NULL]	[NULL]	12	Aliquid quodammodo	consequuntur. Voluptate necessitatibus	606,079	/uploads/artists/artist12.jpg	Belinda Pradipta_12@example.com	[NULL]	[NULL]	13	Sunt laborum adipisci	occaccied recidivis quia. Molitia mollitia et volupt	74,477	/uploads/artists/artist13.jpg	Pia Miftah_13@example.com	[NULL]	[NULL]	14	Voluptates recusandae	officis corrupti vitae commod. Ex aut sapiente ai	543,141	/uploads/artists/artist14.jpg	Uki Kurniawati_14@example.com	[NULL]	[NULL]	15	Eveniet animi facere.	*Doloribus illum placeat debitis. Aspernatur sit nulla	657,175	/uploads/artists/artist15.jpg	Puti Maimunah Martun, S.Pd_15@example.com	[NULL]	[NULL]	16	Voluptatibus eos	voluptate omnis earum. Sint quod voluptate tenerit quia	728,793	/uploads/artists/artist16.jpg	Queen Tamila_17@example.com	[NULL]	[NULL]	17	Voluptatis eos sit	rem iusto asperiores quidem. Similique quam voluptatem	711,460	/uploads/artists/artist17.jpg	Yance Martun_18@example.com	[NULL]	[NULL]	18	Ipsam dolorencere	facilis voluptatum facilis temporibus. Aperiam eius vol	62,756	/uploads/artists/artist18.jpg	Ami Uwais_19@example.com	[NULL]	[NULL]	19	Optio ad accusamus	ratione. Nihil accusamus exque conetur it;lab	467,551	/uploads/artists/artist19.jpg	Kayun Ramadhan_20@example.com	[NULL]	[NULL]
Index	Artist ID	Bio	Monthly Listener Count	Artist PF	Artist Email	Banner	Follower Count																																																																																																																																																									
1	Dolorum deleniti	ipsum in, iure sunt asperiores repellat sequi. Amet volu-	509,855	/uploads/artists/artist1.jpg	Aswani Waskita_1@example.com	[NULL]	[NULL]																																																																																																																																																									
2	Architecto architecto	similique illis. Doloribus nisi nulla in reincident minimu	475,371	/uploads/artists/artist2.jpg	Drs. Endah Hakim, M.TI_2@example.com	[NULL]	[NULL]																																																																																																																																																									
3	Elegiend distinctio	exercitatione porro sequi. Laborum temporibus sapier	6,401	/uploads/artists/artist3.jpg	Paiman Puspasari_3@example.com	[NULL]	[NULL]																																																																																																																																																									
4	Quiibusdam unde	dolorum cum reincident debitis impedit debitis. Quos ille	363,585	/uploads/artists/artist4.jpg	Garang Purwanti_4@example.com	[NULL]	[NULL]																																																																																																																																																									
5	Voluptatum beatae	delectus delectus delectus delectus delectus rem. Fugiat	108,835	/uploads/artists/artist5.jpg	dr. Agustina Sari_5@example.com	[NULL]	[NULL]																																																																																																																																																									
6	Eiusmod et nulla at. Perferendis dolore dicta et ad	adipisci. Ut enim asperiores	238,630	/uploads/artists/artist6.jpg	dr. Eddy Muliawan_6@example.com	[NULL]	[NULL]																																																																																																																																																									
7	Eiarum asperatur	enim earum eum. Aspernatur corrupti fugit corrupti di	688,206	/uploads/artists/artist7.jpg	Hj. Ika Haryanti, M.Kom_7@example.com	[NULL]	[NULL]																																																																																																																																																									
8	Accusantium molestias	voluptatum expedita officiis quis officiis. Veniam	378,046	/uploads/artists/artist8.jpg	dr. Estiwan Andrians, S.Ti_8@example.com	[NULL]	[NULL]																																																																																																																																																									
9	Officia accusamus	in autem inventore facere tempora. Provident ips	186,963	/uploads/artists/artist9.jpg	dr. Mira Uwais_9@example.com	[NULL]	[NULL]																																																																																																																																																									
10	Assumenda quam	laborum quidem porro. Ipsam accusantium eos illo dol	834,772	/uploads/artists/artist10.jpg	Puti Anggrilwan_10@example.com	[NULL]	[NULL]																																																																																																																																																									
11	Nihil dicta beatae	dicta corrupti equal consequuntur. Cumque esse	302,030	/uploads/artists/artist11.jpg	Ema Simorang, Tg@example.com	[NULL]	[NULL]																																																																																																																																																									
12	Aliquid quodammodo	consequuntur. Voluptate necessitatibus	606,079	/uploads/artists/artist12.jpg	Belinda Pradipta_12@example.com	[NULL]	[NULL]																																																																																																																																																									
13	Sunt laborum adipisci	occaccied recidivis quia. Molitia mollitia et volupt	74,477	/uploads/artists/artist13.jpg	Pia Miftah_13@example.com	[NULL]	[NULL]																																																																																																																																																									
14	Voluptates recusandae	officis corrupti vitae commod. Ex aut sapiente ai	543,141	/uploads/artists/artist14.jpg	Uki Kurniawati_14@example.com	[NULL]	[NULL]																																																																																																																																																									
15	Eveniet animi facere.	*Doloribus illum placeat debitis. Aspernatur sit nulla	657,175	/uploads/artists/artist15.jpg	Puti Maimunah Martun, S.Pd_15@example.com	[NULL]	[NULL]																																																																																																																																																									
16	Voluptatibus eos	voluptate omnis earum. Sint quod voluptate tenerit quia	728,793	/uploads/artists/artist16.jpg	Queen Tamila_17@example.com	[NULL]	[NULL]																																																																																																																																																									
17	Voluptatis eos sit	rem iusto asperiores quidem. Similique quam voluptatem	711,460	/uploads/artists/artist17.jpg	Yance Martun_18@example.com	[NULL]	[NULL]																																																																																																																																																									
18	Ipsam dolorencere	facilis voluptatum facilis temporibus. Aperiam eius vol	62,756	/uploads/artists/artist18.jpg	Ami Uwais_19@example.com	[NULL]	[NULL]																																																																																																																																																									
19	Optio ad accusamus	ratione. Nihil accusamus exque conetur it;lab	467,551	/uploads/artists/artist19.jpg	Kayun Ramadhan_20@example.com	[NULL]	[NULL]																																																																																																																																																									



Tabel 107 PL/SQL - Function update_prerelease_daily

Nama Function	update_prerelease_daily
Parameter	
Deskripsi	Mengubah value isPrerelease suatu collection jika sudah lewat tanggal rilis collection tersebut. Dalam bentuk function karena merupakan sebuah cron.
Script SQL	
<pre>CREATE OR REPLACE FUNCTION update_prerelease_daily() RETURNS VOID AS \$\$ BEGIN UPDATE collections SET isPrerelease = FALSE WHERE isPrerelease = TRUE AND collection_release_date <= CURRENT_DATE; END; \$\$ LANGUAGE plpgsql;</pre>	

Screenshot Hasil

Before						
After						

Record	37	38	Iste animi rerum	EP	/uploads/covers/coll_38.jpg		2021-04-08	[NULL]	[NULL]
	38	39	Nostrum magni	Compilation	/uploads/covers/coll_39.jpg		2023-03-14	[NULL]	[NULL]
	39	40	Veritatis qui aliquam	Compilation	/uploads/covers/coll_40.jpg		2025-06-06	[NULL]	[NULL]
	40	1	Inventore suscipit	Single	/uploads/covers/coll_1.jpg		2022-12-15	[NULL]	[]

Tabel 108 PL/SQL - Function get_collection_genres

Nama Function	get_collection_genres
Parameter	p_collection_id INT
Deskripsi	Function ini bertujuan untuk mengambil daftar genre utama
Script SQL	
<pre>CREATE OR REPLACE FUNCTION get_collection_genres(p_collection_id INT) RETURNS TABLE(genre_id INT, genre_name VARCHAR) LANGUAGE plpgsql AS \$\$\$ BEGIN RETURN QUERY SELECT g.genre_id, g.genre_name FROM COLLECTION_TOP_3_GENRES ct3g JOIN GENRES g ON g.genre_id = ct3g.genre_id WHERE ct3g.collection_id = p_collection_id; END; \$\$;</pre>	

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface with the following details:

- Left Panel:** Shows the database structure under "Servers(1) > SpotifyPostgres > Tables(27)".
- Top Bar:** Contains "File", "Object", "Tools", "Edit", "View", "Window", "Help".
- Central Area:**
 - Query Tab:** Displays the SQL code for the function definition.
 - Data Output Tab:** Shows the results of the function execution.
- Code in Query Tab:**

```
CREATE OR REPLACE FUNCTION get_collection_genres(p_collection_id INT)
RETURNS TABLE(genre_id INT, genre_name VARCHAR)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT
        g.genre_id,
        g.genre_name
    FROM COLLECTION_TOP_3_GENRES ct3g
    JOIN GENRES g ON g.genre_id = ct3g.genre_id
    WHERE ct3g.collection_id = p_collection_id;
END;
$$;
```
- Results in Data Output Tab:**

genre_id	genre_name
2	anime
9	sci-fi

Tabel 109 PL/SQL - Function get_collection_average_rating

Nama Function	get_collection_average_rating
Parameter	p_collection_id INT
Deskripsi	Function ini berfungsi untuk mengambil hasil average rating dari trigger
Script SQL	
<pre>CREATE OR REPLACE FUNCTION get_collection_average_rating(p_collection_id INT) RETURNS NUMERIC AS \$\$ DECLARE v_rating NUMERIC; BEGIN -- Mengambil nilai yang sudah dihitung oleh trigger SELECT COLLECTION_RATING INTO v_rating FROM COLLECTIONS WHERE COLLECTION_ID = p_collection_id; RETURN COALESCE(v_rating, 0); END; \$\$ LANGUAGE plpgsql;</pre>	

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface with the following details:

- Left Panel (Object Explorer):** Shows the database structure under "Servers (1) / PostgreSQL 17 / Database (0)".
- Center Panel (Query Editor):** Displays the function definition:


```
CREATE OR REPLACE FUNCTION get_collection_average_rating(p_collection_id INT)
RETURNS NUMERIC AS
$$
DECLARE
    v_rating NUMERIC;
BEGIN
    -- Mengambil nilai yang sudah dihitung oleh trigger
    SELECT COLLECTION_RATING
    INTO v_rating
    FROM COLLECTIONS
    WHERE COLLECTION_ID = p_collection_id;

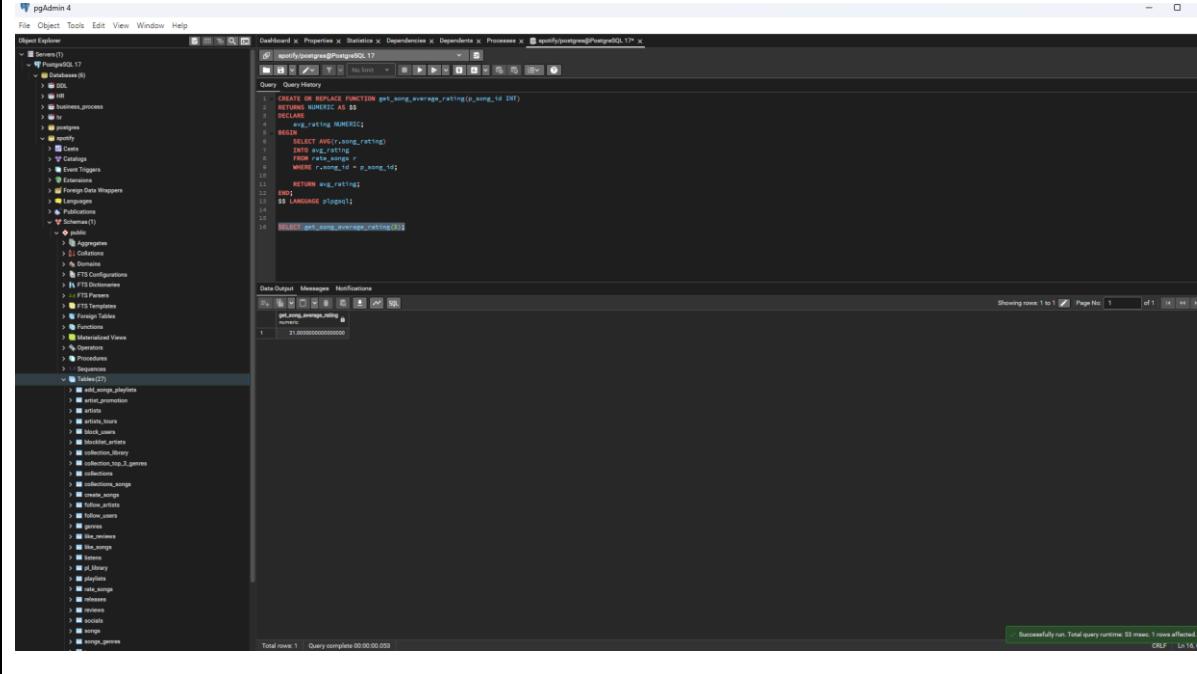
    RETURN COALESCE(v_rating, 0);
END;
$$ LANGUAGE plpgsql;
```
- Bottom Panel (Data Output):** Shows the result of executing the function:


```
get_collection_average_rating
1
```
- Status Bar:** Shows "Successfully ran. Total query runtime: 83 msec. 1 rows affected." and "CMD - Pg 18.03".

Tabel 110 PL/SQL - Function get_song_average_rating

Nama Function	get_song_average_rating(p_song_id INT)
Parameter	p_song_id INT
Deskripsi	Function ini berfungsi untuk menghitung rata rata song dari rate_songs
Script SQL	
<pre>CREATE OR REPLACE FUNCTION get_song_average_rating(p_song_id INT) RETURNS NUMERIC AS \$\$ DECLARE avg_rating NUMERIC; BEGIN SELECT AVG(r.song_rating) INTO avg_rating FROM rate_songs r WHERE r.song_id = p_song_id; RETURN avg_rating; END; \$\$ LANGUAGE plpgsql;</pre>	

Screenshot Hasil



The screenshot shows the pgAdmin 4 interface with the 'Query' tab selected. The query window contains the PL/SQL code for creating the function:

```
CREATE OR REPLACE FUNCTION get_song_average_rating(p_song_id INT)
RETURNS NUMERIC AS $$
DECLARE
    avg_rating NUMERIC;
BEGIN
    SELECT AVG(r.song_rating)
    INTO avg_rating
    FROM rate_songs r
    WHERE r.song_id = p_song_id;

    RETURN avg_rating;
END;
$$ LANGUAGE plpgsql;
```

The results pane shows the output of the function call:

```
get_song_average_rating
1 21.000000000000000
```

At the bottom right, a message indicates the operation was successful.

Tabel 111 PL/SQL - Function get_playlist_duration_minutes

Nama Function	get_playlist_duration_minutes(p_playlist_id INT)
Parameter	p_playlist_id INT
Deskripsi	Function ini berfungsi untuk menjumlahkan durasi seluruh lagu dalam satu playlist
Script SQL	
<pre>CREATE OR REPLACE FUNCTION get_playlist_duration_minutes(p_playlist_id INT) RETURNS INT AS \$\$ DECLARE v_total_sec INT; BEGIN SELECT SUM(s.song_duration) INTO v_total_sec FROM songs s JOIN add_songs_playlists.asp ON s.song_id = asp.song_id WHERE asp.playlist_id = p_playlist_id; RETURN COALESCE(v_total_sec / 60, 0); END; \$\$ LANGUAGE plpgsql;</pre>	

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface with the following details:

- File Tree:** Shows the database structure with servers, databases, tables, and functions.
- Current Window:** Displays the SQL editor with the function definition:


```
CREATE OR REPLACE FUNCTION get_playlist_duration_minutes(p_playlist_id INT)
RETURNS INT AS $$

DECLARE
    v_total_sec INT;
BEGIN
    SELECT SUM(s.song_duration)
    INTO v_total_sec
    FROM songs s
    JOIN add_songs_playlists.asp ON s.song_id = asp.song_id
    WHERE asp.playlist_id = p_playlist_id;

    RETURN COALESCE(v_total_sec / 60, 0);
END;
$$ LANGUAGE plpgsql;
```
- Data Output:** Shows the result of the function call:

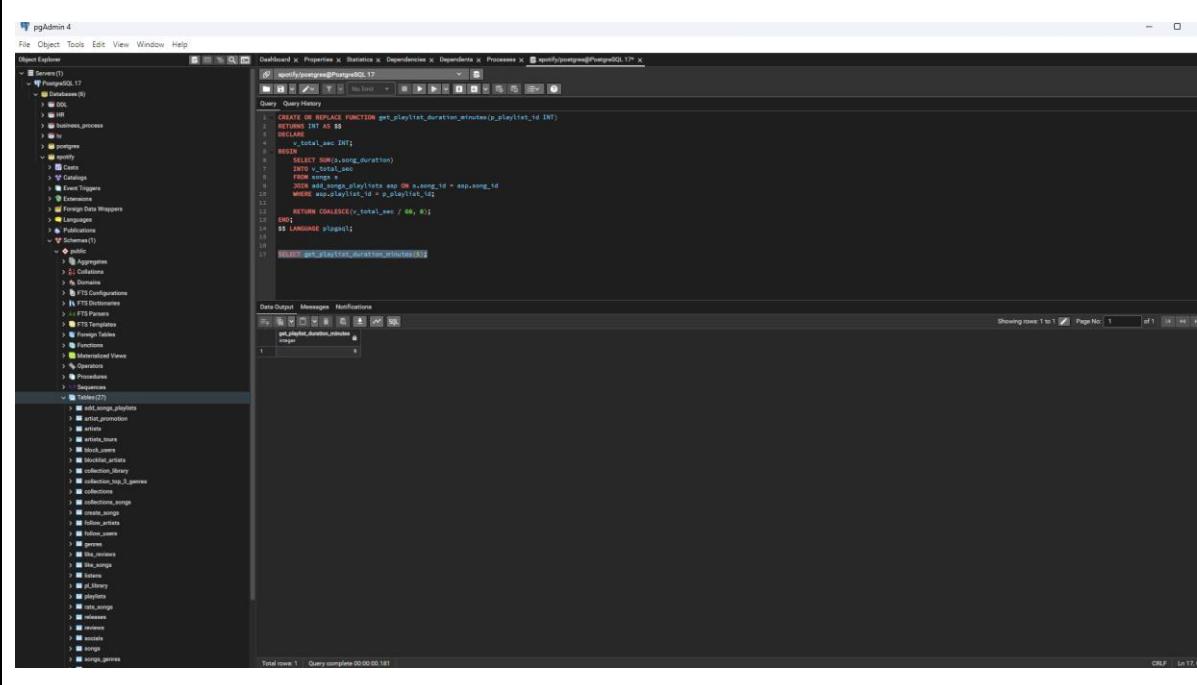

```
SELECT get_playlist_duration_minutes(1);
```

 Result: 9
- Message Bar:** Shows "Showing rows 1 to 1" and "Page No: 1 of 1".
- Bottom Status:** Shows "Total rows: 1 Query complete 00:00:181" and "CRLF Ln 17, Col 1".

Tabel 112 PL/SQL - Function get_artist_total_plays

Nama Function	get_artist_total_plays(p_artist_id INT)
Parameter	p_artist_id INT
Deskripsi	Function ini berfungsi untuk menghitung total jumlah stream suatu artists
Script SQL	
<pre>CREATE OR REPLACE FUNCTION get_artist_total_plays(p_artist_id INT) RETURNS BIGINT AS \$\$ DECLARE v_total_plays BIGINT; BEGIN SELECT COUNT(l.listen_id) INTO v_total_plays FROM create_songs cs JOIN listens l ON cs.song_id = l.song_id WHERE cs.artist_id = p_artist_id; RETURN COALESCE(v_total_plays, 0); END; \$\$ LANGUAGE plpgsql;</pre>	

Screenshot Hasil



The screenshot shows the pgAdmin 4 interface with the 'Query' tab selected. The query window contains the PL/SQL code for creating the function:

```
CREATE OR REPLACE FUNCTION get_artist_total_plays(p_artist_id INT)
RETURNS BIGINT AS $$

DECLARE
    v_total_plays BIGINT;
BEGIN
    SELECT COUNT(l.listen_id)
    INTO v_total_plays
    FROM create_songs cs
    JOIN listens l ON cs.song_id = l.song_id
    WHERE cs.artist_id = p_artist_id;

    RETURN COALESCE(v_total_plays, 0);
END;
$$ LANGUAGE plpgsql;
```

The results pane shows the output of the function:

```
+-----+
| get_artist_total_plays |
+-----+
|                      |
|          0             |
|                      |
+-----+
```

At the bottom of the pgAdmin window, the status bar indicates "Total rows: 1 | Query completed: 00:00:00.181".

Tabel 113 PL/SQL - Function get_album_total_duration

Nama Function	get_album_total_duration(p_collection_id INT)
Parameter	p_collection_id INT
Deskripsi	Function ini berfungsi untuk menghitung durasi dari suatu album
Script SQL	

```

CREATE OR REPLACE FUNCTION get_album_total_duration(p_collection_id INT)
RETURNS INT AS $$

DECLARE
    v_total_seconds INT;
BEGIN
    -- Aggregation: SUM (Menjumlahkan nilai durasi)
    SELECT SUM(s.song_duration)
    INTO v_total_seconds
    FROM collections_songs cs
    JOIN songs s ON cs.song_id = s.song_id
    WHERE cs.collection_id = p_collection_id;

    RETURN COALESCE(v_total_seconds, 0);
END;
$$ LANGUAGE plpgsql;

```

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface with the following details:

- File Bar:** File, Object, Tools, Edit, View, Window, Help.
- Object Explorer:** Shows the database structure under "PostgreSQL (B)".
- Query Editor:** Contains the SQL code for creating the function.
- Data Output:** Shows the result of the query, which is an empty table named "get_album_total_duration" with one row containing the value 245.
- Status Bar:** Shows "Total rows: 1 Query complete 00:00:00.148".
- Bottom Right:** A green status bar with the message "Successfully run. Total query runtime: 148 msec. 1 rows affected CRLF Lin 19 Col 1".

Tabel 113 PL/SQL - Trigger validate_tour_date

Nama Trigger	validate_tour_date
Aksi	Melakukan validasi otomatis terhadap kolom tour_date setiap kali data pada tabel tours ditambahkan atau diperbarui. Trigger ini memastikan bahwa tanggal tour tidak boleh berada di masa lalu. Jika tanggal yang dimasukkan lebih kecil dari tanggal hari ini (CURRENT_DATE), proses akan dibatalkan dan sistem melempar error.
Jenis	BEFORE INSERT OR UPDATE
Deskripsi	<p>Trigger ini digunakan untuk menjamin bahwa setiap jadwal tour (konser, event, atau show) hanya dapat dibuat dengan tanggal yang valid, yaitu hari ini atau masa depan.</p> <p>Setiap ada operasi INSERT atau UPDATE pada tabel tours, trigger akan memanggil function validate_tour_date(), yang kemudian melakukan pengecekan:</p> <ol style="list-style-type: none"> 1. Jika tour_date < CURRENT_DATE, maka fungsi akan mengeluarkan exception yang menghentikan proses insert/update. 2. Jika tanggal valid, baris data akan dilanjutkan untuk diproses. <p>Trigger ini memastikan integritas data sehingga tidak ada tour yang terdaftar dengan tanggal kedaluwarsa.</p>

Script SQL

```
CREATE OR REPLACE FUNCTION validate_tour_date()
RETURNS TRIGGER AS $$ 
BEGIN
IF NEW.tour_date < CURRENT_DATE THEN
RAISE EXCEPTION 'Tour date (%s) shouldnt be before today
(%s)', NEW.tour_date, CURRENT_DATE;
END IF;

RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_validate_tour_date
BEFORE INSERT OR UPDATE ON tours
FOR EACH ROW
EXECUTE FUNCTION validate_tour_date();
```

Screenshot Hasil

```

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
> blocklist_artists
> collection_library
> collection_top_3_genres
> collections
> collections_songs
> create_songs
> follow_artists
> follow_users
> genres
> like_reviews
> like_songs
> listens
> pl_library
> playlists
> rate_songs
> releases
> reviews
> socials
> songs
  > Columns(13)
    song_id
    song_file
    song_title
    listen_count
    song_credits
Query Query History
4426      FALSE      -- isPrerelease FALSE
4427  );
4428
4429
4430
4431  CREATE OR REPLACE FUNCTION validate_tour_date()
4432  RETURNS TRIGGER AS $$%
4433  BEGIN
4434    IF NEW.tour_date < CURRENT_DATE THEN
4435      RAISE EXCEPTION 'Tour date (%) shouldnt be before today (%)';
4436    END IF;
4437
Data Output Messages Notifications
CREATE FUNCTION
Query returned successfully in 677 msec.

Total rows:  Query complete 0:00:00.677
CRLF Ln 4431, Col 1
10:51 27/11/2025

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
> blocklist_artists
> collection_library
> collection_top_3_genres
> collections
> collections_songs
> create_songs
> follow_artists
> follow_users
> genres
> like_reviews
> like_songs
> listens
> pl_library
> playlists
> rate_songs
> releases
> reviews
> socials
> songs
  > Columns(13)
    song_id
    song_file
    song_title
    listen_count
    song_credits
Query Query History
4442
4443  CREATE TRIGGER trg_validate_tour_date
4444    BEFORE INSERT OR UPDATE ON tours
4445    FOR EACH ROW
4446    EXECUTE FUNCTION validate_tour_date();
4447
4448
4449
4450
4451
4452  -- cek (song)
Data Output Messages Notifications
CREATE TRIGGER
Query returned successfully in 97 msec.

Total rows:  Query complete 0:00:00.097
CRLF Ln 4443, Col 1
10:51 27/11/2025

```

Tabel 115 PL/SQL - Trigger trg_update_collection_top3_genres

Nama Trigger	trg_update_collection_top3_genres
Aksi	Update otomatis tabel collection_top_3_genres saat ada perubahan atau penambahan lagu di collections_songs.
Jenis	AFTER INSERT OR UPDATE

<p>Deskripsi</p>	<p>Trigger ini akan secara otomatis menghitung 3 genre teratas yang paling sering muncul dari seluruh lagu dalam sebuah collection setiap kali ada lagu baru ditambahkan atau update di tabel collections_songs.</p> <p>Langkah kerjanya:</p> <ol style="list-style-type: none"> 1. Hapus data lama untuk collection terkait di tabel collection_top_3_genres. 2. Hitung jumlah kemunculan tiap genre dari lagu-lagu collection menggunakan tabel songs_genres. 3. Ambil 3 genre terbanyak dan masukkan ke tabel collection_top_3_genres. 4. Trigger dijalankan per baris (row-level), sehingga setiap lagu yang diinsert/update langsung memicu perhitungan ulang. <p>Contoh Perhitungan :</p> <p>Misalkan sebuah collection memiliki 4 lagu dengan genre sebagai berikut:</p> <p>Lagu 1: Pop, Jazz</p> <p>Lagu 2: Pop, Rock, Noise Pop</p> <p>Lagu 3: Pop, Rock</p> <p>Lagu 4: Pop, Jazz</p> <p>Maka jumlah kemunculan genre:</p> <p>Pop = 4</p> <p>Jazz = 2</p> <p>Rock = 2</p>
-------------------------	--

	<p>Noise Pop = 1</p> <p>Top 3 genre (Pop, Jazz, Rock) akan di-insert ke collection_top_3_genres.</p>
Script SQL	
	<pre> CREATE OR REPLACE FUNCTION update_collection_top3_genres() RETURNS TRIGGER AS \$\$ DECLARE top_genre RECORD; BEGIN -- Hapus data lama untuk collection ini DELETE FROM collection_top_3_genres WHERE collection_id = NEW.collection_id; -- Ambil 3 genre paling sering muncul dari lagu-lagu collection FOR top_genre IN SELECT sg.genre_id FROM songs_genres sg JOIN collections_songs cs ON sg.song_id = cs.song_id -- Menghubungkan genre dengan lagu yang masuk collection WHERE cs.collection_id = NEW.collection_id GROUP BY sg.genre_id -- Kelompokkan berdasarkan genre ORDER BY COUNT(*) DESC -- Mengurutkan genre paling sering muncul -> paling banyak lagu LIMIT 3 -- Ambil 3 teratas LOOP -- Insert ke COLLECTION_TOP_3_GENRES END; \$\$ LANGUAGE plpgsql; </pre>

```

        INSERT INTO collection_top_3_genres (collection_id,
genre_id)
    VALUES (NEW.collection_id, top_genre.genre_id);
END LOOP;

RETURN NEW;
END;
$$ LANGUAGE plpgsql;

```

Screenshot Hasil

The screenshot shows two pgAdmin 4 sessions. The top session shows the creation of a function:

```

-- Update_collection_top3_genres
CREATE OR REPLACE FUNCTION update_collection_top3_genres()
RETURNS TRIGGER AS $$
DECLARE
    top_genre RECORD;
BEGIN
    -- Hapus data lama untuk collection ini
    DELETE FROM collection_top_3_genres
    WHERE collection_id = NEW.collection_id;
    ...

```

The bottom session shows the creation of a trigger:

```

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
-- trigger
CREATE TRIGGER trg_update_collection_top3_genres
AFTER INSERT OR UPDATE ON collections_songs
FOR EACH ROW
EXECUTE FUNCTION update_collection_top3_genres();

```

```

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer Final_Destination/postgres@Postgres16
    songs_genres
        Columns(2)
            genre_id
            song_id
        Constraints
        Indexes
        RLS Policies
        Rules
        Triggers
    tours
        Columns(4)
            tour_id
            tour_date
            tour_name
            venue
        Constraints
        Indexes
        RLS Policies
        Rules
        Triggers
    users
        Columns(7)
            user_id
            username
            user_pfp
Query Query History
3701  SELECT * FROM collections_songs;
3702  -- cek
3703  -- Update supaya trigger jalan
3704  UPDATE collections_songs
3705  SET nomor_track = nomor_track
3706  WHERE collection_id = 1;
3707
3708  -- Lihat hasil top 3 genre
3709  SELECT * FROM collection_top_3_genres
3710  WHERE collection_id = 1;
3711

Data Output Messages Notifications
UPDATE 8
Query returned successfully in 193 msec.

Total rows: 8 Query complete 00:00:00.193 CRLF Ln 3701, Col 1

```



```

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer Final_Destination/postgres@Postgres16
    songs_genres
        Columns(2)
            genre_id
            song_id
        Constraints
        Indexes
        RLS Policies
        Rules
        Triggers
    tours
        Columns(4)
            tour_id
            tour_date
            tour_name
            venue
        Constraints
        Indexes
        RLS Policies
        Rules
        Triggers
    users
        Columns(7)
            user_id
            username
            user_pfp
Query Query History
3701  SELECT * FROM collections_songs;
3702  -- cek
3703  -- Update supaya trigger jalan
3704  UPDATE collections_songs
3705  SET nomor_track = nomor_track
3706  WHERE collection_id = 1;
3707
3708  -- Lihat hasil top 3 genre
3709  SELECT * FROM collection_top_3_genres
3710  WHERE collection_id = 1;
3711

Data Output Messages Notifications
Showing rows: 1 to 3 Page No: 1 of 1 << >> >>
collection_id genre_id
1             1         2
2             1         6
3             1         3

Total rows: 3 Query complete 00:00:00.675 CRLF Ln 3708, Col 1

```

Tabel 116 PL/SQL - Trigger trg_update_song_rating

Nama Trigger	Trg_update_song_rating
Aksi	Update otomatis kolom SONG_RATING saat ada perubahan rating lagu.
Jenis	AFTER INSERT OR UPDATE OR DELETE

<p>Deskripsi</p>	<p>Trigger ini dijalankan setiap kali ada perubahan di tabel RATE_SONGS, baik itu INSERT, UPDATE, maupun DELETE. Fungsi utamanya adalah menghitung ulang rata-rata rating untuk sebuah lagu (song_id) dan menyimpannya di kolom SONGS.SONG_RATING.</p> <p>Logika perhitungan AVG:</p> <ul style="list-style-type: none"> - Sistem akan mengambil semua rating yang ada untuk lagu tertentu (song_id). - Kemudian dihitung rata-ratanya menggunakan AVG(song_rating). - Hasilnya akan otomatis di-update ke song_rating. <p>Dengan mekanisme ini, kolom SONG_RATING selalu mencerminkan rata-rata rating terkini dari semua user untuk lagu tersebut.</p>
<p>Script SQL</p>	
<pre>-- Update song rating -- Fungsi ini digunakan untuk menghitung ulang rata-rata rating sebuah lagu -- setiap kali terjadi perubahan pada tabel rate_songs. -- Trigger akan memanggil fungsi ini saat INSERT, UPDATE, atau DELETE. CREATE OR REPLACE FUNCTION update_song_rating() RETURNS TRIGGER AS \$\$ DECLARE v_new_rating NUMERIC(5,2); -- Variabel untuk menyimpan rata-rata rating terbaru BEGIN -- Hitung ulang rata-rata rating untuk song -- NEW.song_id digunakan saat INSERT/UPDATE. -- OLD.song_id digunakan saat DELETE. -- COALESCE memastikan salah satu pasti terpakai. SELECT AVG(song_rating) INTO v_new_rating FROM rate_songs WHERE song_id = COALESCE(NEW.song_id, OLD.song_id);</pre>	

```
-- Update nilai rata-rata rating ke tabel songs.  
-- Jika semua rating dihapus, AVG akan NULL → lagu tidak punya  
rating.  
UPDATE songs  
SET song_rating = v_new_rating  
WHERE song_id = COALESCE(NEW.song_id, OLD.song_id);  
  
RETURN NULL;  
END;  
$$ LANGUAGE plpgsql;  
  
-- Trigger  
CREATE TRIGGER trg_update_song_rating  
AFTER INSERT OR UPDATE OR DELETE  
ON rate_songs  
FOR EACH ROW  
EXECUTE FUNCTION update_song_rating();
```

Screenshot Hasil

-- insert rating baru

```
FROM collections
WHERE collection_id = 1000;

-- Update song rating
CREATE OR REPLACE FUNCTION update_song_rating()
RETURNS TRIGGER AS $$
DECLARE
    v_new_rating NUMERIC(5,2);
BEGIN
    -- Hitung ulang rata-rata rating untuk song
    SELECT AVG(song_rating)
    INTO v_new_rating
    
```

CREATE FUNCTION

Query returned successfully in 617 msec.

Total rows: Query complete 00:00:00.617

16:15 22/11/2025 CRLF Ln 3627, Col 21


```
RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_update_song_rating
AFTER INSERT OR UPDATE OR DELETE
ON rate_songs
FOR EACH ROW
EXECUTE FUNCTION update_song_rating();
```

CREATE_TRIGGER

Query returned successfully in 95 msec.

Total rows: Query complete 00:00:00.095

16:18 22/11/2025 CRLF Ln 3629, Col 1

pgAdmin 4

Object Explorer

```

3631    ON rate_songs
3632    FOR EACH ROW
3633        EXECUTE FUNCTION update_song_rating();
3634
3635        SELECT * FROM songs;
3636
3637    -- cek
3638    -- Insert rating baru
3639    INSERT INTO rate_songs (user_id, song_id, song_rating)
3640        VALUES (2, 2, 4);
3641
3642
3643    SELECT song_id, song_rating FROM songs WHERE song_id = 2;

```

Total rows: Query complete 00:00:00.375 CRLF Ln 3639, Col 1

Query returned successfully in 375 msec.

pgAdmin 4

Object Explorer

```

3635    SELECT * FROM songs;
3636
3637    -- cek
3638    -- Insert rating baru
3639    INSERT INTO rate_songs (user_id, song_id, song_rating)
3640        VALUES (2, 2, 4);
3641
3642
3643    SELECT song_id, song_rating FROM songs WHERE song_id = 2;

```

Total rows: 1 Query complete 00:00:00.220 CRLF Ln 3644, Col 1

Showing rows: 1 to 1 Page No: 1 of 1

song_id	song_rating
[PK] integer	numeric (3)
1	2
4	

Successfully run. Total query runtime: 220 msec. 1 rows affected.

-- update rating song

```

3639    INSERT INTO rate_songs (user_id, song_id, song_rating)
3640        VALUES (2, 2, 4);
3641
3642    -- Update rating song
3643    UPDATE rate_songs
3644        SET song_rating = 5
3645        WHERE user_id = 2 AND song_id = 2;
3646
3647
3648    SELECT song_id, song_rating FROM songs WHERE song_id = 2;

```

UPDATE 1

Query returned successfully in 688 msec.

Total rows: Query complete 00:00:00.690 CRLF Ln 3642, Col 1

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows two tables: "tours" and "users".
- Query History:** Displays the following SQL code:

```
-- Insert rating baru
INSERT INTO rate_songs (user_id, song_id, song_rating)
VALUES (2, 2, 4);

-- Update rating song
UPDATE rate_songs
SET song_rating = 5
WHERE user_id = 2 AND song_id = 2;

-- Update rating song
UPDATE rate_songs
SET song_rating = 5
WHERE user_id = 3 AND song_id = 2;
```
- Data Output:** Shows the result of the last UPDATE query:

song_id	song_rating
2	5
- Messages:** A green message box indicates: "✓ Query returned successfully in 133 msec. X".
- System Bar:** Shows "Total rows: 0" and "Query complete 00:00:00.133".
- Bottom Bar:** Shows the date and time: "CRLF Ln 3648, Col 1" and "16:26 22/11/2025".

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows the same "tours" and "users" tables.
- Query History:** Displays the following SQL code:

```
INSERT INTO rate_songs (user_id, song_id, song_rating)
VALUES (2, 2, 4);

-- Update rating song
UPDATE rate_songs
SET song_rating = 5
WHERE user_id = 2 AND song_id = 2;

SELECT song_id, song_rating FROM songs WHERE song_id = 2;
```
- Data Output:** Shows the result of the SELECT query:

song_id	song_rating
2	5
- Messages:** A green message box indicates: "✓ Successfully run. Total query runtime: 128 msec. 1 rows affected. X".
- System Bar:** Shows "Total rows: 1" and "Query complete 00:00:00.128".
- Bottom Bar:** Shows the date and time: "CRLF Ln 3648, Col 1" and "16:25 22/11/2025".

-- insert baru

-- Insert baru

```

3643 UPDATE rate_songs
3644 SET song_rating = 5
3645 WHERE user_id = 2 AND song_id = 2;
3646
3647 -- Update rating song
3648 UPDATE rate_songs
3649 SET song_rating = 5
3650 WHERE user_id = 3 AND song_id = 2;
3651
3652 INSERT INTO rate_songs (user_id, song_id, song_rating)
3653 VALUES (3, 2, 3);
3654
3655

```

Query returned successfully in 462 msec.

Total rows: Query complete 00:00:00.462 CRLF Ln 3652, Col 1

-- Insert baru

```

3646
3647 -- Update rating song
3648 UPDATE rate_songs
3649 SET song_rating = 5
3650 WHERE user_id = 3 AND song_id = 2;
3651
3652 -- Insert baru
3653 INSERT INTO rate_songs (user_id, song_id, song_rating)
3654 VALUES (3, 2, 3);
3655
3656
3657 SELECT song_id, song_rating FROM songs WHERE song_id = 2;
3658

```

song_id	song_rating
2	4

Showing rows: 1 to 1 Page No: 1 of 1

Successfully run. Total query runtime: 604 msec. 1 rows affected. CRLF Ln 3657, Col 1

Total rows: 1 Query complete 00:00:00.604 CRLF Ln 3657, Col 1

-- Delete

```

3648 UPDATE rate_songs
3649 SET song_rating = 5
3650 WHERE user_id = 3 AND song_id = 2;
3651
3652 -- Insert baru
3653 INSERT INTO rate_songs (user_id, song_id, song_rating)
3654 VALUES (3, 2, 3);
3655
3656 -- Delete
3657 DELETE FROM rate_songs
3658 WHERE user_id = 2 AND song_id = 2;
3659
3660 SELECT song_id, song_rating FROM songs WHERE song_id = 2;

```

Total rows: Query complete 00:00:00.140 CRLF Ln 3656, Col 1


```

3653 INSERT INTO rate_songs (user_id, song_id, song_rating)
3654 VALUES (3, 2, 3);
3655
3656 -- Delete
3657 DELETE FROM rate_songs
3658 WHERE user_id = 2 AND song_id = 2;
3659
3660 SELECT song_id, song_rating FROM songs WHERE song_id = 2;
3661
3662
3663
3664
3665

```

Total rows: 1 Query complete 00:00:00.142 CRLF Ln 3659, Col 1

Tabel 117 PL/SQL – Trigger trg_validate_prerelease_date

Nama Trigger	trg_validate_prerelease_date
Aksi	<p>Melakukan validasi otomatis terhadap data koleksi sebelum proses INSERT atau UPDATE, meliputi:</p> <ol style="list-style-type: none"> Melarang koleksi berstatus prerelease (isPrerelease = TRUE) memiliki tanggal rilis yang tidak di masa depan. Melarang koleksi yang bukan prerelease (isPrerelease = FALSE) memiliki tanggal rilis di masa depan.

	<p>3. Mengubah otomatis nilai isPrerelease menjadi FALSE apabila tanggal rilis koleksi adalah hari ini (CURRENT_DATE).</p>
Jenis	BEFORE INSERT OR UPDATE
Deskripsi	<p>Trigger trg_validate_prerelease_date digunakan untuk memastikan bahwa setiap data koleksi (album/playlist) pada tabel collections mengikuti aturan validasi terkait status prerelease dan tanggal rilis.</p> <p>Setiap kali entri koleksi ditambahkan atau diperbarui, trigger ini akan memanggil function validate_prerelease_date() untuk melakukan pengecekan:</p> <ol style="list-style-type: none"> 1. Validasi Prerelease (isPrerelease = TRUE) <p>Jika sebuah koleksi ditandai sebagai prerelease, maka tanggal rilisnya harus berada di masa depan. Jika tanggal rilis tidak di masa depan (lebih kecil atau sama dengan CURRENT_DATE), maka proses akan dibatalkan dengan ERROR.</p> <ol style="list-style-type: none"> 2. Validasi Non-Prerelease (isPrerelease = FALSE) <p>Jika koleksi bukan prerelease, maka tanggal rilisnya tidak boleh berada di masa depan. Jika tanggal rilis masih di masa depan, trigger akan menghentikan proses dengan ERROR.</p> <ol style="list-style-type: none"> 3. Pengubahan Otomatis isPrerelease <p>Apabila tanggal rilis koleksi sama dengan hari ini, trigger akan otomatis mengubah nilai isPrerelease menjadi FALSE. Ini memastikan sistem mencerminkan bahwa koleksi tersebut sudah rilis pada hari itu.</p> <p>Dengan mekanisme ini, trigger menjamin integritas data prerelease dan rilis sesuai aturan bisnis yang ditetapkan, sekaligus menghindari ketidaksesuaian status dan tanggal rilis di tabel collections.</p>

Script SQL

```
CREATE OR REPLACE FUNCTION validate_prerelease_date()
RETURNS TRIGGER AS $$

BEGIN
    -- RULE 1: prerelease must be future
    IF NEW.isPrerelease = TRUE  -- Jika isPrerelease = TRUE
        AND NEW.collection_release_date <= CURRENT_DATE THEN    -- Tetapi
            tanggal rilis tidak di masa depan
            RAISE EXCEPTION
                'If isPrerelease = TRUE, then release_date (%s) must be in the
future.',  -- Pesan error
                NEW.collection_release_date;
    END IF;

    -- RULE 2: NOT prerelease must be today or past
    IF NEW.isPrerelease = FALSE  -- Jika isPrerelease = FALSE
        AND NEW.collection_release_date > CURRENT_DATE THEN -- Tetapi tanggal
            rilis berada di masa depan
            RAISE EXCEPTION
                'If isPrerelease = FALSE, release_date (%s) cannot be in the
future.',  -- Pesan error
                NEW.collection_release_date;    -- Menampilkan tanggal yang salah
    END IF;

    -- RULE 3: If release date is today → force prerelease = FALSE
    IF NEW.collection_release_date = CURRENT_DATE THEN    -- Jika tanggal
            rilis sama dengan hari ini
        NEW.isPrerelease := FALSE;  -- isPrerelease menjadi FALSE
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_validate_prerelease_date
BEFORE INSERT OR UPDATE ON collections
FOR EACH ROW
EXECUTE FUNCTION validate_prerelease_date();
```

Screenshot Hasil

pgAdmin 4

Object Explorer

```

song_rating
  > Constraints
  > Indexes
  > RLS Policies
  > Rules
  > Triggers
  > songs_genres
  > tours
  > users
    Trigger Functions (10)
      fix_playlist_collaborators()
      play_song_logic()
      prerelease_check_logic()
      update_artist_follow_count()
      update_collection_genre()
      update_collection_rating()
      update_collection_top3_gen
      update_review_timestamp()
      update_song_rating()
      validate_prerelease_date()
    Types
    Views
    Subscriptions
  Produk

```

Final_Destination/postgres@Postgres16

Query History

```

4366 IF NEW.collection_release_date = CURRENT_DATE THEN
4367   NEW.isPrerelease := FALSE;
4368 END IF;
4369
4370 RETURN NEW;
4371
4372 END;
4373 $ LANGUAGE plpgsql;
4374
4375 CREATE TRIGGER trg_validate_prerelease_date
4376 BEFORE INSERT OR UPDATE ON collections
4377 FOR EACH ROW
4378 EXECUTE FUNCTION validate_prerelease_date();

```

Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 94 msec.

Total rows: Query complete 00:00:00.094 CRLF Ln 4378; Col 21 10:02 27/11/2025

pgAdmin 4

Object Explorer

```

FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences (11)
  1.3 seq_add_songs_playlist_id
  1.3 seq_artists_id
  1.3 seq_collections_id
  1.3 seq_genres_id
  1.3 seq_listens_id
  1.3 seq_playlists_id
  1.3 seq_reviews_id
  1.3 seq_socials_id
  1.3 seq_songs_id
  1.3 seq_tours_id
  1.3 seq_users_id
Tables (27)
  add_songs_playlists
  artist_promotion
  artists
  artiste_tours

```

Final_Destination/postgres@Postgres16

Query History

```

4341 -- cek song_id nya yg gak ada di data
4342 SELECT * FROM get_song_audio_features(500);
4343
4344
4345
4346 CREATE OR REPLACE FUNCTION validate_prerelease_date()
4347 RETURNS TRIGGER AS $$
4348 BEGIN
4349   -- RULE 1: prerelease must be future
4350   IF NEW.isPrerelease = TRUE
4351     AND NEW.collection_release_date <= CURRENT_DATE THEN
4352     RAISE EXCEPTION
4353       !If isPrerelease = TRUE, then release date (%s) must be

```

Data Output Messages Notifications

CREATE TRIGGER

Query returned successfully in 582 msec.

Total rows: Query complete 00:00:00.605 CRLF Ln 4346; Col 1 10:01 27/11/2025

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- > blocklist_artists
- > collection_library
- > collection_top_3_genres
- > collections
- > collections_songs
- > create_songs
- > follow_artists
- > follow_users
- > genres
- > like_reviews
- > like_songs
- > lists
- > p_library
- > playlists
- > rate_songs
- > releases
- > reviews
- > socials
- > songs
- > Columns (13)
 - song_id
 - song_file
 - song_title
 - listen_count
 - song_credits

Final_Destination/postgres@Postgres16

Query History

```

4376 FOR EACH ROW
4377   EXECUTE FUNCTION validate_prerelease_date();
4378
4379   SELECT * FROM collections;
4380
  
```

Data Output Messages Notifications

collection_type	collection_cover	collection_release_date	collection_rating	isprerelease	collection_genre
35 Compilation	/uploads/covers/coll_36.jpg	2022-05-26	[null]	[null]	[null]
36 Compilation	/uploads/covers/coll_37.jpg	2021-04-03	[null]	[null]	[null]
37 EP	/uploads/covers/coll_38.jpg	2021-04-08	[null]	[null]	[null]
38 Compilation	/uploads/covers/coll_39.jpg	2023-03-14	[null]	[null]	[null]
39 Compilation	/uploads/covers/coll_40.jpg	2025-06-06	[null]	[null]	[null]
40 Single	/uploads/covers/coll_1.jpg	2022-12-15	5	[null]	[null]
41 Album	[null]	2025-11-22	[null]	false	[null]
42 Album	[null]	2025-11-22	[null]	false	[null]
43 Album	cover.jpg	2025-12-02	[null]	true	[null]

Total rows: 43 Query complete 00:00:00.117 CRLF Ln 4379, Col 1

10:12 27/11/2025

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- > blocklist_artists
- > collection_library
- > collection_top_3_genres
- > collections
- > collections_songs
- > create_songs
- > follow_artists
- > follow_users
- > genres
- > like_reviews
- > like_songs
- > lists
- > p_library
- > playlists
- > rate_songs
- > releases
- > reviews
- > socials
- > songs
- > Columns (13)
 - song_id
 - song_file
 - song_title
 - listen_count
 - song_credits

Final_Destination/postgres@Postgres16

Query History

```

4399 -- Test prerelease INVALID
4400   INSERT INTO collections VALUES (
4401     52,
4402     'Test Invalid',
4403     'Album',
4404     'cover.jpg',
4405     CURRENT_DATE,
4406     null,
4407     TRUE
4408   );
4409
  
```

Data Output Messages Notifications

INSERT 0 1

Query returned successfully in 675 msec.

✓ Query returned successfully in 675 msec. X

Total rows: 0 Query complete 00:00:00.675 CRLF Ln 4401, Col 1

10:21 27/11/2025

The screenshot shows the pgAdmin 4 interface. In the center, there is a query editor window titled 'Query' containing the following SQL code:

```

4411   INSERT INTO collections (
4412     collection_id,
4413     collection_title,
4414     collection_type,
4415     collection_cover,
4416     collection_release_date,
4417     isPrerelease
4418   )
4419   VALUES (
4420     80,
4421     'Rule 2 Test',

```

Below the code, an error message is displayed:

ERROR: If isPrerelease = FALSE, release_date (2025-11-30s) cannot be in the future.
CONTEXT: PL/pgSQL function validate_prerelease_date() line 14 at RAISE
SQL state: P0001

At the bottom of the query editor, it says 'Total rows: 5 Query complete 00:00:00.213'. To the right, there is status information: 'CRLF Ln 4428, Col 1' and a timestamp '17:54 27/11/2025'.

Tabel 118 PL/SQL - Trigger trg_update_review_timestamp

Nama Trigger	trg_update_review_timestamp
Aksi	Memperbarui kolom TIMESTAMP pada tabel reviews ketika nilai review atau rating diubah.
Jenis	BEFORE UPDATE

<p>Deskripsi</p>	<p>Trigger ini digunakan untuk meng-update otomatis timestamp setiap kali user mengubah:</p> <ol style="list-style-type: none"> 1. Isi review (review) 2. Nilai rating (rating) <p>Jika salah satu dari dua kolom tersebut berubah, maka kolom "TIMESTAMP" akan diisi ulang dengan waktu saat perubahan terjadi (CURRENT_TIMESTAMP).</p> <p>Trigger memastikan data riwayat perubahan review selalu tercatat akurat.</p>
<p>Script SQL</p>	
<pre>-- FUNCTION untuk update timestamp saat review/rating berubah CREATE OR REPLACE FUNCTION update_review_timestamp() RETURNS TRIGGER AS \$\$ BEGIN -- Jika kolom review atau rating berubah IF NEW.review IS DISTINCT FROM OLD.review -- IS DISTINCT FROM digunakan agar aman untuk nilai NULL. OR NEW.rating IS DISTINCT FROM OLD.rating THEN NEW."TIMESTAMP" = CURRENT_TIMESTAMP; -- Jika ada perubahan pada review atau rating, update kolom TIMESTAMP dengan waktu saat ini. END IF; RETURN NEW; END; \$\$ LANGUAGE plpgsql; -- TRIGGER CREATE TRIGGER trg_update_review_timestamp BEFORE UPDATE</pre>	

```

ON reviews
FOR EACH ROW
EXECUTE FUNCTION update_review_timestamp();

```

Screenshot Hasil

```

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
releases
    Columns(2)
        collection_id
        artist_id
    Constraints
    Indexes
    RLS Policies
    Rules
    Triggers
reviews
    Columns(6)
        review
        rating
        TIMESTAMP
        review_id
        user_id
        collection_id
    Constraints
    Indexes
    RLS Policies
    Rules
    Triggers
socials
songs
    Columns(13)

Query Query History
3595     FUNCTION untuk update timestamp saat review/rating berubah
3596     CREATE OR REPLACE FUNCTION update_review_timestamp()
3597     RETURNS TRIGGER AS $$
3598     BEGIN
3599         -- Jika kolom review atau rating berubah
3600         IF NEW.review IS DISTINCT FROM OLD.review
3601             OR NEW.rating IS DISTINCT FROM OLD.rating THEN
3602             NEW."TIMESTAMP" = CURRENT_TIMESTAMP;
3603         END IF;
3604
3605         RETURN NEW;
3606     END;
3607     $$ LANGUAGE plpgsql;
CREATE FUNCTION
Query returned successfully in 628 msec.

Total rows: 0 Query complete 00:00:00.628
09:31 22/11/2025

```

```

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
releases
    Columns(2)
        collection_id
        artist_id
    Constraints
    Indexes
    RLS Policies
    Rules
    Triggers
reviews
    Columns(6)
        review
        rating
        TIMESTAMP
        review_id
        user_id
        collection_id
    Constraints
    Indexes
    RLS Policies
    Rules
    Triggers
socials
songs
    Columns(13)

Query Query History
3516
3517     RETURN NEW;
3518 END;
3519 $$ LANGUAGE plpgsql;
3520
3521 -- TRIGGER
3522 CREATE TRIGGER trg_update_review_timestamp
3523 BEFORE UPDATE
3524 ON reviews
3525 FOR EACH ROW
3526 EXECUTE FUNCTION update_review_timestamp();
CREATE TRIGGER
Query returned successfully in 514 msec.

Total rows: 0 Query complete 00:00:00.514
09:34 22/11/2025

```

pgAdmin 4

Object Explorer

```

releases
  - Columns(2)
    - collection_id
    - artist_id
  - Constraints
  - Indexes
  - RLS Policies
  - Rules
  - Triggers
reviews
  - Columns(6)
    - review
    - rating
    - TIMESTAMP
    - review_id
    - user_id
    - collection_id
  - Constraints
  - Indexes
  - RLS Policies
  - Rules
  - Triggers
  - socials
  - songs
    - Columns(13)

```

Final_Destination/postgres@Postgres

Query History

```

3524 ON reviews
3525 FOR EACH ROW
3526 EXECUTE FUNCTION update_review_timestamp();
3527
3528 -- tabel review
3529 SELECT * FROM reviews;
3530
3531
3532
3533

```

Data Output

review	rating	TIMESTAMP	review_id	user_id	collection_id
baguss	3	2025-11-21 16:46:24.93363	1	10	1
kerenn	5	2025-11-21 17:35:28.791737	2	1	1

Total rows: 2 Query complete 00:00:00.158 CRLF Ln 3529, Col 1

09:36 22/11/2025

pgAdmin 4

Object Explorer

```

releases
  - Columns(2)
    - collection_id
    - artist_id
  - Constraints
  - Indexes
  - RLS Policies
  - Rules
  - Triggers
reviews
  - Columns(6)
    - review
    - rating
    - TIMESTAMP
    - review_id
    - user_id
    - collection_id
  - Constraints
  - Indexes
  - RLS Policies
  - Rules
  - Triggers
  - socials
  - songs
    - Columns(13)

```

Final_Destination/postgres@Postgres

Query History

```

3526
3527
3528 FOR EACH ROW
3529 EXECUTE FUNCTION update_review_timestamp();
3530
3531 -- tabel review
3532 UPDATE reviews
3533 SET review = 'Review baru test trigger'
3534 WHERE review_id = 1;
3535

```

Data Output

UPDATE 1

Query returned successfully in 138 msec.

Total rows: 1 Query complete 00:00:00.138 CRLF Ln 3531, Col 1

09:38 22/11/2025

pgAdmin 4

Object Explorer

```

releases
  - Columns(2)
    - collection_id
    - artist_id
  - Constraints
  - Indexes
  - RLS Policies
  - Rules
  - Triggers
reviews
  - Columns(6)
    - review
    - rating
    - TIMESTAMP
    - review_id
    - user_id
    - collection_id
  - Constraints
  - Indexes
  - RLS Policies
  - Rules
  - Triggers
  - socials
  - songs
    - Columns(13)

```

Final_Destination/postgres@Postgres

Query History

```

3526
3527
3528 FOR EACH ROW
3529 EXECUTE FUNCTION update_review_timestamp();
3530
3531 -- tabel review
3532 UPDATE reviews
3533 SET review = 'Review baru test trigger'
3534 WHERE review_id = 1;
3535

```

Data Output

review	rating	TIMESTAMP	review_id	user_id	collection_id
kerenn	5	2025-11-21 17:35:28.791737	2	1	1
Review baru test trigger	3	2025-11-22 09:38:14.888103	1	10	1

Total rows: 2 Query complete 00:00:00.414 CRLF Ln 3529, Col 1

09:39 22/11/2025

-- cek update rating

The screenshot shows two instances of pgAdmin 4. Both instances have the same database connection: Final_Destination/postgres@Postgres16.

Session 1 (Top):

```
-- update review timestamp
UPDATE reviews
SET review = 'Review baru test trigger'
WHERE review_id = 1;

-- update rating timestamp
UPDATE reviews
SET rating = rating + 1
WHERE review_id = 1;
```

Session 2 (Bottom):

```
EXECUTE FUNCTION update_review_timestamp();

-- tabel review
SELECT * FROM reviews;
```

Data Output:

review	rating	TIMESTAMP	review_id	user_id	collection_id
kerenn	5	2025-11-21 17:35:28.791737	1	2	1
Review baru test trigger	4	2025-11-22 09:42:38.400275	2	10	1

Tabel 119 PL/SQL - Trigger trg_fix_playlist_collaborators

Nama Trigger	trg_fix_playlist_collaborators
Aksi	UPDATE (kolom: iscollaborative)
Jenis	AFTER UPDATE

<p>Deskripsi</p>	<p>Trigger ini digunakan ketika sebuah playlist yang sebelumnya collaborative (true) diubah menjadi non-collaborative (false). Saat perubahan ini terjadi, seluruh lagu yang sebelumnya ditambahkan oleh user selain pemilik playlist akan diperbarui user_id-nya menjadi user_id pemilik playlist.</p> <p>Tujuan trigger ini adalah memastikan bahwa ketika playlist tidak lagi bersifat kolaboratif, seluruh lagu di dalamnya dianggap berasal dari pemilik playlist.</p> <p>Logika Perhitungan:</p> <p>Jika OLD.iscollaborative = TRUE dan NEW.iscollaborative = FALSE → maka untuk setiap lagu dalam playlist tersebut:</p> <p>Jika user_id != owner (NEW.user_id) → ganti menjadi NEW.user_id.</p>
<p>Script SQL</p>	<pre>-- Function: Update user_id lagu ketika playlist tidak lagi collaborative CREATE OR REPLACE FUNCTION fix_playlist_collaborators() RETURNS TRIGGER AS \$\$ BEGIN -- Jika sebelumnya collaborative dan sekarang tidak -- collaborative IF OLD.iscollaborative = TRUE AND NEW.iscollaborative = FALSE THEN -- Update semua lagu yang ditambahkan oleh user lain UPDATE add_songs_playlists SET user_id = NEW.user_id -- ganti pemilik lagu WHERE playlist_id = NEW.playlist_id -- hanya untuk playlist ini AND user_id <> NEW.user_id; -- hanya yang bukan owner END IF;</pre>

```

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Trigger: berjalan ketika playlist di-update
CREATE TRIGGER trg_fix_playlist_collaborators
AFTER UPDATE OF iscollaborative
ON playlists -- trigger untuk tabel playlists
FOR EACH ROW
EXECUTE FUNCTION fix_playlist_collaborators();

```

Screenshot Hasil

The screenshot displays two pgAdmin 4 windows side-by-side, showing the results of running SQL code.

Top Window:

- Object Explorer shows the schema structure, including tables like 'artists' and 'artist_promotion'.
- Query Editor window contains the following SQL code:

```

-- Function: Update user_id lagu ketika playlist tidak lagi collaboratif
CREATE OR REPLACE FUNCTION fix_playlist_collaborators()
RETURNS TRIGGER AS $$

BEGIN
    -- Jika sebelumnya collaborative dan sekarang tidak collaborative
    IF OLD.iscollaborative = TRUE AND NEW.iscollaborative = FALSE THEN
        -- Update semua lagu yang ditambahkan oleh user lain
        UPDATE add_songs_playlists
        SET user_id = NEW.user_id
        WHERE playlist_id = NEW.playlist_id
        AND user_id <> NEW.user_id;
    END IF;

```

- Messages tab shows: "Query returned successfully in 138 msec."

Bottom Window:

- Object Explorer shows the schema structure.
- Query Editor window contains the following SQL code:

```

END;
$$ LANGUAGE plpgsql;

-- Trigger: berjalan ketika playlist di-update
CREATE TRIGGER trg_fix_playlist_collaborators
AFTER UPDATE OF iscollaborative
ON playlists
FOR EACH ROW
EXECUTE FUNCTION fix_playlist_collaborators();

SELECT * FROM USERS; --dongorankunthara
SELECT * FROM add_songs_playlists;

```

- Messages tab shows: "Query returned successfully in 200 msec."

pgAdmin 4

Object Explorer Final_Destination/postgres@Postgres16 02_dml_seed.sql Final Destination/... 01_ddl_schema.sql TRIGGER1.sql coba_coba_virli.sql*

Query Data Output Messages Notifications

```

3495 -- Insert playlist dengan collaborative = TRUE
3496 INSERT INTO playlists (playlist_id, user_id, playlist_title, ispublic)
3497 VALUES (1, 10, 'My Playlist', TRUE, TRUE, TRUE, CURRENT_DATE);
3498
3499 -- Data sebelum trigger bekerja
3500
3501 SELECT add_song_pl_id, user_id, playlist_id, song_id
3502 FROM add_songs_playlists
3503 WHERE playlist_id = 2;
3504

```

Total rows: 5 Query complete 00:00:00.231 CRLF Ln 3504, Col 1

Successfully run. Total query runtime: 231 msec. 5 rows affected.

pgAdmin 4

Object Explorer Final_Destination/postgres@Postgres16 02_dml_seed.sql Final Destination/... 01_ddl_schema.sql TRIGGER1.sql coba_coba_virli.sql*

Query Data Output Messages Notifications

```

3485 -- EXECUTE FUNCTION fix_playlist_collaborators();
3486
3487
3488 EXECUTE FUNCTION fix_playlist_collaborators();
3489
3490
3491
3492
3493
3494

```

Total rows: 30 Query complete 00:00:00.201 CRLF Ln 3494, Col 1

Successfully run. Total query runtime: 201 msec. 30 rows affected.

pgAdmin 4

Object Explorer Final_Destination/postgres@Postgres16 02_dml_seed.sql Final Destination/... 01_ddl_schema.sql TRIGGER1.sql coba_coba_virli.sql*

Query Data Output Messages Notifications

```

3500 -- Data sebelum trigger bekerja
3501
3502
3503
3504
3505
3506
3507
3508
3509

```

UPDATE 1

Query returned successfully in 141 msec.

Total rows: 1 Query complete 00:00:00.141 CRLF Ln 3509, Col 1

Query returned successfully in 141 msec.

pgAdmin 4

Object Explorer

Query History

```

3488  SELECT * FROM users; --oongorankuntara
3489  SELECT * FROM add_songs_playlists;
3490  SELECT * FROM PLAYLISTS;
3491
3492  -- cek
3493  -- Insert playlist owner (user_id = 10 misalnya)
3494  INSERT INTO users (user_id, username) VALUES (10, 'owner');
3495
3496  -- Insert playlist dengan collaborative = TRUE
3497  INSERT INTO playlists (playlist_id, user_id, playlist_title, ispubl...
3498  VALUES (1, 10, 'My Playlist', TRUE, TRUE, CURRENT_DATE);
3499

```

Data Output

playlist_id	user_id	playlist_cover	playlist_title	ispublic	iscollaborative	playlist_desc
26	27	21 [null]	Playlist sit Merah collab	true	true	Playlist asik br.
27	28	17 [null]	Playlist libero Hijau collab	true	false	Playlist asik br.
28	29	27 [null]	Playlist beatas Hitam collab	true	false	Playlist asik br.
29	30	11 [null]	Playlist officia Merah jambu collab	true	false	Playlist asik br.
30	2	36 [null]	Playlist reprehenderit Merah jambu collab	true	false	Playlist asik br.

Total rows: 30 Query complete 00:00:00.211 CRLF Ln 3490, Col 1

pgAdmin 4

Object Explorer

Query History

```

3499
3500  -- Data sebelum/sesudah trigger bekerja
3501  SELECT add_song_pl_id, user_id, playlist_id, song_id
3502  FROM add_songs_playlists
3503  WHERE playlist_id = 2
3504
3505  -- Ubah playlist collaborative --> tidak collab
3506  UPDATE playlists
3507  SET iscollaborative = false

```

Data Output

add_song_pl_id	user_id	playlist_id	song_id
1	9	36	2 46
2	6	36	2 14
3	7	36	2 131
4	8	36	2 21
5	10	36	2 40

Total rows: 5 Query complete 00:00:00.376 CRLF Ln 3500, Col 1

Tabel 120 PL/SQL - Trigger trg_update_artist_follower_count

Nama Trigger	trg_update_artist_follow_count
Aksi	INSERT OR DELETE
Jenis	AFTER

<p>Deskripsi</p>	<p>Trigger ini digunakan untuk menjaga konsistensi jumlah followers artis pada tabel artists. Setiap kali terjadi aksi FOLLOW (insert ke tabel follow_artists), nilai follower_count artis akan bertambah 1. Sebaliknya, saat terjadi UNFOLLOW (delete dari tabel follow_artists), nilai follower_count akan berkurang 1. Dan trigger ini memastikan bahwa jumlah followers selalu sesuai dengan jumlah baris aktual dalam tabel follow_artists.</p>
<p>Script SQL</p>	
<pre>-- Trigger follow-unfollow artis CREATE OR REPLACE FUNCTION update_artist_follow_count() RETURNS TRIGGER AS \$\$ BEGIN -- FOLLOW → INSERT IF TG_OP = 'INSERT' THEN -- TG_OP berisi string yang menunjukkan operasi apa yang memicu trigger nya UPDATE artists SET follower_count = COALESCE(follower_count, 0) + 1 WHERE artist_id = NEW.artist_id; -- artist_id dari baris baru END IF; -- UNFOLLOW → DELETE IF TG_OP = 'DELETE' THEN -- TG_OP berisi string yang menunjukkan operasi apa yang memicu trigger nya UPDATE artists SET follower_count = COALESCE(follower_count, 0) - 1 WHERE artist_id = OLD.artist_id; -- artist_id dari baris lama END IF; </pre>	

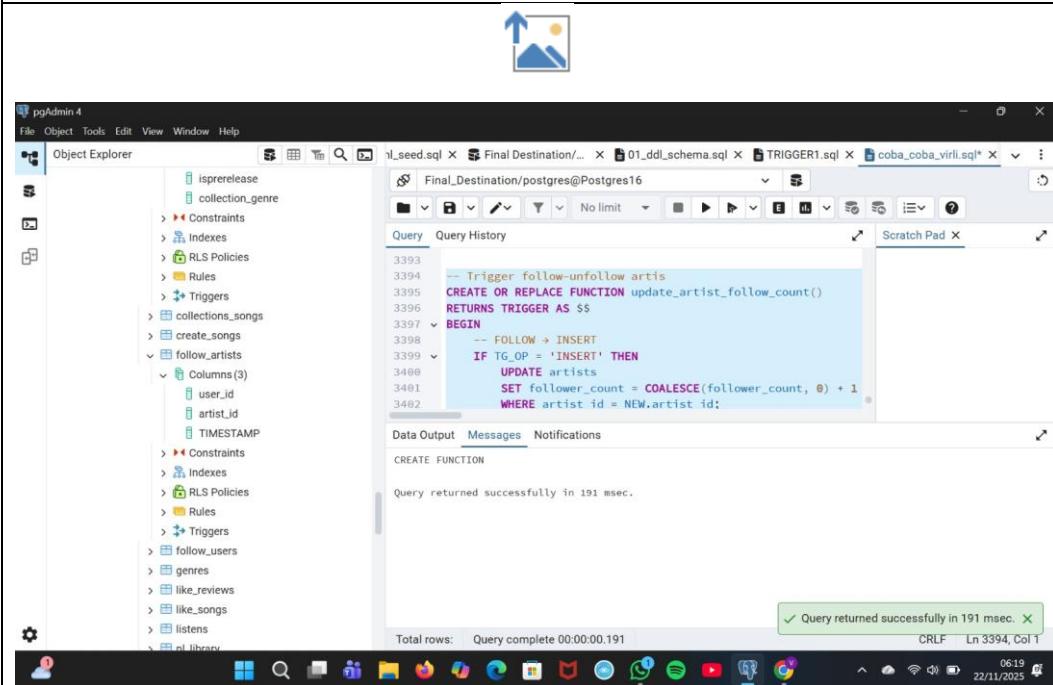
```

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

-- trigger
CREATE TRIGGER trg_update_artist_follow_count
AFTER INSERT OR DELETE
ON follow_artists
FOR EACH ROW
EXECUTE FUNCTION update_artist_follow_count();

```

Screenshot Hasil

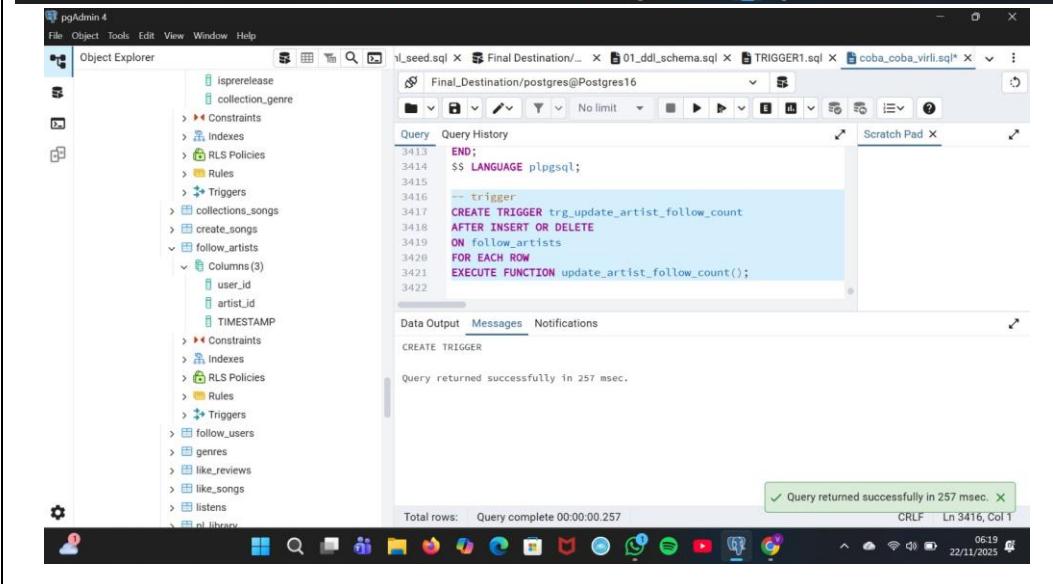


The screenshot shows the pgAdmin 4 interface with two windows. The left window is the Object Explorer displaying database schema, and the right window is a query editor. The query editor contains the SQL code for creating a trigger named 'trg_update_artist_follow_count'. The code includes logic to handle both 'INSERT' and 'DELETE' operations on the 'follow_artists' table, calling a function 'update_artist_follow_count()' to update the follower count.

```

-- Trigger follow-unfollow artis
CREATE OR REPLACE FUNCTION update_artist_follow_count()
RETURNS TRIGGER AS $$
BEGIN
    -- FOLLOW => INSERT
    IF TG_OP = 'INSERT' THEN
        UPDATE artists
        SET follower_count = COALESCE(follower_count, 0) + 1
        WHERE artist_id = NEW.artist_id;
    END IF;
    -- UNFOLLOW => DELETE
    IF TG_OP = 'DELETE' THEN
        UPDATE artists
        SET follower_count = COALESCE(follower_count, 0) - 1
        WHERE artist_id = OLD.artist_id;
    END IF;
END;
$$ LANGUAGE plpgsql;

```

This screenshot shows the same pgAdmin 4 interface, but the trigger name has been changed to 'TRIGGER1'. The rest of the code and schema are identical to the first screenshot.

```

-- trigger
CREATE TRIGGER TRIGGER1
AFTER INSERT OR DELETE
ON follow_artists
FOR EACH ROW
EXECUTE FUNCTION update_artist_follow_count();

```

-- Follow (Insert)

The screenshot shows two separate sessions in pgAdmin 4. Both sessions are connected to the database 'Final_Destination/postgres@Postgres16'.

Session 1 (Top):

```
4477 WHERE user_id = 1 AND artist_id = 1;
4478
4479 -- cek trigger
4480 INSERT INTO follow_artists (user_id, artist_id)
4481 VALUES (9, 4);
4482
4483 INSERT INTO follow_artists (user_id, artist_id)
4484 VALUES (7, 2);
4485
4486 -- unfollow
4487 DELETE FROM follow_artists
```

Session 2 (Bottom):

```
3438 VALUES (9, 4);
3439
3440 -- unfollow
3441 DELETE FROM follow_artists
3442 WHERE user_id = 5 AND artist_id = 2;
3443 DELETE FROM follow_artists
3444 WHERE user_id = 2 AND artist_id = 1;
3445
3446 SELECT follower_count FROM artists WHERE artist_id = 4;
3447
3448 SELECT * FROM artists;
3449
3450 SELECT follower_count FROM artists WHERE artist_id = 1;
```

In both sessions, the queries are run successfully, and the results are displayed in the Data Output tab. The first session shows an empty result set for the DELETE query. The second session shows a single row with a follower count of 1 for the specified artist.

-- Unffollow (Delete)

The image consists of three vertically stacked screenshots of the pgAdmin 4 interface. Each screenshot shows the Object Explorer on the left and a query editor window on the right.

- Screenshot 1:** Shows a series of INSERT INTO statements for the 'follow_artists' table. The first two inserts add rows (9, 4) and (10, 4). The third insert adds (3, 1). A comment '-- unfollow' is present before the final insert statement.
- Screenshot 2:** Shows a series of DELETE FROM statements for the 'follow_artists' table. The first delete is for user_id = 5 AND artist_id = 2; the second is for user_id = 2 AND artist_id = 1;. A comment '-- unfollow' is present before the first delete. Below the queries, a data output grid shows a single row with follower_count = 2.
- Screenshot 3:** Shows a SELECT query for follower_count from the 'artists' table where artist_id = 4. It also includes a SELECT * from artists and a SELECT follower_count from artists where artist_id = 1; for reference. The data output grid shows 20 rows of artist data, including columns like artist_id, artist_name, bio, monthly_listener_count, artist_pfp, artist_email, banner, and follower_count.

pgAdmin 4

Object Explorer

```

aristi_promotion
  Columns(3)
    artist_id
    collection_id
    komentar_promosi
  Constraints
  Indexes
  RLS Policies
  Rules
  Triggers
artists
  Columns(8)
    artist_id
    artist_name
    bio
    monthly_listener_count
    artist_pf
    artist_email
    banner
    follower_count
  Constraints
  Indexes
  RLS Policies
  Rules
  Triggers

```

Query

```

-- cek trigger
INSERT INTO follow_artists (user_id, artist_id)
VALUES (9, 4);

INSERT INTO follow_artists (user_id, artist_id)
VALUES (10, 4);

-- unfollow
DELETE FROM follow_artists
WHERE user_id = 9 AND artist_id = 4;
DELETE FROM follow_artists

```

Data Output

Messages Notifications

DELETE 1

Query returned successfully in 355 msec.

Total rows: Query complete 00:00:00.355 CRLF Ln 3437, Col 1

06:44 22/11/2025

pgAdmin 4

Object Explorer

```

aristi_promotion
  Columns(3)
    artist_id
    collection_id
    komentar_promosi
  Constraints
  Indexes
  RLS Policies
  Rules
  Triggers
artists
  Columns(8)
    artist_id
    artist_name
    bio
    monthly_listener_count
    artist_pf
    artist_email
    banner
    follower_count
  Constraints
  Indexes
  RLS Policies
  Rules
  Triggers

```

Query

```

WHERE user_id = 9 AND artist_id = 4;
DELETE FROM follow_artists
WHERE user_id = 2 AND artist_id = 1;

SELECT follower_count FROM artists WHERE artist_id = 4;
SELECT * FROM artists;

SELECT follower_count FROM artists WHERE artist_id = 1;
SELECT * FROM update_artist_follow_count();

```

Data Output

Messages Notifications

Showing rows: 1 to 1 Page No: 1 of 1

follower_count	bigint
1	1

Successfully run. Total query runtime: 138 msec. 1 rows affected.

Total rows: 1 Query complete 00:00:00.138 CRLF Ln 3442, Col 1

06:44 22/11/2025

pgAdmin 4

Object Explorer

```

aristi_promotion
  Columns(3)
    artist_id
    collection_id
    komentar_promosi
  Constraints
  Indexes
  RLS Policies
  Rules
  Triggers
artists
  Columns(8)
    artist_id
    artist_name
    bio
    monthly_listener_count
    artist_pf
    artist_email
    banner
    follower_count
  Constraints
  Indexes
  RLS Policies
  Rules
  Triggers

```

Query

```

-- unfollow
DELETE FROM follow_artists
WHERE user_id = 9 AND artist_id = 4;
DELETE FROM follow_artists
WHERE user_id = 2 AND artist_id = 1;

SELECT follower_count FROM artists WHERE artist_id = 4;
SELECT * FROM artists;

SELECT follower_count FROM artists WHERE artist_id = 1;

```

Data Output

Messages Notifications

Showing rows: 1 to 20 Page No: 1 of 1

artist_pf	character varying (2048)	artist_email	character varying (320)	banner	character varying (2048)	follower_count
1	Garang Purwanti_4.jpg	Garang.Purwanti_4@example.com	[null]	1	1	
19	585	/uploads/artists/artist_4.jpg	Garang.Purwanti_4@example.com	[null]	1	
20	371	/uploads/artists/artist_2.jpg	Drs. Endah Hakim, M.TI_2@example.com	[null]	1	

Total rows: 20 Query complete 00:00:00.149 CRLF Ln 3443, Col 1

06:45 22/11/2025

Tabel 121 PL/SQL - Trigger trg_update_collection_rating

Nama Trigger	trg_update_collection_rating
Aksi	INSERT, UPDATE, DELETE
Jenis	AFTER
Deskripsi	<p>Trigger (trg_update_collection_rating) digunakan untuk memperbarui nilai collection_rating pada tabel collections setiap kali terjadi perubahan data pada tabel reviews. Trigger ini menghitung ulang rata-rata rating dari seluruh review berdasarkan collection_id terkait, lalu menyimpannya kembali ke kolom collection_rating. Proses perhitungan dilakukan dengan cara mengambil rerata (AVG) dari semua nilai rating pada tabel reviews untuk koleksi tersebut.</p> <p>Contoh perhitungan nya terjadi selama pengujian:</p> <ul style="list-style-type: none"> - Review 1 memberikan rating 3 - Review 2 nya memberikan rating 5 <p>Maka perhitungan rata – rata:</p> $(3 + 5) / 2 = 4$ <p>Nilai 4 ini kemudian disimpan ke kolom collection_rating pada tabel collections. Dengan adanya trigger ini, nilai rating koleksi akan selalu akurat tanpa perlu diupdate secara manual.</p>
Script SQL	<pre>-- Trigger review collection, akan mengubah collection_rating</pre>

```

CREATE OR REPLACE FUNCTION update_collection_rating()
RETURNS TRIGGER AS $$

DECLARE
    v_new_rating NUMERIC(5,2); -- Variable untuk menyimpan rata-
rata rating terbaru
BEGIN
    -- Hitung ulang rating dari seluruh review untuk collection
    SELECT AVG(rating) -- Mengambil nilai rata-rata (AVG) dari
semua rating pada review
        INTO v_new_rating -- Menyimpan hasil rata-rata tersebut ke
variabel v_new_rating
    FROM reviews -- Mengambil data dari tabel reviews
    -- INSERT → pakai NEW.collection_id
    -- UPDATE → pakai NEW.collection_id
    -- DELETE → pakai OLD.collection_id
    WHERE collection_id = COALESCE(NEW.collection_id,
OLD.collection_id);

    -- Update ke tabel collections
    UPDATE collections
    SET collection_rating = v_new_rating -- Menuliskan nilai rata-
rata terbaru ke kolom collection_rating
    WHERE collection_id = COALESCE(NEW.collection_id,
OLD.collection_id); -- Menentukan collection mana yang harus di-
update

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

-- trigger perubahan rating insert review, update review, delete
review
CREATE TRIGGER trg_update_collection_rating
AFTER INSERT OR UPDATE OR DELETE
ON reviews
FOR EACH ROW
EXECUTE FUNCTION update_collection_rating();

```

Screenshot Hasil

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer pane displays a database schema with several tables and their columns. In the center, the Query Editor pane contains a SQL script for creating a function named 'update_collection_rating'. The script includes a declaration of a numeric variable 'v_new_rating', a begin block, a select statement to calculate the average rating from the 'reviews' table where 'collection_id' matches the new collection ID, and an update statement to set this average rating back into the 'collections' table for the specified collection. The status bar at the bottom right indicates the query was completed successfully in 132 msec.

```
CREATE OR REPLACE FUNCTION update_collection_rating()
RETURNS TRIGGER AS $$
DECLARE
    v_new_rating NUMERIC(5,2);
BEGIN
    -- Hitung ulang rating dari seluruh review untuk coll
    SELECT AVG(rating)
    INTO v_new_rating
    FROM reviews
    WHERE collection_id = COALESCE(NEW.collection_id, 0);
    -- Update ke tabel collections
    UPDATE collections
    SET collection_rating = v_new_rating
    WHERE collection_id = COALESCE(NEW.collection_id, 0);
END;
$$ LANGUAGE plpgsql;
```

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

```

> block_users
> blocklist_artists
> collection_library
> collection_top_3_genres
> Columns
> Constraints
> Indexes
> RLS Policies
> Rules
> Triggers
> collections
> Columns (8)
    collection_id
    collection_title
    collection_type
    collection_cover
    collection_release_date
    collection_rating
    isprelease
    collection_genre
> Constraints
> Indexes
> RLS Policies
> Rules
> Triggers

```

Final_Destination/postgres@Postgres16

Query Query History

```

3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373

        RETURN NULL;
END;
$$ LANGUAGE plpgsql;

-- trigger perubahan rating insert review, update review,
CREATE TRIGGER trg_update_collection_rating
AFTER INSERT OR UPDATE OR DELETE
ON reviews
FOR EACH ROW
EXECUTE FUNCTION update_collection_rating();

```

Data Output Messages Notifications

CREATE TRIGGER

Query returned successfully in 237 msec.

Total rows: Query complete 00:00:00.237 CRLF Ln 3366, Col 1

27°C Hujan ringan

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

```

> collections_songs
> create_songs
> follow_artists
> follow_users
> genres
> like_reviews
> like_songs
> listens
> plibrary
> playlists
> rate_songs
> releases
> reviews
> Columns (6)
    review
    rating
    TIMESTAMP
    review_id
    user_id
    collection_id
> Constraints
> Indexes
> RLS Policies
> Rules
> Triggers

```

Final_Destination/postgres@Postgres16

Query Query History

```

3368 AFTER INSERT OR UPDATE OR DELETE
3369 ON reviews
3370 FOR EACH ROW
3371 EXECUTE FUNCTION update_collection_rating();
3372
3373
3374
3375
3376
3377
3378
3379
3380

SELECT * FROM reviews;
INSERT INTO reviews (review, rating, review_id, user_id,
VALUES ('bagussss', 5, 1, 10, 1);
SELECT collection_rating FROM collections WHERE collectio

```

Data Output Messages Notifications

INSERT 0 1

Query returned successfully in 491 msec.

Total rows: Query complete 00:00:00.491 CRLF Ln 3375, Col 1

Hari hujan yang... 27°C

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

```

> collections_songs
> create_songs
> follow_artists
> follow_users
> genres
> like_reviews
> like_songs
> listens
> plibrary
> playlists
> rate_songs
> releases
> reviews
> Columns (6)
    review
    rating
    TIMESTAMP
    review_id
    user_id
    collection_id
> Constraints
> Indexes
> RLS Policies
> Rules
> Triggers

```

Final_Destination/postgres@Postgres16

Query Query History

```

3368 AFTER INSERT OR UPDATE OR DELETE
3369 ON reviews
3370 FOR EACH ROW
3371 EXECUTE FUNCTION update_collection_rating();
3372
3373
3374
3375
3376
3377
3378
3379
3380

SELECT * FROM reviews;
INSERT INTO reviews (review, rating, review_id, user_id,
VALUES ('bagussss', 5, 1, 10, 1);
SELECT collection_rating FROM collections WHERE collectio

```

Data Output Messages Notifications

Showing rows: 1 to 1 Page No: 1 of 1 <> << >> >

collection_rating	lock
1	5

Successfully run. Total query runtime: 226 msec. 1 rows affected.

Total rows: 1 Query complete 00:00:00.226 CRLF Ln 3378, Col 1

2 cm hujan Minggu

pgAdmin 4

Object Explorer

```

FOR EACH ROW
EXECUTE FUNCTION update_collection_rating();

SELECT * FROM reviews;

INSERT INTO reviews (review, rating, review_id, user_id,
VALUES ('bagusss', 5, 1, 10, 1);

SELECT collection_rating FROM collections WHERE collective
-- update collection_rating nya
UPDATE reviews
SET rating = 3

```

Data Output

review	rating	TIMESTAMP	review_id	user_id	collection_id
bagusss	5	2023-11-21 16:46:24.933663	1	10	1

Showing rows: 1 to 1 Page No: 1 of 1

Total rows: 1 Query complete 00:00:00.161 CRLF Ln 3372; Col 1

-- update

pgAdmin 4

Object Explorer

```

INSERT INTO reviews (review, rating, review_id, user_id,
VALUES ('bagusss', 5, 1, 10, 1);

SELECT collection_rating FROM collections WHERE collective
-- update collection_rating nya
UPDATE reviews
SET rating = 3
WHERE review_id = 1;

```

Data Output

UPDATE 1

Query returned successfully in 253 msec.

Total rows: 1 Query complete 00:00:00.253 CRLF Ln 3380; Col 1

The screenshot shows the pgAdmin 4 interface. In the Object Explorer, there is a database named 'Final_Destination' containing several schemas and tables. The 'reviews' table is selected, showing its columns: review (text), rating (numeric(3)), timestamp (TIMESTAMP), review_id (integer, PK), user_id (integer), and collection_id (integer). A query window displays a sequence of SQL commands. The last command is an INSERT INTO statement:

```
ON reviews
FOR EACH ROW
EXECUTE FUNCTION update_collection_rating();

SELECT * FROM reviews;

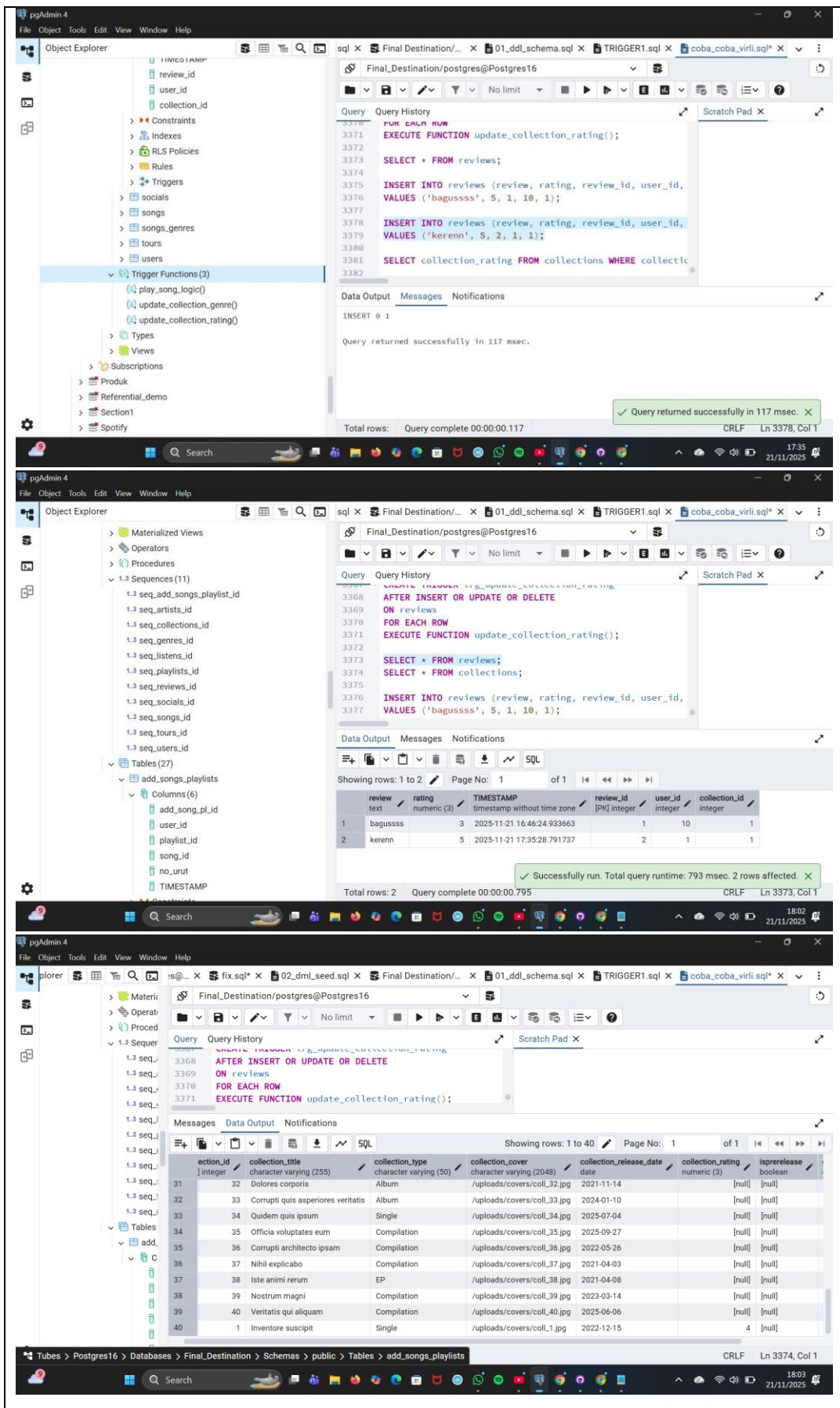
INSERT INTO reviews (review, rating, review_id, user_id,
VALUES ('bagussss', 5, 1, 10, 1);
```

The Data Output tab shows the result of the last query:

	review	rating	timestamp	review_id	user_id	collection_id
1	bagussss	3	2025-11-21 16:46:24.933663	1	10	1

A message at the bottom right indicates: "Successfully run. Total query runtime: 148 msec. 1 rows affected."

-- nambah review baru dari user_id berbeda



Tabel 122 PL/SQL - Trigger trg_reorder_playlist_sequence

Nama Trigger	reorder_playlist_sequence
Aksi	DELETE
Jenis	AFTER
Deskripsi	Mengubah urutan lagu saat lagu dihapus dari playlist
Script SQL	
<pre>-- Cek data SELECT user_id, playlist_id, song_id, no_urut FROM add_songs_playlists WHERE playlist_id = 2 ORDER BY no_urut; --tes trigger -- Hapus lagu urutan ke-3 di Playlist 2</pre>	

```
DELETE FROM add_songs_playlists  
WHERE playlist_id = 2  
AND no_urut = 3;
```

--cek data

```
SELECT  
    song_id,  
    no_urut  
FROM add_songs_playlists  
WHERE playlist_id = 2  
ORDER BY no_urut;
```

Screenshot Hasil

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

Dashboards Properties Statistics Dependencies Processes spotify@postgres@PostgreSQL_17

Servers (1) Databases (6) HR add_songs_playlist Song Artist Genre Playlist

Tables (27)

Query Query History

```

1: --del data
2: DELETE FROM add_songs_playlist WHERE playlist_id = 2
3: 
4: WITH add_songs_playlist AS
5: (
6:   SELECT user_id, playlist_id, song_id, no_uritxt
7:   FROM add_songs_playlist
8:   WHERE playlist_id = 2
9:   AND no_uritxt = 3
10: )
11: 
12: --coo data
13: SELECT
14:   user_id,
15:   playlist_id,
16:   no_uritxt
17:   FROM add_songs_playlist
18:   WHERE playlist_id = 2
19:   ORDER BY no_uritxt;
20:

```

Data Output Messages Notifications

	user_id	playlist_id	no_uritxt
1	1	2	3
2	1	2	2
3	1	2	1
4	1	2	4
5	1	2	5

Total rows: 5 - Query complete in 00:00:00.121 - Success! Total query runtime: 121 msec, 5 rows affected. OMLF - Ln 1, Col 1

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

Dashboards Properties Statistics Dependencies Processes spotify@postgres@PostgreSQL_17

Servers (1) Databases (6) HR add_songs_playlist Song Artist Genre Playlist

Tables (27)

Query Query History

```

1: --del data
2: DELETE FROM add_songs_playlist WHERE playlist_id = 2
3: 
4: WITH add_songs_playlist AS
5: (
6:   SELECT user_id, playlist_id, song_id, no_uritxt
7:   FROM add_songs_playlist
8:   WHERE playlist_id = 2
9:   AND no_uritxt = 3
10: )
11: 
12: --coo data
13: SELECT
14:   user_id,
15:   playlist_id,
16:   no_uritxt
17:   FROM add_songs_playlist
18:   WHERE playlist_id = 2
19:   ORDER BY no_uritxt;
20:

```

Data Output Messages Notifications

delete 1

Query returned successfully in 47 msec.

Total rows: 0 - Query complete in 00:00:00.047 - Success! Total query runtime: 47 msec. OMLF - Ln 1, Col 1

```

-- Cek data
1.   SELECT no_urut, playlist_id, song_id, no_urut
2.   FROM add_song_playlist
3.   WHERE playlist_id = 2
4.   ORDER BY no_urut;
5.
6. --触发器
7. --on insert untuk urutan keadaan di Playlist 1
8. CREATE TRIGGER trg_enforce_block_logic
9. BEFORE INSERT ON add_song_playlist
10.    FOR EACH ROW
11.      WHERE new.playlist_id = 1
12.          AND new.no_urut > 2
13.
14. DELETE FROM add_song_playlist
15. WHERE add_song_playlist.playlist_id = 1
16.     AND add_song_playlist.no_urut > 2
17.
18. ORDER BY no_urut;
19.
20.
21.
22.
23.
24.
25.
26.
27.
28.
29.
30.
31.
32.
33.
34.
35.
36.
37.
38.
39.
40.
41.
42.
43.
44.
45.
46.
47.
48.
49.
50.
51.
52.
53.
54.
55.
56.
57.
58.
59.
60.
61.
62.
63.
64.
65.
66.
67.
68.
69.
70.
71.
72.
73.
74.
75.
76.
77.
78.
79.
80.
81.
82.
83.
84.
85.
86.
87.
88.
89.
90.
91.
92.
93.
94.
95.
96.
97.
98.
99.
100.
101.
102.
103.
104.
105.
106.
107.
108.
109.
110.
111.
112.
113.
114.
115.
116.
117.
118.
119.
120.
121.
122.
123.
124.
125.
126.
127.
128.
129.
130.
131.
132.
133.
134.
135.
136.
137.
138.
139.
140.
141.
142.
143.
144.
145.
146.
147.
148.
149.
150.
151.
152.
153.
154.
155.
156.
157.
158.
159.
160.
161.
162.
163.
164.
165.
166.
167.
168.
169.
170.
171.
172.
173.
174.
175.
176.
177.
178.
179.
180.
181.
182.
183.
184.
185.
186.
187.
188.
189.
190.
191.
192.
193.
194.
195.
196.
197.
198.
199.
200.
201.
202.
203.
204.
205.
206.
207.
208.
209.
210.
211.
212.
213.
214.
215.
216.
217.
218.
219.
220.
221.
222.
223.
224.
225.
226.
227.
228.
229.
230.
231.
232.
233.
234.
235.
236.
237.
238.
239.
240.
241.
242.
243.
244.
245.
246.
247.
248.
249.
250.
251.
252.
253.
254.
255.
256.
257.
258.
259.
259.
260.
261.
262.
263.
264.
265.
266.
267.
268.
269.
270.
271.
272.
273.
274.
275.
276.
277.
278.
279.
279.
280.
281.
282.
283.
284.
285.
286.
287.
288.
289.
289.
290.
291.
292.
293.
294.
295.
296.
297.
298.
299.
299.
300.
301.
302.
303.
304.
305.
306.
307.
308.
309.
309.
310.
311.
312.
313.
314.
315.
316.
317.
318.
319.
319.
320.
321.
322.
323.
324.
325.
326.
327.
328.
329.
329.
330.
331.
332.
333.
334.
335.
336.
337.
338.
339.
339.
340.
341.
342.
343.
344.
345.
346.
347.
348.
349.
349.
350.
351.
352.
353.
354.
355.
356.
357.
358.
359.
359.
360.
361.
362.
363.
364.
365.
366.
367.
368.
369.
369.
370.
371.
372.
373.
374.
375.
376.
377.
378.
379.
379.
380.
381.
382.
383.
384.
385.
386.
387.
388.
389.
389.
390.
391.
392.
393.
394.
395.
396.
397.
398.
399.
399.
400.
401.
402.
403.
404.
405.
406.
407.
408.
409.
409.
410.
411.
412.
413.
414.
415.
416.
417.
418.
419.
419.
420.
421.
422.
423.
424.
425.
426.
427.
428.
429.
429.
430.
431.
432.
433.
434.
435.
436.
437.
438.
439.
439.
440.
441.
442.
443.
444.
445.
446.
447.
448.
449.
449.
450.
451.
452.
453.
454.
455.
456.
457.
458.
459.
459.
460.
461.
462.
463.
464.
465.
466.
467.
468.
469.
469.
470.
471.
472.
473.
474.
475.
476.
477.
478.
479.
479.
480.
481.
482.
483.
484.
485.
486.
487.
488.
489.
489.
490.
491.
492.
493.
494.
495.
496.
497.
498.
499.
499.
500.
501.
502.
503.
504.
505.
506.
507.
508.
509.
509.
510.
511.
512.
513.
514.
515.
516.
517.
518.
519.
519.
520.
521.
522.
523.
524.
525.
526.
527.
528.
529.
529.
530.
531.
532.
533.
534.
535.
536.
537.
538.
539.
539.
540.
541.
542.
543.
544.
545.
546.
547.
548.
549.
549.
550.
551.
552.
553.
554.
555.
556.
557.
558.
559.
559.
560.
561.
562.
563.
564.
565.
566.
567.
568.
569.
569.
570.
571.
572.
573.
574.
575.
576.
577.
578.
579.
579.
580.
581.
582.
583.
584.
585.
586.
587.
588.
589.
589.
590.
591.
592.
593.
594.
595.
596.
597.
598.
599.
599.
600.
601.
602.
603.
604.
605.
606.
607.
608.
609.
609.
610.
611.
612.
613.
614.
615.
616.
617.
618.
619.
619.
620.
621.
622.
623.
624.
625.
626.
627.
628.
629.
629.
630.
631.
632.
633.
634.
635.
636.
637.
638.
639.
639.
640.
641.
642.
643.
644.
645.
646.
647.
648.
649.
649.
650.
651.
652.
653.
654.
655.
656.
657.
658.
659.
659.
660.
661.
662.
663.
664.
665.
666.
667.
668.
669.
669.
670.
671.
672.
673.
674.
675.
676.
677.
678.
679.
679.
680.
681.
682.
683.
684.
685.
686.
687.
688.
689.
689.
690.
691.
692.
693.
694.
695.
696.
697.
698.
698.
699.
700.
701.
702.
703.
704.
705.
706.
707.
708.
709.
709.
710.
711.
712.
713.
714.
715.
716.
717.
718.
719.
719.
720.
721.
722.
723.
724.
725.
726.
727.
728.
729.
729.
730.
731.
732.
733.
734.
735.
736.
737.
738.
739.
739.
740.
741.
742.
743.
744.
745.
746.
747.
748.
749.
749.
750.
751.
752.
753.
754.
755.
756.
757.
758.
759.
759.
760.
761.
762.
763.
764.
765.
766.
767.
768.
769.
769.
770.
771.
772.
773.
774.
775.
776.
777.
778.
779.
779.
780.
781.
782.
783.
784.
785.
786.
787.
788.
789.
789.
790.
791.
792.
793.
794.
795.
796.
797.
798.
798.
799.
800.
801.
802.
803.
804.
805.
806.
807.
808.
809.
809.
810.
811.
812.
813.
814.
815.
816.
817.
818.
819.
819.
820.
821.
822.
823.
824.
825.
826.
827.
828.
829.
829.
830.
831.
832.
833.
834.
835.
836.
837.
838.
839.
839.
840.
841.
842.
843.
844.
845.
846.
847.
848.
849.
849.
850.
851.
852.
853.
854.
855.
856.
857.
858.
859.
859.
860.
861.
862.
863.
864.
865.
866.
867.
868.
869.
869.
870.
871.
872.
873.
874.
875.
876.
877.
878.
879.
879.
880.
881.
882.
883.
884.
885.
886.
887.
888.
889.
889.
890.
891.
892.
893.
894.
895.
896.
897.
898.
898.
899.
900.
901.
902.
903.
904.
905.
906.
907.
908.
909.
909.
910.
911.
912.
913.
914.
915.
916.
917.
917.
918.
919.
920.
921.
922.
923.
924.
925.
926.
927.
928.
929.
929.
930.
931.
932.
933.
934.
935.
936.
937.
938.
939.
939.
940.
941.
942.
943.
944.
945.
946.
947.
948.
949.
949.
950.
951.
952.
953.
954.
955.
956.
957.
958.
959.
959.
960.
961.
962.
963.
964.
965.
966.
967.
968.
969.
969.
970.
971.
972.
973.
974.
975.
976.
977.
978.
979.
979.
980.
981.
982.
983.
984.
985.
986.
987.
988.
989.
989.
990.
991.
992.
993.
994.
995.
996.
997.
998.
999.
999.
1000.
1001.
1002.
1003.
1004.
1005.
1006.
1007.
1008.
1009.
1009.
1010.
1011.
1012.
1013.
1014.
1015.
1016.
1017.
1017.
1018.
1019.
1019.
1020.
1021.
1022.
1023.
1024.
1025.
1026.
1027.
1028.
1029.
1029.
1030.
1031.
1032.
1033.
1034.
1035.
1036.
1037.
1038.
1039.
1039.
1040.
1041.
1042.
1043.
1044.
1045.
1046.
1047.
1048.
1049.
1049.
1050.
1051.
1052.
1053.
1054.
1055.
1056.
1057.
1058.
1059.
1059.
1060.
1061.
1062.
1063.
1064.
1065.
1066.
1067.
1068.
1069.
1069.
1070.
1071.
1072.
1073.
1074.
1075.
1076.
1077.
1078.
1079.
1079.
1080.
1081.
1082.
1083.
1084.
1085.
1086.
1087.
1088.
1089.
1089.
1090.
1091.
1092.
1093.
1094.
1095.
1096.
1097.
1097.
1098.
1099.
1099.
1100.
1101.
1102.
1103.
1104.
1105.
1106.
1107.
1108.
1109.
1109.
1110.
1111.
1112.
1113.
1114.
1115.
1116.
1117.
1118.
1119.
1119.
1120.
1121.
1122.
1123.
1124.
1125.
1126.
1127.
1128.
1129.
1129.
1130.
1131.
1132.
1133.
1134.
1135.
1136.
1137.
1138.
1139.
1139.
1140.
1141.
1142.
1143.
1144.
1145.
1146.
1147.
1148.
1149.
1149.
1150.
1151.
1152.
1153.
1154.
1155.
1156.
1157.
1158.
1159.
1159.
1160.
1161.
1162.
1163.
1164.
1165.
1166.
1167.
1168.
1169.
1169.
1170.
1171.
1172.
1173.
1174.
1175.
1176.
1177.
1178.
1178.
1179.
1180.
1181.
1182.
1183.
1184.
1185.
1186.
1187.
1187.
1188.
1189.
1190.
1191.
1192.
1193.
1194.
1195.
1196.
1197.
1197.
1198.
1199.
1199.
1200.
1201.
1202.
1203.
1204.
1205.
1206.
1207.
1208.
1209.
1209.
1210.
1211.
1212.
1213.
1214.
1215.
1216.
1217.
1218.
1219.
1219.
1220.
1221.
1222.
1223.
1224.
1225.
1226.
1227.
1228.
1229.
1229.
1230.
1231.
1232.
1233.
1234.
1235.
1236.
1237.
1238.
1239.
1239.
1240.
1241.
1242.
1243.
1244.
1245.
1246.
1247.
1248.
1249.
1249.
1250.
1251.
1252.
1253.
1254.
1255.
1256.
1257.
1258.
1259.
1259.
1260.
1261.
1262.
1263.
1264.
1265.
1266.
1267.
1268.
1269.
1269.
1270.
1271.
1272.
1273.
1274.
1275.
1276.
1277.
1278.
1278.
1279.
1280.
1281.
1282.
1283.
1284.
1285.
1286.
1287.
1287.
1288.
1289.
1290.
1291.
1292.
1293.
1294.
1295.
1295.
1296.
1297.
1298.
1299.
1299.
1300.
1301.
1302.
1303.
1304.
1305.
1306.
1307.
1308.
1309.
1309.
1310.
1311.
1312.
1313.
1314.
1315.
1316.
1317.
1318.
1319.
1319.
1320.
1321.
1322.
1323.
1324.
1325.
1326.
1327.
1328.
1329.
1329.
1330.
1331.
1332.
1333.
1334.
1335.
1336.
1337.
1338.
1339.
1339.
1340.
1341.
1342.
1343.
1344.
1345.
1346.
1347.
1348.
1349.
1349.
1350.
1351.
1352.
1353.
1354.
1355.
1356.
1357.
1358.
1359.
1359.
1360.
1361.
1362.
1363.
1364.
1365.
1366.
1367.
1368.
1369.
1369.
1370.
1371.
1372.
1373.
1374.
1375.
1376.
1377.
1378.
1378.
1379.
1380.
1381.
1382.
1383.
1384.
1385.
1386.
1387.
1387.
1388.
1389.
1390.
1391.
1392.
1393.
1394.
1395.
1395.
1396.
1397.
1398.
1399.
1399.
1400.
1401.
1402.
1403.
1404.
1405.
1406.
1407.
1408.
1408.
1409.
1410.
1411.
1412.
1413.
1414.
1415.
1416.
1417.
1418.
1419.
1419.
1420.
1421.
1422.
1423.
1424.
1425.
1426.
1427.
1428.
1429.
1429.
1430.
1431.
1432.
1433.
1434.
1435.
1436.
1437.
1438.
1439.
1439.
1440.
1441.
1442.
1443.
1444.
1445.
1446.
1447.
1448.
1449.
1449.
1450.
1451.
1452.
1453.
1454.
1455.
1456.
1457.
1458.
1459.
1459.
1460.
1461.
1462.
1463.
1464.
1465.
1466.
1467.
1468.
1469.
1469.
1470.
1471.
1472.
1473.
1474.
1475.
1476.
1477.
1478.
1478.
1479.
1480.
1481.
1482.
1483.
1484.
1485.
1486.
1487.
1487.
1488.
1489.
1490.
1491.
1492.
1493.
1494.
1495.
1495.
1496.
1497.
1498.
1499.
1499.
1500.
1501.
1502.
1503.
1504.
1505.
1506.
1507.
1508.
1508.
1509.
1510.
1511.
1512.
1513.
1514.
1515.
1516.
1517.
1518.
1519.
1519.
1520.
1521.
1522.
1523.
1524.
1525.
1526.
1527.
1528.
1529.
1529.
1530.
1531.
1532.
1533.
1534.
1535.
1536.
1537.
1538.
1539.
1539.
1540.
1541.
1542.
1543.
1544.
1545.
1546.
1547.
1548.
1549.
1549.
1550.
1551.
1552.
1553.
1554.
1555.
1556.
1557.
1558.
1559.
1559.
1560.
1561.
1562.
1563.
1564.
1565.
1566.
1567.
1568.
1569.
1569.
1570.
1571.
1572.
1573.
1574.
1575.
1576.
1577.
1578.
1578.
1579.
1580.
1581.
1582.
1583.
1584.
1585.
1586.
1587.
1587.
1588.
1589.
1590.
1591.
1592.
1593.
1594.
1595.
1595.
1596.
1597.
1598.
1599.
1599.
1600.
1601.
1602.
1603.
1604.
1605.
1606.
1607.
1608.
1608.
1609.
1610.
1611.
1612.
1613.
1614.
1615.
1616.
1617.
1618.
1619.
1619.
1620.
1621.
1622.
1623.
1624.
1625.
1626.
1627.
1628.
1629.
1629.
1630.
1631.
1632.
1633.
1634.
1635.
1636.
1637.
1638.
1639.
1639.
1640.
1641.
1642.
1643.
1644.
1645.
1646.
1647.
1648.
1649.
1649.
1650.
1651.
1652.
1653.
1654.
1655.
1656.
1657.
1658.
1659.
1659.
1660.
1661.
1662.
1663.
1664.
1665.
1666.
1667.
1668.
1669.
1669.
1670.
1671.
1672.
1673.
1674.
1675.
1676.
1677.
1678.
1678.
1679.
1680.
1681.
1682.
1683.
1684.
1685.
1686.
1687.
1687.
1688.
1689.
1690.
1691.
1692.
1693.
1694.
1695.
1695.
1696.
1697.
1698.
1699.
1699.
1700.
1701.
1702.
1703.
1704.
1705.
1706.
1707.
1708.
1708.
1709.
1710.
1711.
1712.
1713.
1714.
1715.
1716.
1717.
1718.
1719.
1719.
1720.
1721.
1722.
1723.
1724.
1725.
1726.
1727.
1728.
1729.
1729.
1730.
1731.
1732.
1733.
1734.
1735.
1736.
1737.
1738.
1739.
1739.
1740.
1741.
1742.
1743.
1744.
1745.
1746.
1747.
1748.
1749.
1749.
1750.
1751.
1752.
1753.
1754.
1755.
1756.
1757.
1758.
1759.
1759.
1760.
1761.
1762.
1763.
1764.
1765.
1766.
1767.
1768.
1769.
1769.
1770.
1771.
1772.
1773.
1774.
1775.
1776.
1777.
1778.
1778.
1779.
1780.
1781.
1782.
1783.
1784.
1785.
1786.
1787.
1787.
1788.
1789.
1790.
1791.
1792.
1793.
1794.
1795.
1795.
1796.
1797.
1798.
1799.
1799.
1800.
1801.
1802.
1803.
1804.
1805.
1806.
1807.
1808.
1808.
1809.
1810.
1811.
1812.
1813.
1814.
1815.
1816.
1817.
1818.
1819.
1819.
1820.
1821.
1822.
1823.
1824.
1825.
1826.
1827.
1828.
1829.
1829.
1830.
1831.
1832.
1833.
1834.
1835.
1836.
1837.
1838.
1839.
1839.
1840.
1841.
1842.
1843.
1844.
1845.
1846.
1847.
1848.
1849.
1849.
1850.
1851.
1852.
1853.
1854.
1855.
1856.
1857.
1858.
1859.
1859.
1860.
1861.
1862.
1863.
1864.
1865.
1866.
1867.
1868.
1869.
1869.
1870.
1871.
1872.
1873.
1874.
1875.
1876.
1877.
1878.
1878.
1879.
1880.
1881.
1882.
1883.
1884.
1885.
1886.
1887.
1887.
1888.
1889.
1890.
1891.
1892.
1893.
1894.
1895.
1895.
1896.
1897.
1898.
1899.
1899.
1900.
1901.
1902.
1903.
1904.
1905.
1906.
1907.
1908.
1908.
1909.
1910.
1911.
1912.
1913.
1914.
1915.
1916.
1917.
1918.
1919.
1919.
1920.
1921.
1922.
1923.
1924.
1925.
1926.
1927.
1928.
1929.
1929.
1930.
1931.
1932.
1933.
1934.
1935.
1936.
1937.
1938.
1939.
1939.
1940.
1941.
1942.
1943.
1944.
1945.
1946.
1947.
1948.
1949.
1949.
1950.
1951.
1952.
1953.
1954.
1955.
1956.
1957.
1958.
1959.
1959.
1960.
1961.
1962.
1963.
1964.
1965.
1966.
1967.
1968.
1969.
1969.
1970.
1971.
1972.
1973.
1974.
1975.
1976.
1977.
1978.
1978.
1979.
1980.
1981.
1982.
1983.
1984.
1985.
1986.
1987.
1988.
1989.
1989.
1990.
1991.
1992.
1993.
1994.
1995.
1996.
1997.
1998.
1999.
1999.
2000.
2001.
2002.
2003.
2004.
2005.
2006.
2007.
2008.
2009.
2009.
2010.
2011.
2012.
2013.
2014.
2015.
2016.
2017.
2018.
2019.
2019.
2020.
2021.
2022.
2023.
2024.
2025.
2026.
2027.
2028.
2029.
2029.
2030.
2031.
2032.
2033.
2034.
2035.
2036.
2037.
2038.
2039.
2039.
2040.
2041.
2042.
2043.
2044.
2045.
2046.
2047.
2048.
2049.
2049.
2050.
2051.
2052.
2053.
2054.
2055.
2056.
2057.
2058.
2059.
2059.
2060.
2061.
2062.
2063.
2064.
2065.
2066.
2067.
2068.
2069.
2069.
2070.
2071.
2072.
2073.
2074.
2075.
2076.
2077.
2078.
2078.
2079.
2080.
2081.
2082.
2083.
2084.
2085.
2086.
2087.
2088.
2089.
2089.
2090.
2091.
2092.
2093.
2094.
2095.
2096.
2097.
2098.
2099.
2099.
2100.
2101.
2102.
2103.
2104.
2105.
2106.
2107.
2108.
2108.
2109.
2110.
2111.
2112.
2113.
2114.
2115.
2116.
2117.
2118.
2119.
2119.
2120.
2121.
2122.
2123.
2124.
2125.
2126.
2127.
2128.
2129.
2129.
2130.
2131.
2132.
2133.
2134.
2135.
2136.
2137.
2138.
2139.
2139.
2140.
2141.
2142.
2143.
2144.
2145.
2146.
2147.
2148.
2149.
2149.
2150.
2151.
2152.
2153.
2154.
2155.
2156.
2157.
2158.
2159.
2159.
2160.
2161.
2162.
2163.
2164.
2165.
2166.
2167.
2168.
2169.
2169.
2170.
2171.
2172.
2173.
2174.
2175.
2176.
2177.
2178.
2178.
2179.
2180.
2181.
2182.
2183.
2184.
2185.
2186.
2187.
2188.
2189.
2189.
2190.
2191.
2192.
2193.
2194.
2195.
2196.
2197.
2198.
2199.
2199.
2200.
2201.
2202.
2203.
2204.
2205.
2206.
2207.
2208.
2208.
2209.
2210.
22
```

Deskripsi	Otomatis men-unfollow user apabila user mem-block user
--menyiapkan data	<pre>INSERT INTO follow_users (follower_id, followed_id) VALUES (1, 2); INSERT INTO follow_users (follower_id, followed_id) VALUES (2, 1);</pre>

--cek

`SELECT * FROM follow_users`

`WHERE (follower_id = 1 AND followed_id = 2) OR (follower_id = 2 AND followed_id = 1);`

--trigger

`INSERT INTO block_users (blocker_id, blocked_id)`

`VALUES (1, 2);`

--cek

`SELECT * FROM follow_users`

`WHERE (follower_id = 1 AND followed_id = 2)`

OR (follower_id = 2 AND followed_id = 1);

Screenshot Hasil

The screenshot shows two pgAdmin 4 sessions side-by-side.

Session 1 (Left):

```
1 INSERT INTO follower_users (follower_id, followed_id) VALUES (1, 2);
2 INSERT INTO follower_users (follower_id, followed_id) VALUES (2, 1);
```

Session 2 (Right):

```
1 SELECT * FROM follower_users
2 WHERE (follower_id = 1 AND followed_id = 2)
3 OR (follower_id = 2 AND followed_id = 1);
```

follower_id	followed_id
1	2
2	1

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

Dashboards Properties Statistics Dependencies Dependencies x Processes x specify/postgres@PostgreSQL_17 x

Servers (1)

- PostgreSQL_17
 - Databases (6)
 - HR
 - HR
 - business_process
 - lv
 - nyct
 - public
 - Tables (27)
 - add_song_playlist
 - artist_promotion
 - artists
 - artists_hours
 - block_users
 - block_user_stats
 - collection_library
 - collection_song_3_genres
 - collections
 - collection_song
 - create_song
 - follow_artists
 - follow_user
 - genres
 - like_movies
 - like_songs
 - library
 - playlists
 - rate_song
 - releases
 - reviews
 - sociale
 - songs
 - song_genres

Query Query History

```
1. INSERT INTO block_users (blocker_id, blocked_id)
2. VALUES (1, 2);
```

Data Output Messages Notifications

user: x1

Query returned successfully in 61 msec.

Total rows: 0 Query complete 00:00:00.001

Query returned successfully in 61 msec. CRLF Ln 2. Do

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

Dashboards Properties Statistics Dependencies Dependencies x Processes x specify/postgres@PostgreSQL_17 x

Servers (1)

- PostgreSQL_17
 - Databases (6)
 - HR
 - HR
 - business_process
 - lv
 - nyct
 - public
 - Tables (27)
 - add_song_playlist
 - artist_promotion
 - artists
 - artists_hours
 - block_users
 - block_user_stats
 - collection_library
 - collection_song_3_genres
 - collections
 - collection_song
 - create_song
 - follow_artists
 - follow_user
 - genres
 - like_movies
 - like_songs
 - library
 - playlists
 - rate_song
 - releases
 - reviews
 - sociale
 - songs
 - song_genres

Query Query History

```
1. SELECT * FROM follower_user
2. WHERE (follower_id = 1 AND followed_id = 2)
3. OR (follower_id = 2 AND followed_id = 1);
```

Data Output Messages Notifications

Follower_ID Followed_ID

1 2

2 1

Successfully run. Total query runtime: 90 msec. 0 rows affected.

CRLF Ln 2. Do

6. KESIMPULAN

Pada tugas besar ini, kelompok berhasil merancang dan mengimplementasikan sistem basis data untuk layanan *Music Streaming Service* sesuai ruang lingkup perencanaan yang telah ditetapkan. Implementasi meliputi modul inti seperti manajemen pengguna, artis, lagu, koleksi/album, playlist, interaksi sosial antar pengguna, sistem rating-review, promosi artis, serta pencarian terintegrasi. Struktur database yang dibangun telah memenuhi kebutuhan normalisasi, integritas referensial, dan keamanan (termasuk hashing password), serta menerapkan business rule yang ketat untuk menjamin konsistensi.

Fitur-fitur penting yang telah diimplementasikan meliputi:

- a. Registrasi dan login pengguna.
- b. Sistem follow antar pengguna dan antar artis.
- c. Pembuatan review, rating, serta like/unlike review.
- d. Pengelolaan playlist, collaborative playlist, dan manajemen track.
- e. Manajemen koleksi (album/EP/single/compilation) beserta informasi detail dan tracklist.
- f. Manajemen artis, tur artis, dan promosi koleksi.
- g. Validasi tanggal prerelease koleksi dan tanggal tur melalui trigger.
- h. Fungsi pencarian mencakup lagu, koleksi, playlist, artis, dan pengguna.

Selain progress yang sudah tercapai, masih terdapat beberapa ruang lingkup yang *belum diimplementasikan* dari perencanaan awal, yaitu:

- a. Sistem streaming (pencatatan play count, riwayat pemutaran).
- b. Sistem rekomendasi berbasis riwayat pengguna atau popularitas.
- c. Fitur blocking/blacklist antar pengguna secara penuh (baru dicatat dalam tabel namun belum memiliki prosedur lengkap).
- d. Pengelolaan genre secara lanjutan, seperti rekomendasi genre atau kategori trending.
- e. Analytics artis, seperti statistik mendalam terkait streaming, tren bulanan, atau data chart otomatis.
- f. CRUD admin untuk mengelola konten (lagu, artis, genre, koleksi) secara sistematis.
- g. Moderasi konten review (report, hidden review, audit log).

Dalam proses penggerjaan, beberapa kendala utama muncul, seperti penanganan relasi many-to-many yang kompleks, konsistensi tipe data pada fungsi yang memiliki aturan return ketat, desain trigger yang harus mengikuti business rule spesifik, serta penyesuaian query dalam view yang memiliki agregasi yang rumit.

Secara keseluruhan, progress penggerjaan telah memenuhi tujuan utama studi kasus, yaitu menghasilkan rancangan database yang terstruktur, aman, scalable, dan sesuai kebutuhan operasional layanan musik digital. Pekerjaan ini memberikan pembelajaran penting dalam mengintegrasikan desain skema, business rule, implementasi SQL, trigger, view, hingga prosedur/fungsi secara end-to-end.

7. PEMBAGIAN TUGAS

Persentase kontribusi anggota:

Tabel Kontribusi Pengerjaan

Nama	Kontribusi Pengerjaan (%)
Arkan Ramadhan Nugraha	25%
Fauzi Ismail	25%
Nurahma Rahayu	25%
Virli Nasyila Putri	25%

Rincian kontribusi pekerjaan masing-masing anggota:

Tabel Daftar Pengerjaan

Nama	Daftar Pekerjaan
Arkan Ramadhan Nugraha	<ol style="list-style-type: none">Menentukan Entity-entity dan Relationship-nya dan juga membuat ERDMenetukan Domain Constraint (selain tipe data), seperti unique, not null, check, dll.Membuat semua cronjobDebugging/fixing LDM dan PDMMembuat semua viewIde awal dan kontribusi signifikan di Ruang LingkupTrigger validate_tour_date()
Fauzi Ismail	<ol style="list-style-type: none">Membuat, debugging, fixing Logical Data Model. Menentukan seluruh tipe data tiap atribut tabelMembuat, debugging, fixing Physical Data Model(PDM). Menentukan seluruh referential integrity constraint

	<p>3. Membuat trigger user_block dan reorder_playlist</p> <p>4. Membuat beberapa function/procedure untuk fitur/modul collection top genres</p> <p>5. Membuat beberapa function/procedure untuk fitur/modul automatic_aggregations</p> <p>6. Membuat script generate data dummy dan integrasi terhadap API Spotify serta library faker</p>
Nurahma Rahayu	<p>1. Membuat Conceptual Data Model (CDM)</p> <p>2. Membuat function dan procedure untuk fitur manajemen user.</p> <p>3. Membuat function dan procedure untuk fitur collections.</p> <p>4. Membuat function dan procedure untuk fitur review collection.</p> <p>5. Membuat function dan procedure untuk fitur manajemen artist.</p> <p>6. Membuat function dan procedure untuk fitur promotion and tour management.</p> <p>7. Membuat function dan procedure untuk fitur search system.</p>
Virli Nasyila Putri	<p>1. Sequence (User_id, artist_id, song_id, playlist_id, collection_id, genre_id, review_id, tour_id, social_id, listen_id, add_songs_playlist_id)</p> <p>2. Membuat function dan procedure untuk fitur Playlist (create_playlist, add_song_to_playlist, remove_song_to_playlist, get_Playlist_detail, get_playlist_tracks)</p> <p>3. Membuat function dan procedure untuk fitur SONG & LIKE/RATING SYSTEM (get_song_detail_, toggle_like_song, rate_song)</p> <p>4. Membuat function dan procedure untuk fitur LISTENING HISTORY (log_listen, dan get_recently_played)</p>

	5. Membuat trigger playlist (fix_playlist_collaborators)
	6. Membuat trigger update_artist_follow_count
	7. Membuat trigger update_collection_rating
	8. Membuat trigger update_collection_top3_genres
	9. Membuat trigger update_review_timestamp
	10. Membuat trigger update_song_rating
	11. Membuat trigger validate_prerelease_date

8. LESSON LEARNED

Arkan Ramadhan Nugraha :

- Proyek ini memberikan pengalaman dalam merancang model data mulai dari ERD sederhana hingga menjadi LDM dan PDM.
- Saya belajar menetukan domain constraint sesuai kebutuhan.
- Saya belajar pentingnya pengorganisasian repository proyek agar penggerjaan database menjadi lebih terstruktur, rapi, dan mudah dikelola.
- Mempelajari penggunaan *view* untuk mempermudah pengambilan data dan mengurangi duplikasi query.
- Mengenal dan mengimplementasikan *cron job* dengan pg_cron untuk penjadwalan tugas otomatis.
- Memahami cara mendefinisikan ruang lingkup proyek agar pengembangan lebih terarah.

Fauzi Ismail :

- Proyek ini memberikan pemahaman bagaimana merancang schema database untuk sebuah aplikasi
- Mempelajari mengenai tipe data dan pentingnya menentukan tipe data yang sesuai bagi schema database/DDL agar optimal dan tidak menghambutkan banyak ruang.
- Mempelajari lebih mendalam mengenai implementasi referential integrity constraints dan hubungannya dengan DML agar menjaga relasi antar tabel serta konsistensi data
- Mempelajari mengenai business rule aplikasi streaming music melalui pembuatan beberapa trigger
- Mempelajari bagaimana view, function/procedure data mengoptimasi query dari sisi keterbacaan

Nurahma Rahayu :

- Proyek ini memberikan pemahaman mendalam tentang perancangan backend berbasis PostgreSQL, mulai dari desain schema, pembuatan function/procedure/trigger, modularisasi fitur hingga error handling.

- Pentingnya menentukan dan memahami struktur data dari tiap tablete, desain output function, dan konsistensi karena bisa berdampak ke pembuatan functionnya.
- Modularisasi fitur mempermudah pengelolaan function dan procedure yang dibutuhkan system, membantu agar kode lebih rapi, mudah dipahami, dan memudahkan pengembangan serta debugging.
- View dapat mempermudah function yang kompleks dan mengurangi duplikasi query.

Virli Nasyila Putri :

- Proyek ini memberikan pengetahuan lebih mendalam mengenai pembangunan backend dengan PostgreSQL, dari penyusunan schema, perancangan function/procedure/trigger.
- Implementasi dari trigger memastikan aturan bisnis berjalan otomatis di level database, menjaga integritas dan konsistensi data .
- Sequence yang konsisten dan jelas penamaannya membantu memastikan primary key unik dan aman saat insert berlangsung secara paralel.
- Perancangan dari sistem rating, like, dan listening history memberikan pemahaman lebih dalam data agregat dan log aktivitas harus diproses secara real-time.
- Pada proyek ini meningkatkan kemampuan dalam membuat query kompleks seperti join, aggregate function, dan looping dalam PL/pgSQL.
- Menambah pemahaman untuk pentingnya melakukan validasi input di dalam procedure, function, maupun trigger untuk menjaga keamanan data dan mencegah inkonsistensi.

Secara keseluruhan, proyek ini memberikan pengalaman komprehensif bagi kelompok dalam merancang dan membangun backend database menggunakan PostgreSQL, dimulai dari penyusunan ERD hingga pengembangan LDM dan PDM yang lengkap. Sepanjang prosesnya, kelompok mempelajari penerapan domain constraint, referential integrity, serta penggunaan function, procedure, trigger, view, dan sequence untuk memastikan aturan bisnis berjalan otomatis, konsisten, dan sesuai kebutuhan sistem. Melalui pengembangan modul seperti playlist, rating, listening history, search system, serta fitur promotion dan tour management, kelompok memperoleh pemahaman menyeluruh tentang bagaimana logika dan alur kerja aplikasi musik dijalankan pada sisi backend. Implementasi pg_cron juga memperkenalkan konsep penjadwalan tugas periodik yang tidak dapat ditangani oleh trigger, sehingga

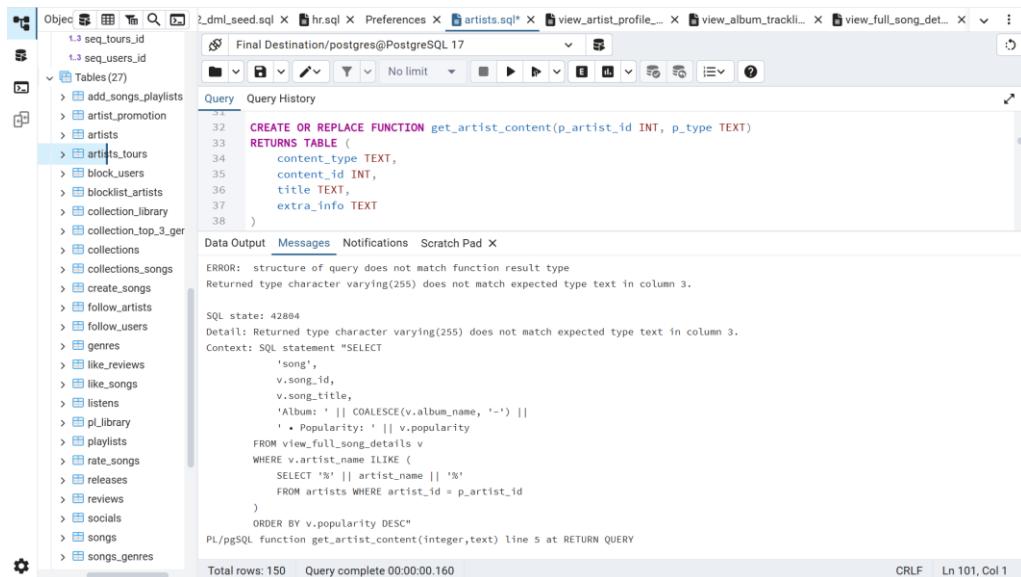
memperluas wawasan mengenai otomasi proses di tingkat database. Selain itu, proyek ini memperkuat kemampuan kelompok dalam menulis query kompleks, melakukan modularisasi fitur, mengelola struktur repository secara terorganisir, serta menyelaraskan ruang lingkup pengembangan agar sistem lebih terstruktur, efisien, dan mudah dikelola di tahap berikutnya.

9. REPOSITORY GITHUB

https://github.com/VirliNasyila/Final_Destination

Dokumentasi Masalah dan Solusi

- Masalah : tipe data kolom hasil query tidak sama dengan tipe data yang dideklarasikan di RETURN TABLE.



```

CREATE OR REPLACE FUNCTION get_artist_content(p_artist_id INT, p_type TEXT)
RETURNS TABLE (
    content_type TEXT,
    content_id INT,
    title TEXT,
    extra_info TEXT
)

```

Error: structure of query does not match function result type
 Returned type character varying(255) does not match expected type text in column 3.
 SQL state: 42884
 Detail: Returned type character varying(255) does not match expected type text in column 3.
 Context: SQL statement "SELECT
 'song',
 v.song_id,
 v.song_title,
 'Album: ' || COALESCE(v.album_name, '-') ||
 ' Popularity: ' || v.popularity
 FROM view_full_song_details v
 WHERE v.artist_name ILIKE (
 SELECT '%' || artist_name || '%'
 FROM artists WHERE artist_id = p_artist_id
)
 ORDER BY v.popularity DESC"
 PL/pgSQL function get_artist_content(integer,text) line 5 at RETURN QUERY

Total rows: 150 Query complete 00:00:00.160 CRLF Ln 101, Col 1

Solusi : Menambahkan cast ::TEXT pada semua kolom yang bukan TEXT

```

-- ===== SONGS =====
IF p_type = 'songs' THEN
  RETURN QUERY
  SELECT
    'song',
    v.song_id,
    v.song_title ::TEXT,
    'Album: ' || COALESCE(v.album_name, '-') || ' Popularity: ' || v.popularity
  FROM view_full_song_details v
  WHERE v.artist_name ILIKE (
    SELECT '%' || artist_name || '%'
    FROM artists WHERE artist_id = p_artist_id
  )
  ORDER BY v.popularity DESC;

```

- Masalah : Function mendefinisikan kolom kelima sebagai INTEGER, tetapi query di dalam RETURN QUERY menghasilkan kolom kelima sebagai BIGINT. Akibatnya

muncul error.

The screenshot shows the pgAdmin interface with a query editor window. The code is a function definition:

```
83 < BEGIN
84   RETURN QUERY
85   SELECT
86     c.collection_id,
87     c.collection_title ::TEXT,
```

An error message is displayed below the code:

ERROR: structure of query does not match function result type
Returned type bigint does not match expected type integer in column 5.

SQL state: 42884
Detail: Returned type bigint does not match expected type integer in column 5.
Context: SQL statement "SELECT
 c.collection_id,
 c.collection_title ::TEXT,
 STRING_AGG(DISTINCT a.artist_name, ', ') AS artist_name,
 c.collection_release_date,
 COUNT(cs.song_id) AS total_tracks
 FROM collections c
 LEFT JOIN releases r ON r.collection_id = c.collection_id
 LEFT JOIN artists a ON a.artist_id = r.artist_id
 LEFT JOIN collections_songs cs ON cs.collection_id = c.collection_id
 GROUP BY
 c.collection_id,
 c.collection_title,
 c.collection_release_date
 ORDER BY c.collection_release_date DESC
 LIMIT p_limit"

Total rows: Query complete 00:00:00.266 CRLF Ln 74, Col 1

Solusi : Ubah tipe data kolom kelima pada deklarasi RETURNS TABLE menjadi BIGINT

```
CREATE OR REPLACE FUNCTION get_new_releases(p_limit INT)
RETURNS TABLE (
    collection_id INT,
    title TEXT,
    artist_name TEXT,
    release_date DATE,
    total_tracks BIGINT
)
```

3. Masalah : meng-CREATE OR REPLACE FUNCTION dengan perubahan tipe data RETURN TABLE

Namun PostgreSQL tidak mengizinkan mengganti return type function yang sudah ada.

The screenshot shows the pgAdmin interface with a query editor window. The code is the same as above, but it fails because the function already exists:

```
76   RETURNS TABLE (
77     collection_id INT,
78     title TEXT,
79     artist_name TEXT,
80     release_date DATE,
81     total_tracks BIGINT
82   )
83 < $1
84 < BEGIN
85   RETURN QUERY
86   SELECT
87     c.collection_id,
88     c.collection_title ::TEXT,  
     STRING_AGG(DISTINCT a.artist_name, ', ') AS artist_name,  
     c.collection_release_date,  
     COUNT(cs.song_id) AS total_tracks  
   FROM collections c
```

An error message is displayed below the code:

ERROR: cannot change return type of existing function
Row type defined by OUT parameters is different.
Hint: Use DROP FUNCTION get_new_releases(Integer) first.

SQL state: 42P13
Detail: Row type defined by OUT parameters is different.
Hint: Use DROP FUNCTION get_new_releases(Integer) first.

Total rows: Query complete 00:00:00.100 CRLF Ln 74, Col 1

Solusi : Menambahkan DROP FUNCTION/PROCEDURE IF EXISTS lalu CREATE ulang.

```
object <| search <| Procedures <| Sequences(1) <| seq_add_songs.playlists <| seq_artists_id <| seq_collections_id <| seq_genres_id <| seq_lists_id <| seq_playlists_id <| seq_reviews_id <| seq_socials_id <| seq_songs_id <| seq_tours_id <| seq_users_id <| Tables(27) <| add_songs.playlists <| artist_promotion <| artists <| artists.tours <| block_users <| blocklist_artists <| collection_library <| collection_top_3.genres <| collections <| collections.songs <| create_songs <| follow_artists <| follow_users <| genres

Final Destination:postgres@PostgreSQL 17
Query History
Query
Scratch Pad

$ LANGUAGE plpgsql STABLE;
SELECT * FROM get_collection_tracks(1);

DROP FUNCTION IF EXISTS get_new_releases
CREATE OR REPLACE FUNCTION get_new_releases(p_limit INT)
RETURNS TABLE (
    collection_id INT,
    title TEXT,
    artist_name TEXT,
    release_date DATE,
    total_tracks BIGINT
)
AS $$
BEGIN
    RETURN QUERY
    SELECT
Data Output
Messages
Notifications
DROP FUNCTION
Query returned successfully in 122 msec.

Total rows: 1 Query complete 00:00:00.122
```

4. Masalah: Circular dependency di Artis-Promosi-Collection, Karena promosi promosi butuh artis untuk ada, tetapi artist itu dependent ke promosi. Kesalahan pemilihan dominant entity dalam relationship.

Solusi: Memilih dominant dalam hubungan 1-to-1 dengan tepat. Berarti, disini Artist yang dominan, karena artis dapat berdiri sendiri, sedangkan promosi butuh artis untuk ada.