

# 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.....4

    Consulta 4.....5

    Consulta 5.....6

    Consulta 6.....6

    Consulta 7.....7

    Consulta 8.....7

    Consulta 9.....8

    Consulta 10.....9

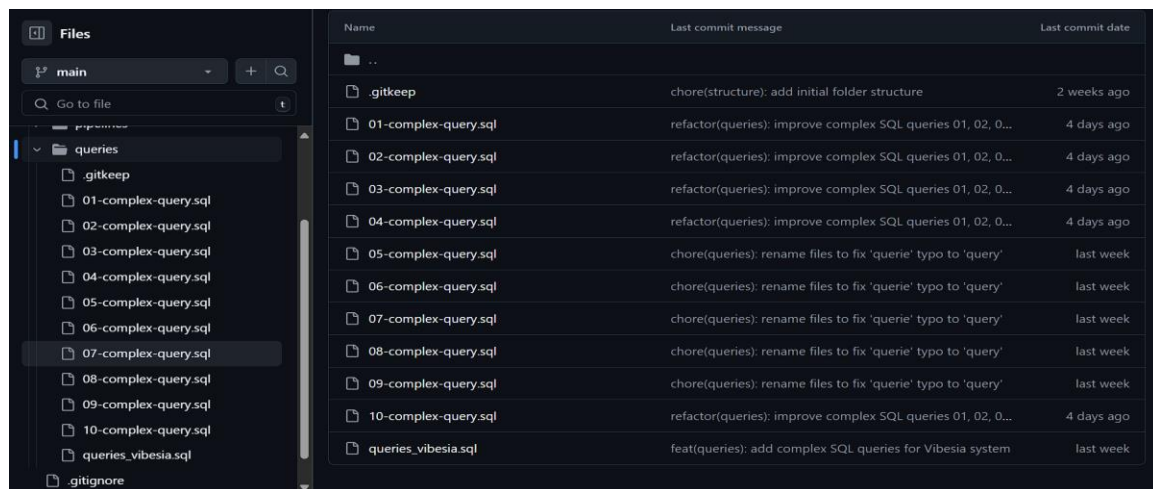
## Introducción

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-complex-query.sql to 10-complex-query.sql, and a 'queries\_vibesia.sql' file. The commit history table lists the commit messages and dates for these files.

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.

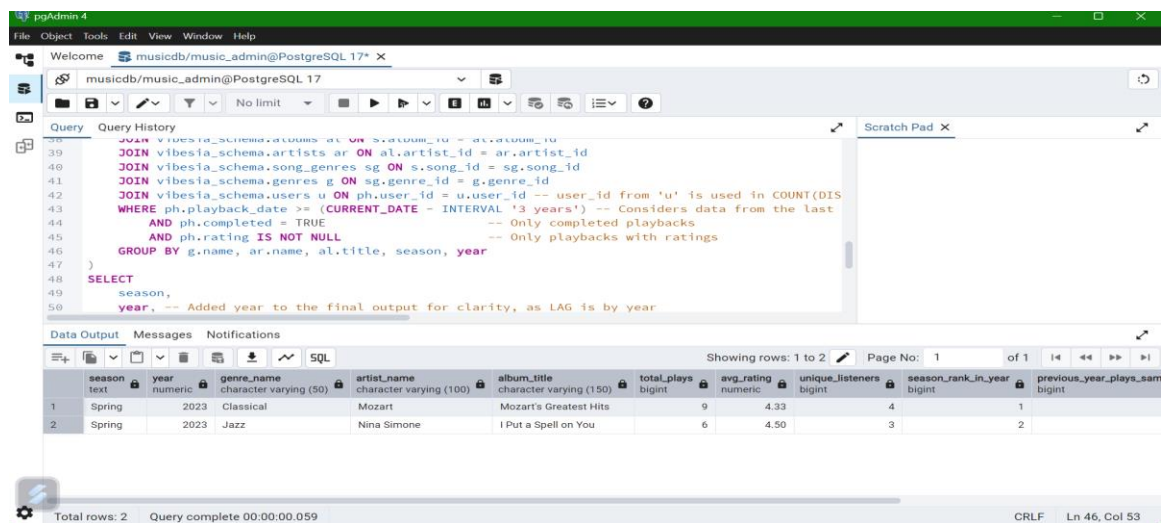
Debido a los datos de la base de datos “musicdb” ahora mira esta línea en el 01-complex-query.sql dentro del CTE seasonal\_stats:

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. The bottom pane shows the results of the query in a table format.

Query:

```
38 JOIN vibesia_schema.albums al ON s.album_id = al.album_id
39 JOIN vibesia_schema.artists ar ON al.artist_id = ar.artist_id
40 JOIN vibesia_schema.song_genres sg ON s.song_id = sg.song_id
41 JOIN vibesia_schema.genres g ON sg.genre_id = g.genre_id
42 JOIN vibesia_schema.users u ON ph.user_id = u.user_id -- user_id from 'u' is used in COUNT(DISTINCT u)
43 WHERE ph.playback_date >= (CURRENT_DATE - INTERVAL '3 years') -- Considers data from the last 3 years
44 AND ph.completed = TRUE -- Only completed playbacks
45 AND ph.rating IS NOT NULL -- Only playbacks with ratings
46 )
47 )
48 SELECT
49 season,
50 year, -- Added year to the final output for clarity, as LAG is by year
```

Data Output:

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	

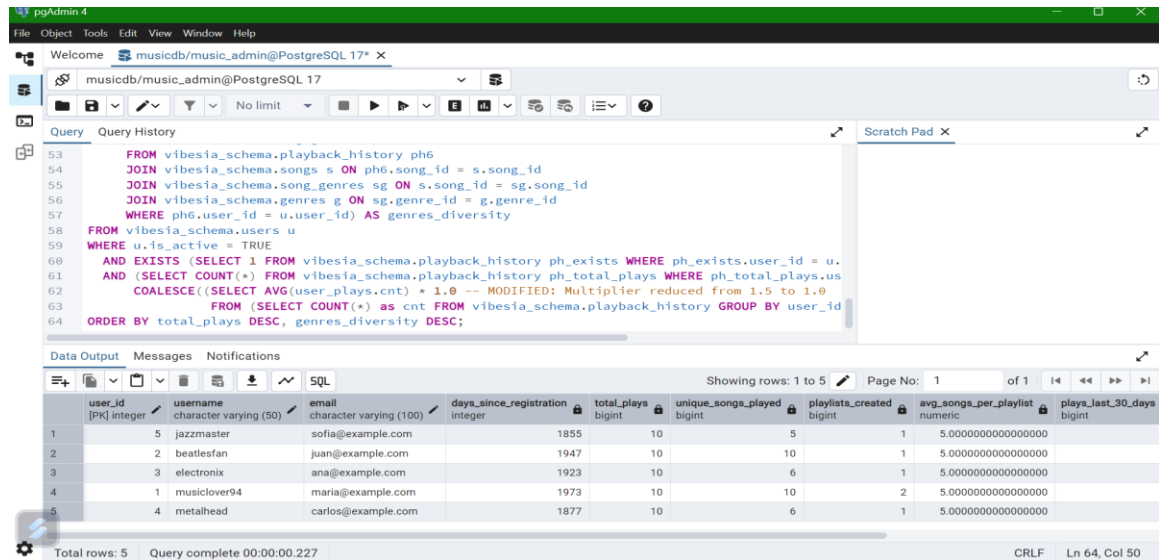
Total rows: 2 Query complete 00:00:00.059 CRLF Ln 46, Col 53

## 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.

	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	10	5	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	

## 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:

pgAdmin 4

Welcome musicdb/music\_admin@PostgreSQL 17\*

musicdb/music\_admin@PostgreSQL 17

Query Query History Scratch Pad

```

1  -- Query 3: ALBUM COHESION AND DURATION ANALYSIS WITH ADVANCED METRICS
2
3  -- Objective: Evaluates albums on structure (song count, duration), popularity (plays, listeners),
4  -- and cohesion (song duration consistency).
5
6  -- Techniques:
7  --   - CTE 'album_metrics': Calculates per-album metrics (song count, total/avg/stddev/min/max dur
8  --   - Joins: Combines albums, artists, songs.
9  --   - Correlated Subqueries (in CTE): Gathers album-specific play counts, unique listeners,
10 --   and recent plays from 'playback_history'.
11 --   - Aggregations: COUNT, SUM, AVG, STDDEV, MIN, MAX.
12 --   - Window Functions: DENSE_RANK for overall and per-year popularity ranking.

```

Data Output Messages Notifications

Showing rows: 1 to 20 Page No: 1 of 1

	album_title character varying (150)	artist_name character varying (100)	country character varying (50)	release_year integer	album_type character varying (30)	total_songs bigint	total_duration_minutes numeric	avg_song_minutes numeric	duration_co numeric
1	Mozart's Greatest Hits	Mozart	Austria	1985	compilation	2	9.42	4.71	
2	I Put a Spell on You	Nina Simone	United States	1965	studio	2	6.73	3.37	
3	A Night at the Opera	Queen	United Kingdom	1975	studio	2	9.55	4.78	
4	Abbey Road	The Beatles	United Kingdom	1969	studio	2	7.33	3.67	
5	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:

Query:

```

17      AVG(s.duration) AS avg_song_duration,
18      COUNT(CASE WHEN ph.rating >= 4 THEN 1 END) AS high_ratings
19  FROM vibesia_schema.playback_history ph
20  JOIN vibesia_schema.devices d ON ph.device_id = d.device_id
21  JOIN vibesia_schema.songs s ON ph.song_id = s.song_id
22  -- JOIN vibesia_schema.users u ON ph.user_id = u.user_id -- u is not used in this CTE, can be
23  WHERE ph.playback_date >= CURRENT_DATE - INTERVAL '30 months'
24  GROUP BY d.device_type, d.operating_system,
25           EXTRACT(HOUR FROM ph.playback_date),
26           EXTRACT(DOW FROM ph.playback_date)
27  )
28  SELECT
29  device_type

```

Data Output:

	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	2	1	100.00	4.60	2
2	computer	Linux	15	Tuesday	2	1	1	100.00	4.03	2
3	computer	macOS	20	Tuesday	2	1	1	100.00	5.58	1
4	computer	macOS	21	Wednesday	2	1	1	100.00	4.71	2
5	computer	Windows	14	Monday	2	1	1	100.00	2.51	2

Total rows: 28 Query complete 00:00:00.107

## 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:

Query:

```

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;

```

Data Output:

	user_id	username	registration_date	lifetime_plays	active_months_last_6	avg_plays_per_active_month	days_since_last_play	unique_artists_played	completion_f
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	

Total rows: 10 Query complete 00:00:00.082

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

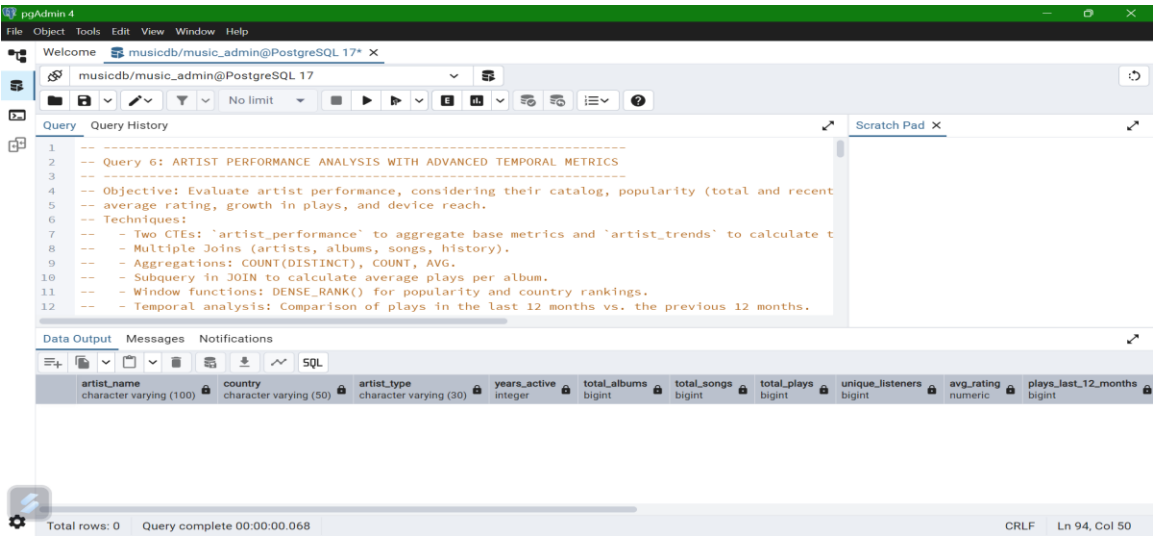
## Consulta 6

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:

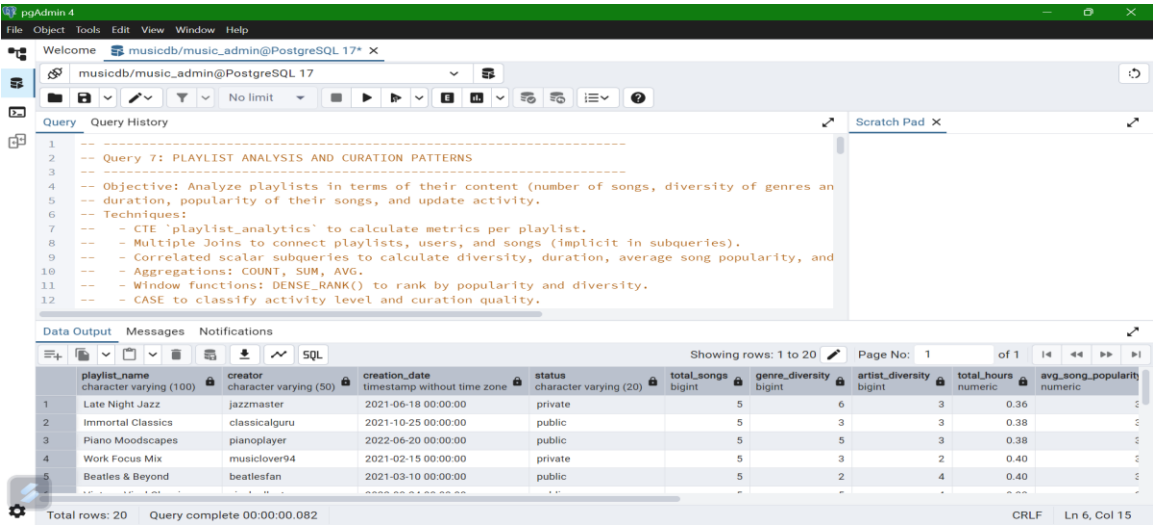


Consulta 7

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:

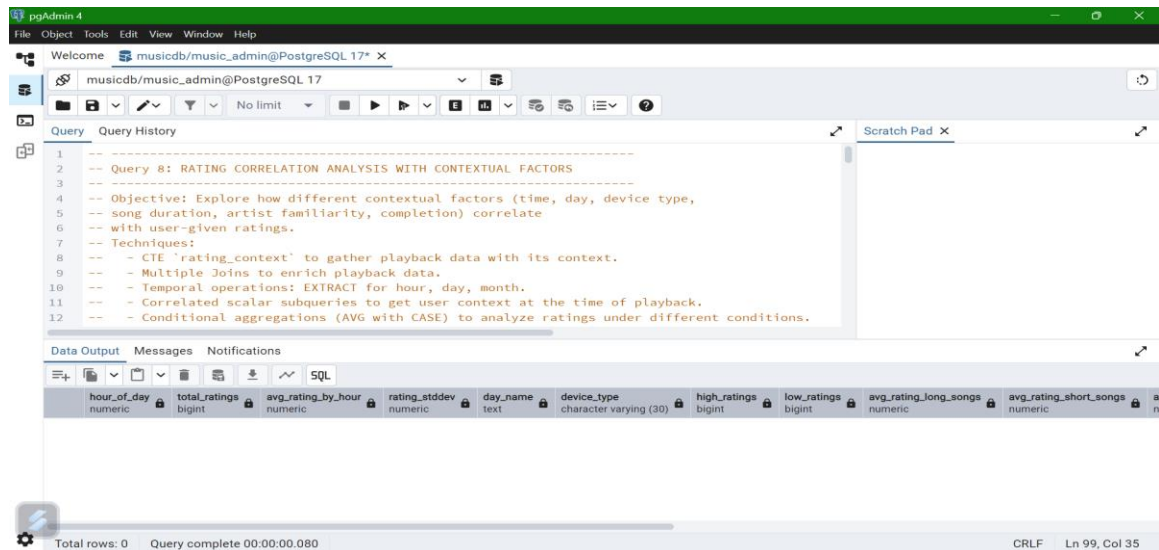


Consulta 8

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 SQL query editor. The query is titled "Query 8: RATING CORRELATION ANALYSIS WITH CONTEXTUAL FACTORS". It includes a detailed objective and techniques section. The query itself is a complex SQL statement that uses CTEs to gather playback data, enrich it with context, and then performs temporal operations and conditional aggregations to analyze ratings under different conditions. The results pane shows a table with columns: hour\_of\_day, total\_ratings, avg\_rating\_by\_hour, rating\_stddev, day\_name, device\_type, high\_ratings, low\_ratings, avg\_rating\_long\_songs, and avg\_rating\_short\_songs.

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

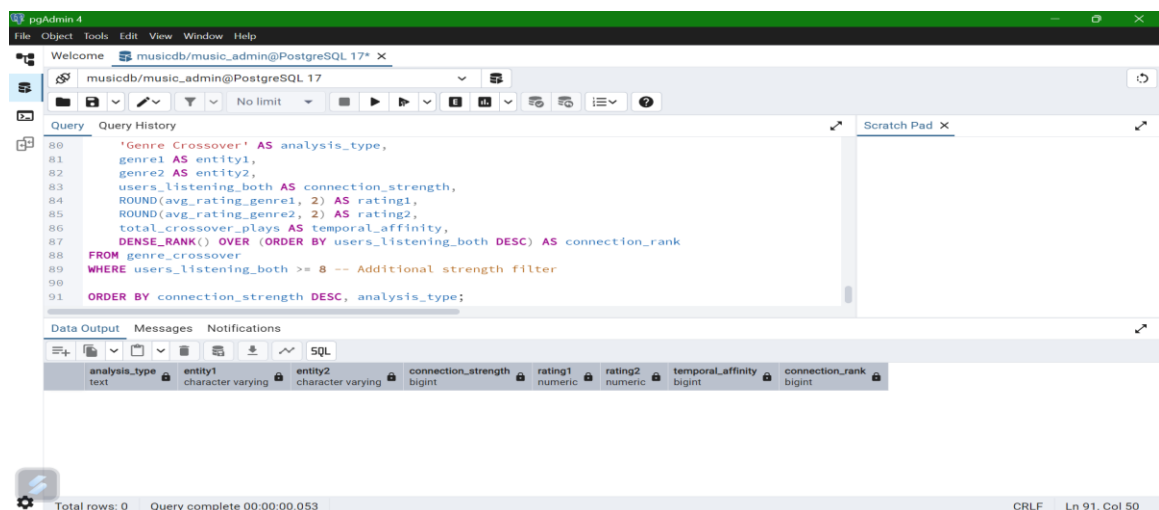
hour_of_day	total_ratings	avg_rating_by_hour	rating_stddev	day_name	device_type	high_ratings	low_ratings	avg_rating_long_songs	avg_rating_short_songs
-------------	---------------	--------------------	---------------	----------	-------------	--------------	-------------	-----------------------	------------------------

## 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. The query is titled "Genre Crossover" and is an AS analysis type. It uses a complex SQL statement to identify connections between artists and genres based on user listening data. The query includes a DENSE\_RANK() function to rank connections by strength. The results pane shows a table with columns: 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
---------------	---------	---------	---------------------	---------	---------	-------------------	-----------------



## Consulta 10

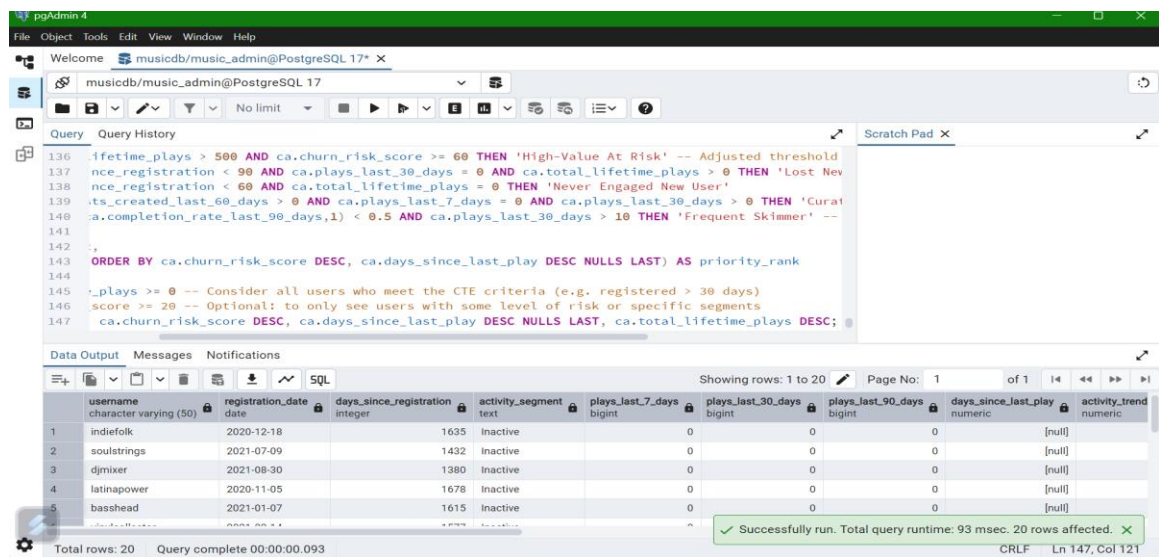
Archivo: 10-complex-query.sql

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