

Manual de Consultas Complejas SQL - Proyecto Vibesia

Proyecto: Vibesia

Versión: 1.0

Fecha: 10/06/2025

Equipo: Ad Astra

Elaborado por:

- Oscar Alejandro Prasca Chacón
- Carlos Julio Vergel Wilches
- Karen Silvana Duque Leal
- Duvan Arley Ramírez Duran
- Juan David Jaimes Rojas

Manual de Consultas Complejas SQL - Proyecto Vibesia.....	1
Introducción.....	2
Consulta 1	3
Consulta 2	4
Consulta 3	5
Consulta 4	6
Consulta 5	7
Consulta 6	8
Consulta 7	9
Consulta 8	10
Consulta 9	11
Consulta 10.....	12

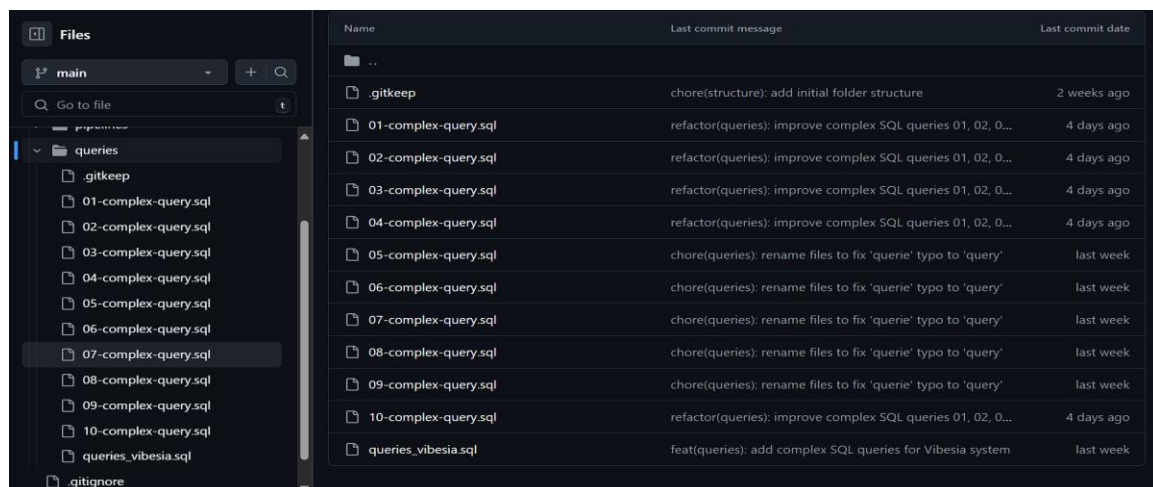
Introducción

En este documento se presentan diez consultas complejas desarrolladas sobre la base de datos musicdb, correspondiente al proyecto final del curso de Bases de Datos. Cada consulta fue diseñada para resolver necesidades avanzadas de análisis y operación dentro del sistema Vibesia, utilizando técnicas como subconsultas correlacionadas, funciones de ventana, CTEs (Common Table Expressions), agregaciones múltiples y filtros condicionales.

El propósito de este manual es documentar cada consulta con su propósito funcional, técnicas empleadas y contexto de aplicación. Este material es útil tanto para evaluadores como para desarrolladores interesados en comprender y reutilizar estas consultas como parte de futuras mejoras o mantenimientos del sistema.

Se recomienda haber seguido correctamente el proceso de instalación y carga de datos antes de ejecutar estas consultas. Para mayor claridad, cada consulta está referenciada por su archivo correspondiente dentro del repositorio.

QUERIES:

The image shows a file explorer on the left and a commit history table on the right. The file explorer shows a directory structure with a 'queries' folder containing files from 01 to 10, and a 'queries_vibesia.sql' file. The commit history table lists the following commits:

Name	Last commit message	Last commit date
..		
.gitkeep	chore(structure): add initial folder structure	2 weeks ago
01-complex-query.sql	refactor(queries): improve complex SQL queries 01, 02, 0...	4 days ago
02-complex-query.sql	refactor(queries): improve complex SQL queries 01, 02, 0...	4 days ago
03-complex-query.sql	refactor(queries): improve complex SQL queries 01, 02, 0...	4 days ago
04-complex-query.sql	refactor(queries): improve complex SQL queries 01, 02, 0...	4 days ago
05-complex-query.sql	chore(queries): rename files to fix 'querie' typo to 'query'	last week
06-complex-query.sql	chore(queries): rename files to fix 'querie' typo to 'query'	last week
07-complex-query.sql	chore(queries): rename files to fix 'querie' typo to 'query'	last week
08-complex-query.sql	chore(queries): rename files to fix 'querie' typo to 'query'	last week
09-complex-query.sql	chore(queries): rename files to fix 'querie' typo to 'query'	last week
10-complex-query.sql	refactor(queries): improve complex SQL queries 01, 02, 0...	4 days ago
queries_vibesia.sql	feat(queries): add complex SQL queries for Vibesia system	last week

TECNICAS UTILIZADAS:

- Subconsultas correlacionadas
- Funciones de ventana
- CTEs (WITH)
- Agregaciones avanzadas
- Operaciones temporales
- Filtrado condicional y clasificación

Consulta 1

Archivo: 01-complex-query.sql

Esta consulta analiza cómo cambian las tendencias musicales a lo largo de las estaciones del año. Se enfoca en géneros, artistas y álbumes, midiendo popularidad (reproducciones, oyentes únicos), calificaciones y crecimiento año tras año. Utiliza agregaciones, funciones de ventana y comparaciones temporales para mostrar los ítems más populares por estación y su evolución.

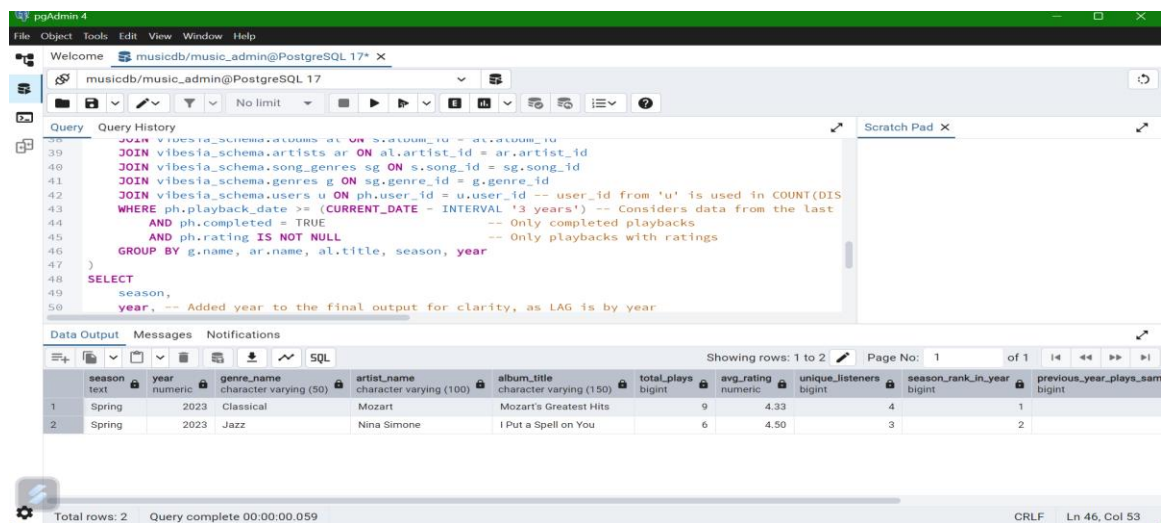
Debido a los datos de la base de datos “musicdb” ahora mira esta línea en el 01-complex-query.sql dentro del CTE seasonal_stats:

```
WHERE ph.playback_date >= (CURRENT_DATE - INTERVAL '2 years')
```

Cambia el '2 years' por '3 years'

```
WHERE ph.playback_date >= (CURRENT_DATE - INTERVAL '3 years')
```

Resultado:



The screenshot shows the pgAdmin 4 interface. The top pane displays a SQL query with several JOINs and a WHERE clause filtering by playback_date. The bottom pane shows the query results in a table format.

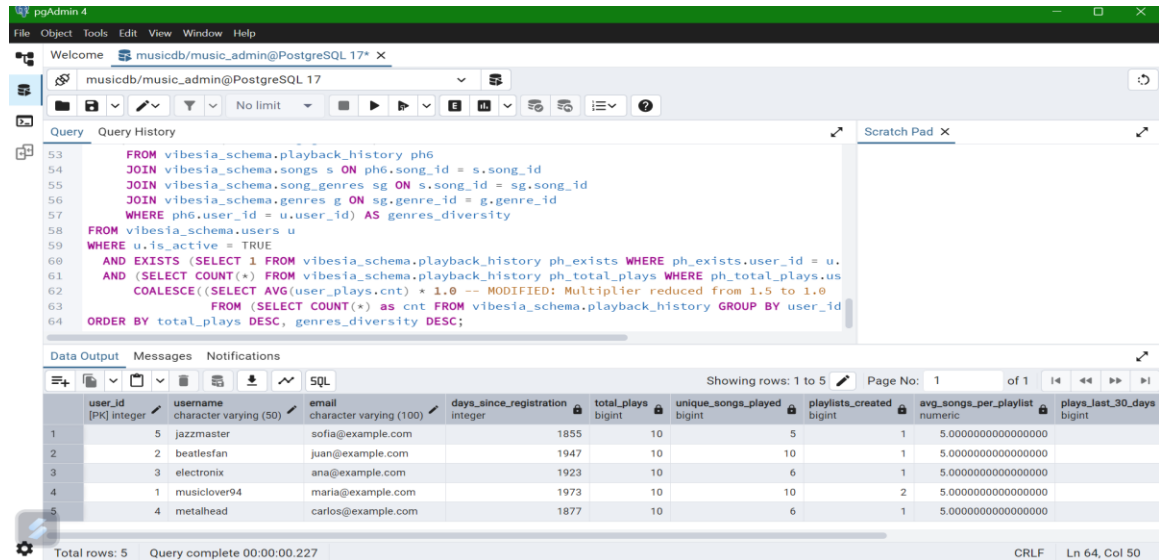
season	year	genre_name	artist_name	album_title	total_plays	avg_rating	unique_listeners	season_rank_in_year	previous_year_plays_same
Spring	2023	Classical	Mozart	Mozart's Greatest Hits	9	4.33	4	1	
Spring	2023	Jazz	Nina Simone	I Put a Spell on You	6	4.50	3	2	

Consulta 2

Archivo: 02-complex-query.sql

Esta consulta identifica a los usuarios más activos e influyentes de la plataforma según su historial de reproducción, creación de playlists, diversidad de escucha y actividad reciente. Clasifica a los usuarios que más aportan al ecosistema musical y analiza su nivel de compromiso.

Resultados:



The screenshot shows the pgAdmin 4 interface with a SQL query executed. The query is a complex JOIN and aggregation query. The results are displayed in a table with 5 rows and 10 columns.

```
53 FROM vibesia_schema.playback_history ph6
54 JOIN vibesia_schema.songs s ON ph6.song_id = s.song_id
55 JOIN vibesia_schema.song_genres sg ON s.song_id = sg.song_id
56 JOIN vibesia_schema.genres g ON sg.genre_id = g.genre_id
57 WHERE ph6.user_id = u.user_id) AS genres_diversity
58 FROM vibesia_schema.users u
59 WHERE u.is_active = TRUE
60 AND EXISTS (SELECT 1 FROM vibesia_schema.playback_history ph_exists WHERE ph_exists.user_id = u.
61 AND (SELECT COUNT(*) FROM vibesia_schema.playback_history ph_total_plays WHERE ph_total_plays.us
62 COALESCE((SELECT AVG(user_plays.cnt) * 1.0 -- MODIFIED: Multiplier reduced from 1.5 to 1.0
63 FROM (SELECT COUNT(*) as cnt FROM vibesia_schema.playback_history GROUP BY user_id
64 ORDER BY total_plays DESC, genres_diversity DESC;
```

	user_id [PK] integer	username character varying (50)	email character varying (100)	days_since_registration integer	total_plays bigint	unique_songs_played bigint	playlists_created bigint	avg_songs_per_playlist numeric	plays_last_30_days bigint
1	5	jazzmaster	sofia@example.com	1855	10	5	1	5.0000000000000000	
2	2	beatlesfan	juan@example.com	1947	10	10	1	5.0000000000000000	
3	3	electronix	ana@example.com	1923	10	6	1	5.0000000000000000	
4	1	musiclover94	maria@example.com	1973	10	10	2	5.0000000000000000	
5	4	metalhead	carlos@example.com	1877	10	6	1	5.0000000000000000	

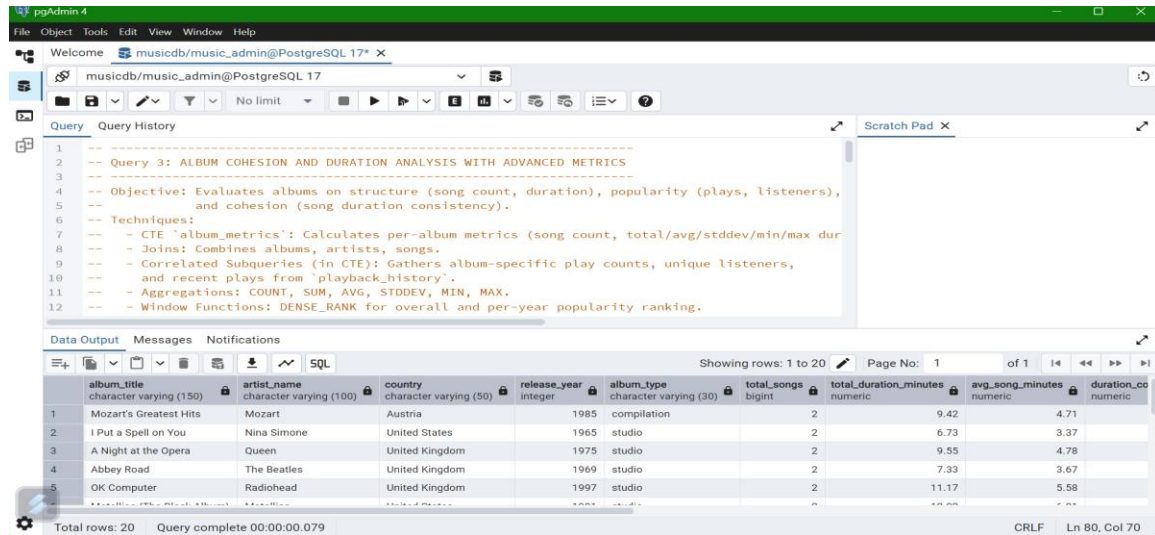
Total rows: 5 Query complete 00:00:00.227 CRLF Ln 64, Col 50

Consulta 3

Archivo: 03-complex-query.sql

Esta consulta evalúa los álbumes según su estructura (cantidad de canciones y duración), popularidad (reproducciones y oyentes) y cohesión (consistencia en duración de canciones). Incluye métricas estadísticas como media, desviación estándar, y clasifica los álbumes según su calidad estructural y popularidad general.

Resultado:



The screenshot shows the pgAdmin 4 interface. The top pane displays a SQL query titled 'Query 3: ALBUM COHESION AND DURATION ANALYSIS WITH ADVANCED METRICS'. The query is a complex SQL statement using CTEs, joins, and window functions to calculate album metrics. The bottom pane shows the results of the query in a table format.

album_title	artist_name	country	release_year	album_type	total_songs	total_duration_minutes	avg_song_minutes	duration_cs
Mozart's Greatest Hits	Mozart	Austria	1985	compilation	2	9.42	4.71	
I Put a Spell on You	Nina Simone	United States	1965	studio	2	6.73	3.37	
A Night at the Opera	Queen	United Kingdom	1975	studio	2	9.55	4.78	
Abbey Road	The Beatles	United Kingdom	1969	studio	2	7.33	3.67	
OK Computer	Radiohead	United Kingdom	1997	studio	2	11.17	5.58	

Total rows: 20 Query complete 00:00:00.079 CRLF Ln 80, Col 70

Consulta 4

Archivo: 04-complex-query.sql

Esta consulta analiza cómo varía el comportamiento de escucha según el tipo de dispositivo, sistema operativo, hora del día y día de la semana. Muestra patrones de consumo según tecnología usada y franja horaria.

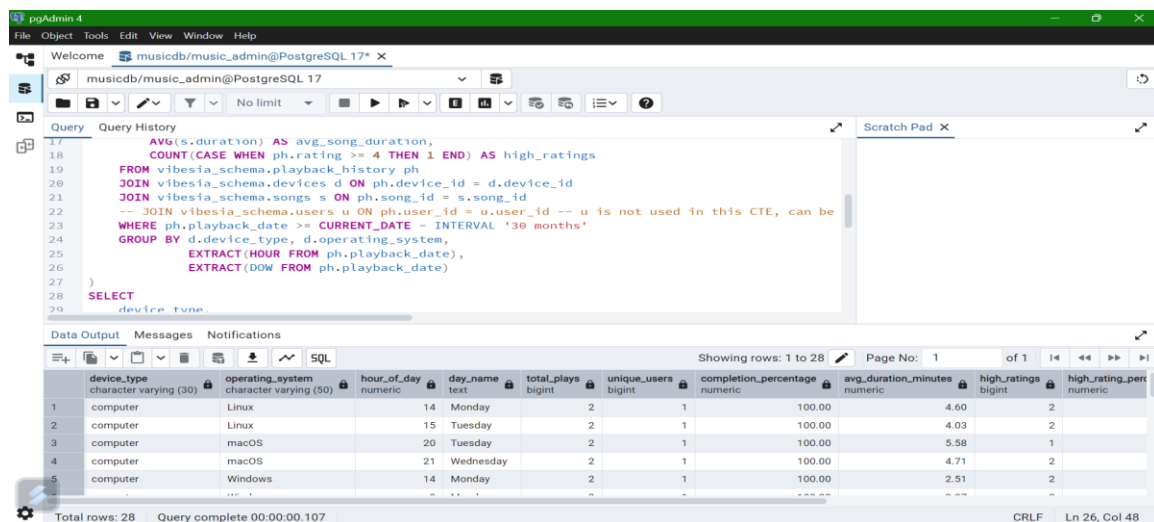
Debido a los datos de la base de datos “musicdb” ahora mira esta línea en el 04-complex-query.sql:

```
WHERE ph.playback_date >= CURRENT_DATE - INTERVAL '6 months'
```

Cambia el ‘6 months’ por ‘30 months’

```
WHERE ph.playback_date >= (CURRENT_DATE - INTERVAL '30 months')
```

Resultado:



The screenshot shows the pgAdmin 4 interface. The top pane displays a SQL query that calculates average song duration and high ratings, grouped by device type, operating system, hour of day, and day name. The query filters for playback dates within the last 30 months. The bottom pane shows the results of the query, which are 28 rows. The first five rows are visible in the table below.

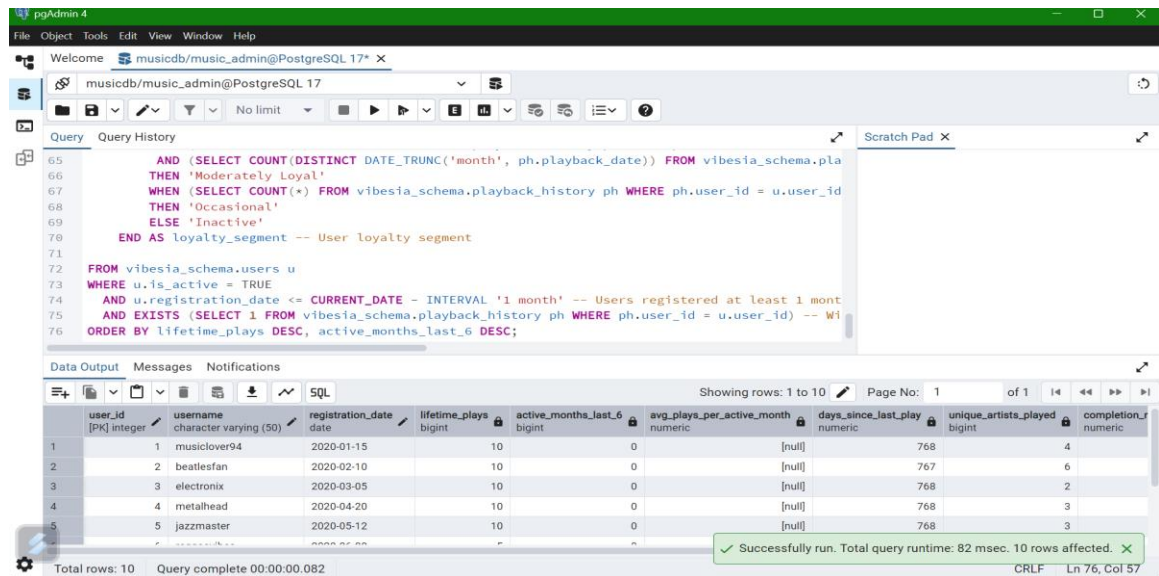
	device_type	operating_system	hour_of_day	day_name	total_plays	unique_users	completion_percentage	avg_duration_minutes	high_ratings	high_rating_per
1	computer	Linux	14	Monday	2	1	100.00	4.60	2	
2	computer	Linux	15	Tuesday	2	1	100.00	4.03	2	
3	computer	macOS	20	Tuesday	2	1	100.00	5.58	1	
4	computer	macOS	21	Wednesday	2	1	100.00	4.71	2	
5	computer	Windows	14	Monday	2	1	100.00	2.51	2	

Consulta 5

Archivo: 05-complex-query.sql

Esta consulta evalúa la lealtad de los usuarios considerando su actividad histórica, consistencia en la escucha, diversidad de artistas, y participación reciente en playlists. Asigna a cada usuario un segmento de lealtad basado en su comportamiento a largo plazo.

Resultado:



The screenshot shows the pgAdmin 4 interface with a SQL query executed. The query is a complex CASE statement that categorizes users into loyalty segments based on their playback history. The results are displayed in a table with 10 rows.

```
65 AND (SELECT COUNT(DISTINCT DATE_TRUNC('month', ph.playback_date)) FROM vibesia_schema.pla
66 THEN 'Moderately Loyal'
67 WHEN (SELECT COUNT(*) FROM vibesia_schema.playback_history ph WHERE ph.user_id = u.user_id
68 THEN 'Occasional'
69 ELSE 'Inactive'
70 END AS loyalty_segment -- User loyalty segment
71
72 FROM vibesia_schema.users u
73 WHERE u.is_active = TRUE
74 AND u.registration_date <= CURRENT_DATE - INTERVAL '1 month' -- Users registered at least 1 mont
75 AND EXISTS (SELECT 1 FROM vibesia_schema.playback_history ph WHERE ph.user_id = u.user_id) -- Wi
76 ORDER BY lifetime_plays DESC, active_months_last_6 DESC;
```

	user_id [pk] integer	username character varying (50)	registration_date date	lifetime_plays bigint	active_months_last_6 bigint	avg_plays_per_active_month numeric	days_since_last_play numeric	unique_artists_played bigint	completion_f numeric
1	1	musiclover94	2020-01-15	10	0	[null]	768	4	
2	2	beatlesfan	2020-02-10	10	0	[null]	767	6	
3	3	electronix	2020-03-05	10	0	[null]	768	2	
4	4	metalhead	2020-04-20	10	0	[null]	768	3	
5	5	jazzmaster	2020-05-12	10	0	[null]	768	3	

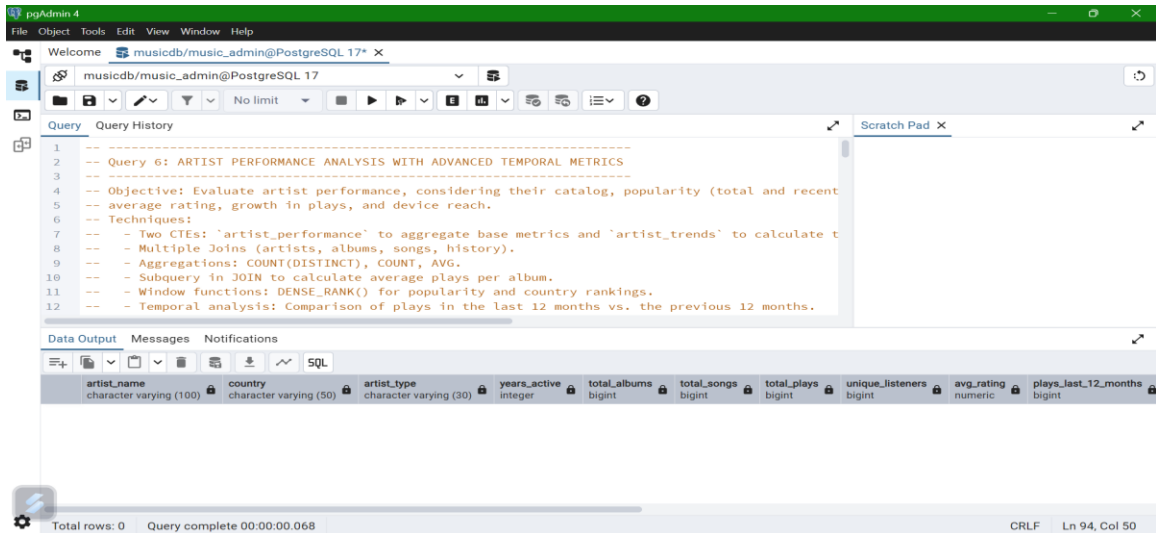
Successfully run. Total query runtime: 82 msec. 10 rows affected.

Consulta 6

Archivo: 06-complex-query.sql

Esta consulta mide el rendimiento de los artistas en cuanto a su catálogo musical, popularidad total y reciente, calificación promedio, crecimiento en reproducciones y alcance por dispositivos. Incluye comparaciones año a año y clasificaciones por popularidad y país.

Resultado:



The screenshot shows the pgAdmin 4 interface with a query editor and a data output pane. The query editor contains a complex SQL query with comments explaining its purpose and techniques. The data output pane shows the schema of the query results, including columns for artist information, performance metrics, and temporal analysis.

```
1
2 -- Query 6: ARTIST PERFORMANCE ANALYSIS WITH ADVANCED TEMPORAL METRICS
3
4 -- Objective: Evaluate artist performance, considering their catalog, popularity (total and recent
5 -- average rating, growth in plays, and device reach.
6 -- Techniques:
7 --   - Two CTEs: 'artist_performance' to aggregate base metrics and 'artist_trends' to calculate t
8 --   - Multiple Joins (artists, albums, songs, history).
9 --   - Aggregations: COUNT(DISTINCT), COUNT, AVG.
10 --   - Subquery in JOIN to calculate average plays per album.
11 --   - Window functions: DENSE_RANK() for popularity and country rankings.
12 --   - Temporal analysis: Comparison of plays in the last 12 months vs. the previous 12 months.
```

artist_name	country	artist_type	years_active	total_albums	total_songs	total_plays	unique_listeners	avg_rating	plays_last_12_months
character varying (100)	character varying (50)	character varying (30)	integer	bigint	bigint	bigint	bigint	numeric	bigint

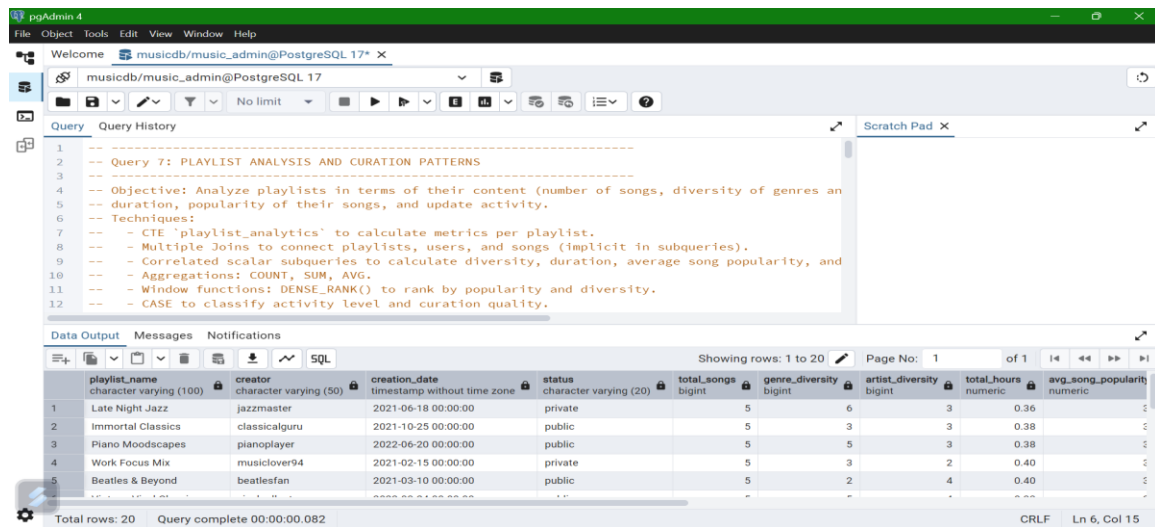
Total rows: 0 Query complete 00:00:00.068 CRLF Ln 94, Col 50

Consulta 7

Archivo: 07-complex-query.sql

Esta consulta estudia las playlists según su contenido (cantidad y diversidad de canciones, artistas y géneros), duración total, popularidad promedio de las canciones, y frecuencia de actualizaciones. Asigna calificaciones de calidad de curación y actividad de los usuarios.

Resultado:



The screenshot shows the pgAdmin 4 interface. The top pane displays a SQL query titled 'Query 7: PLAYLIST ANALYSIS AND CURATION PATTERNS'. The query is a complex SQL statement using CTEs, joins, and window functions to analyze playlist data. The bottom pane shows the results of the query in a table format.

Query 7: PLAYLIST ANALYSIS AND CURATION PATTERNS

```
-- Objective: Analyze playlists in terms of their content (number of songs, diversity of genres and
-- duration, popularity of their songs, and update activity.
-- Techniques:
-- - CTE 'playlist_analytics' to calculate metrics per playlist.
-- - Multiple Joins to connect playlists, users, and songs (implicit in subqueries).
-- - Correlated scalar subqueries to calculate diversity, duration, average song popularity, and
-- - Aggregations: COUNT, SUM, AVG.
-- - Window functions: DENSE_RANK() to rank by popularity and diversity.
-- - CASE to classify activity level and curation quality.
```

Data Output

playlist_name	creator	creation_date	status	total_songs	genre_diversity	artist_diversity	total_hours	avg_song_popularity
Late Night Jazz	jazzmaster	2021-06-18 00:00:00	private	5	6	3	0.36	
Immortal Classics	classicalgunu	2021-10-25 00:00:00	public	5	3	3	0.38	
Piano Moods	pianoplayer	2022-06-20 00:00:00	public	5	5	3	0.38	
Work Focus Mix	musiclover94	2021-02-15 00:00:00	private	5	3	2	0.40	
Beatles & Beyond	beatlesfan	2021-03-10 00:00:00	public	5	2	4	0.40	

Total rows: 20 Query complete 00:00:00.082 CRLF Ln 6, Col 15

Consulta 8

Archivo: 08-complex-query.sql

Esta consulta explora cómo el contexto de la reproducción (hora, día, tipo de dispositivo, duración de la canción, familiaridad con el artista, etc.) influye en las calificaciones dadas por los usuarios. Permite detectar patrones en la forma en que se valoran las canciones.

Resultado:

The screenshot shows the pgAdmin 4 interface with a query editor and a data output pane. The query is a complex SQL statement designed to analyze rating correlations with contextual factors.

```
1
2 -- Query 8: RATING CORRELATION ANALYSIS WITH CONTEXTUAL FACTORS
3
4 -- Objective: Explore how different contextual factors (time, day, device type,
5 -- song duration, artist familiarity, completion) correlate
6 -- with user-given ratings.
7 -- Techniques:
8 --   - CTE 'rating_context' to gather playback data with its context.
9 --   - Multiple Joins to enrich playback data.
10 --   - Temporal operations: EXTRACT for hour, day, month.
11 --   - Correlated scalar subqueries to get user context at the time of playback.
12 --   - Conditional aggregations (AVG with CASE) to analyze ratings under different conditions.
```

The Data Output pane shows the following schema:

hour_of_day	total_ratings	avg_rating_by_hour	rating_stddev	day_name	device_type	high_ratings	low_ratings	avg_rating_long_songs	avg_rating_short_songs
numeric	bigint	numeric	numeric	text	character varying (30)	bigint	bigint	numeric	numeric

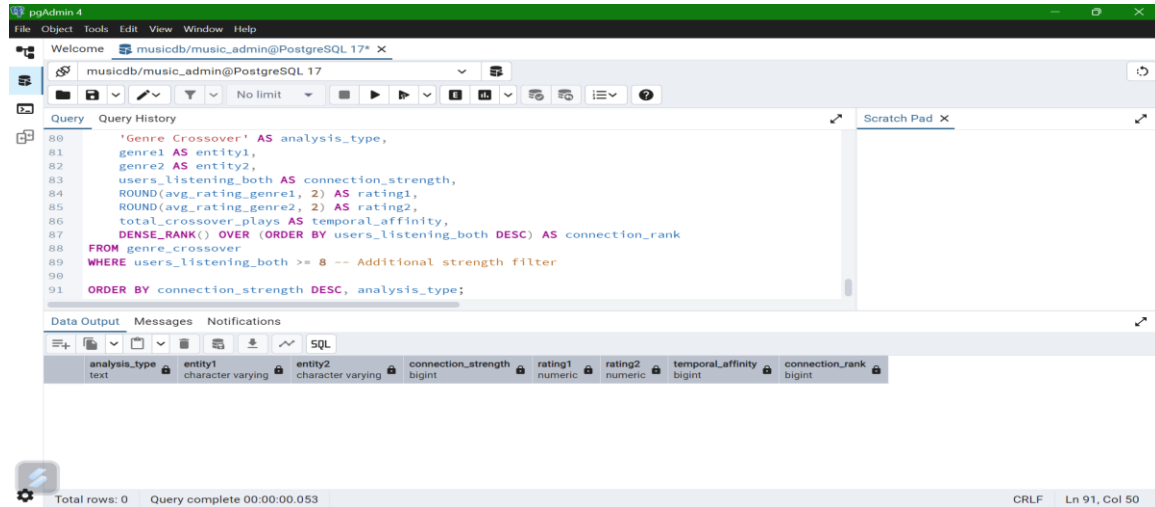
At the bottom, the status bar indicates: Total rows: 0, Query complete 00:00:00.080, CRLF, Ln 99, Col 35.

Consulta 9

Archivo: 09-complex-query.sql

Esta consulta detecta conexiones entre artistas (por oyentes en común) y entre géneros (por usuarios que escuchan ambos). Ayuda a comprender la red musical de colaboraciones y afinidades, destacando vínculos fuertes en el ecosistema musical.

Resultado:



The screenshot shows the pgAdmin 4 interface with a SQL query editor and a result table. The query is a complex SQL statement that joins the 'genre_crossover' table with itself to find connections between artists and genres based on shared listeners and ratings. The result table displays the output of the query, showing columns for analysis_type, entity1, entity2, connection_strength, rating1, rating2, temporal_affinity, and connection_rank.

```
80 'Genre Crossover' AS analysis_type,
81 genre1 AS entity1,
82 genre2 AS entity2,
83 users_listening_both AS connection_strength,
84 ROUND(avg_rating_genre1, 2) AS rating1,
85 ROUND(avg_rating_genre2, 2) AS rating2,
86 total_crossover_plays AS temporal_affinity,
87 DENSE_RANK() OVER (ORDER BY users_listening_both DESC) AS connection_rank
88 FROM genre_crossover
89 WHERE users_listening_both >= 8 -- Additional strength filter
90
91 ORDER BY connection_strength DESC, analysis_type;
```

analysis_type	entity1	entity2	connection_strength	rating1	rating2	temporal_affinity	connection_rank
---------------	---------	---------	---------------------	---------	---------	-------------------	-----------------

Total rows: 0 Query complete 00:00:00.053 CRLF Ln 91, Col 50

Consulta 10

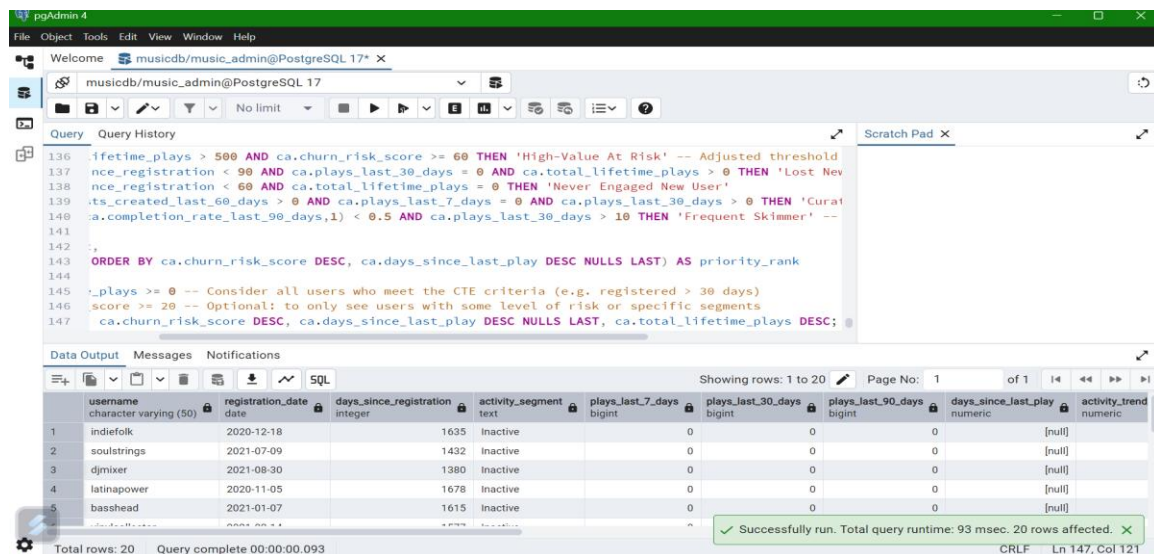
Archivo: 10-complex-query.sql

Esta consulta identifica usuarios con riesgo de abandonar la plataforma, basándose en su comportamiento reciente, tendencias de actividad y nivel de compromiso. Asigna puntuaciones de riesgo y segmenta usuarios para aplicar estrategias de retención personalizadas.

Técnicas utilizadas:

- Subconsultas correlacionadas
- Funciones de ventana
- CTEs (WITH)
- Agregaciones avanzadas
- Operaciones temporales
- Filtrado condicional y clasificación

Resultados:



The screenshot shows the pgAdmin 4 interface with a complex SQL query executed. The query uses CTEs to identify users at risk of churning based on their activity and engagement. The results table shows 20 rows of user data.

username	registration_date	days_since_registration	activity_segment	plays_last_7_days	plays_last_30_days	plays_last_90_days	days_since_last_play	activity_trend
indiefolk	2020-12-18	1635	Inactive	0	0	0	[null]	
soulstrings	2021-07-09	1432	Inactive	0	0	0	[null]	
djmixer	2021-08-30	1380	Inactive	0	0	0	[null]	
latinapower	2020-11-05	1678	Inactive	0	0	0	[null]	
basshead	2021-01-07	1615	Inactive	0	0	0	[null]	

Successfully run. Total query runtime: 93 msec. 20 rows affected.

Fin del Manual