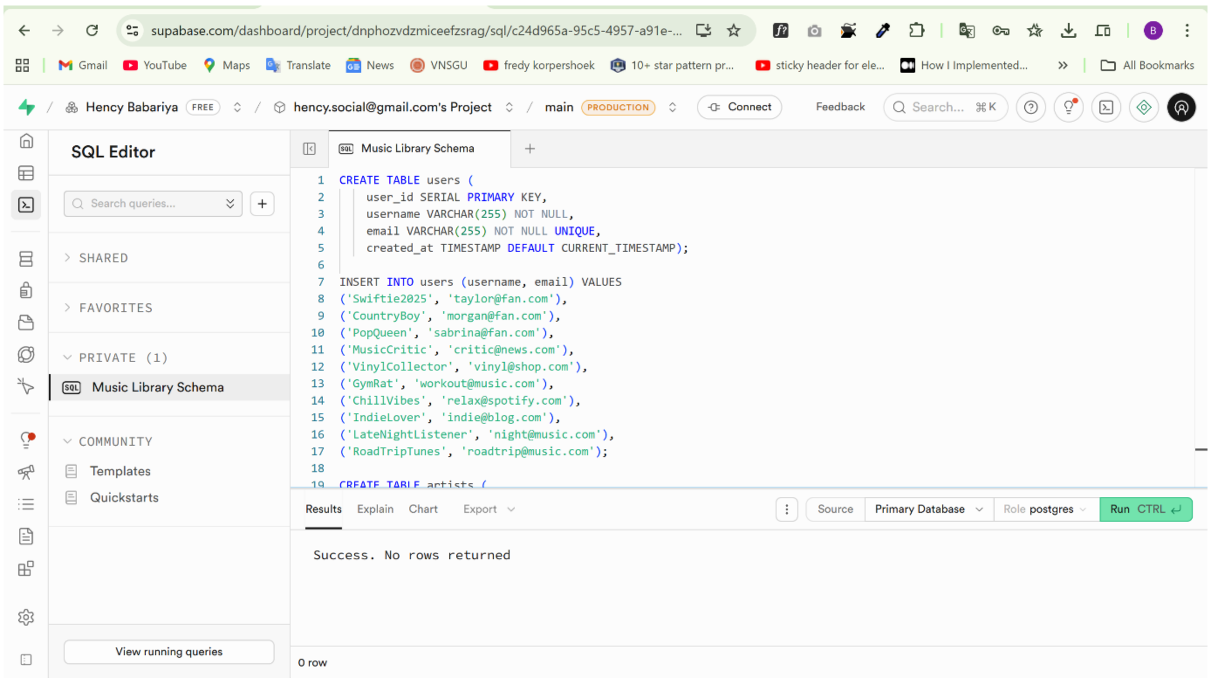
Output of Section 2: Data Implementation

1. Successful creation of table and insertion of data

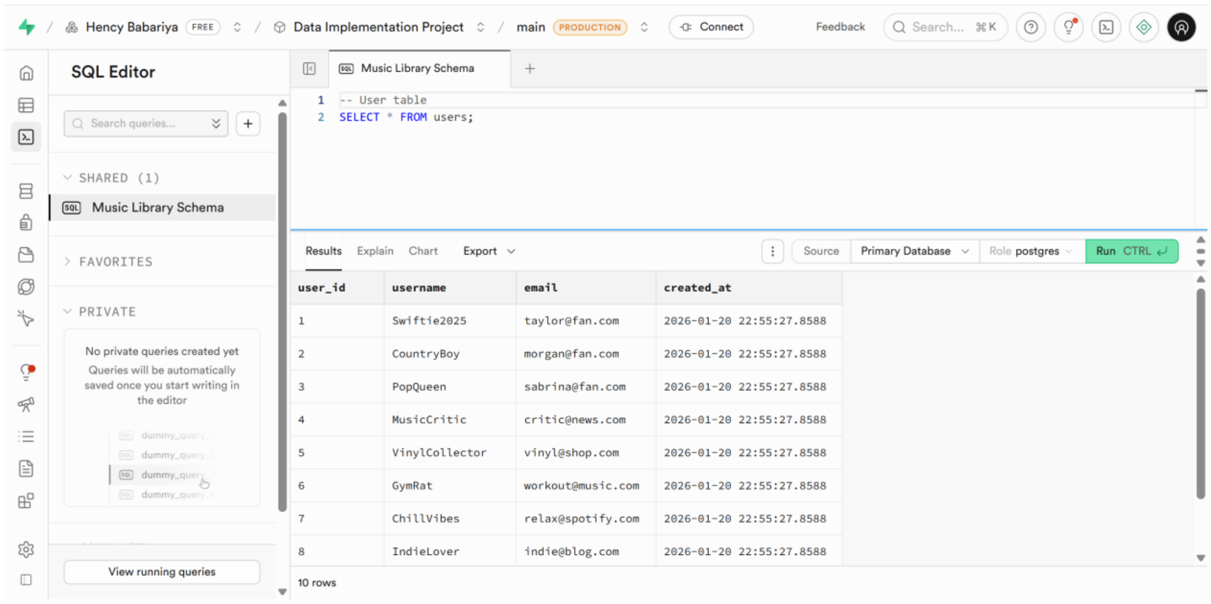


```
CREATE TABLE users (
    user_id SERIAL PRIMARY KEY,
    username VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL UNIQUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP);

INSERT INTO users (username, email) VALUES
('Swiftie2025', 'taylor@fan.com'),
('CountryBoy', 'morgan@fan.com'),
('PopQueen', 'sabrina@fan.com'),
('MusicCritic', 'critic@news.com'),
('VinylCollector', 'vinyl@shop.com'),
('GymRat', 'workout@music.com'),
('ChillVibes', 'relax@spotify.com'),
('IndieLover', 'indie@blog.com'),
('LateNightListener', 'night@music.com'),
('RoadTripTunes', 'roadtrip@music.com');

CREATE TABLE artists (
```

```
Success. No rows returned
```

0 row

2. User Table

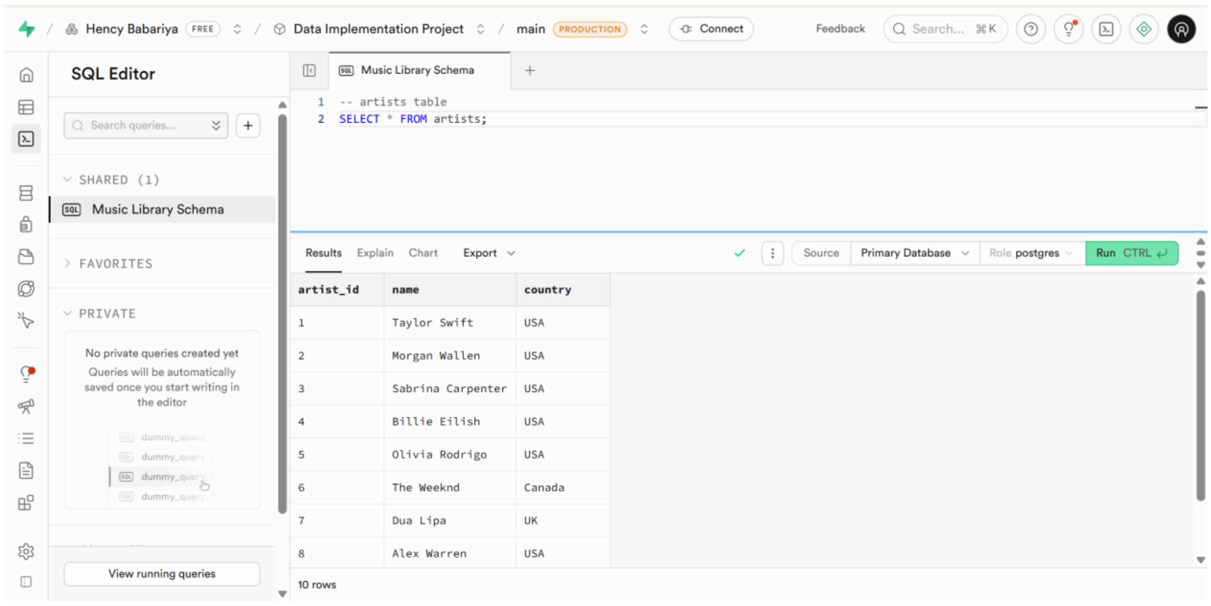


```
-- User table
SELECT * FROM users;
```

user_id	username	email	created_at
1	Swiftie2025	taylor@fan.com	2026-01-20 22:55:27.8588
2	CountryBoy	morgan@fan.com	2026-01-20 22:55:27.8588
3	PopQueen	sabrina@fan.com	2026-01-20 22:55:27.8588
4	MusicCritic	critic@news.com	2026-01-20 22:55:27.8588
5	VinylCollector	vinyl@shop.com	2026-01-20 22:55:27.8588
6	GymRat	workout@music.com	2026-01-20 22:55:27.8588
7	ChillVibes	relax@spotify.com	2026-01-20 22:55:27.8588
8	IndieLover	indie@blog.com	2026-01-20 22:55:27.8588

10 rows

3. Artists Table

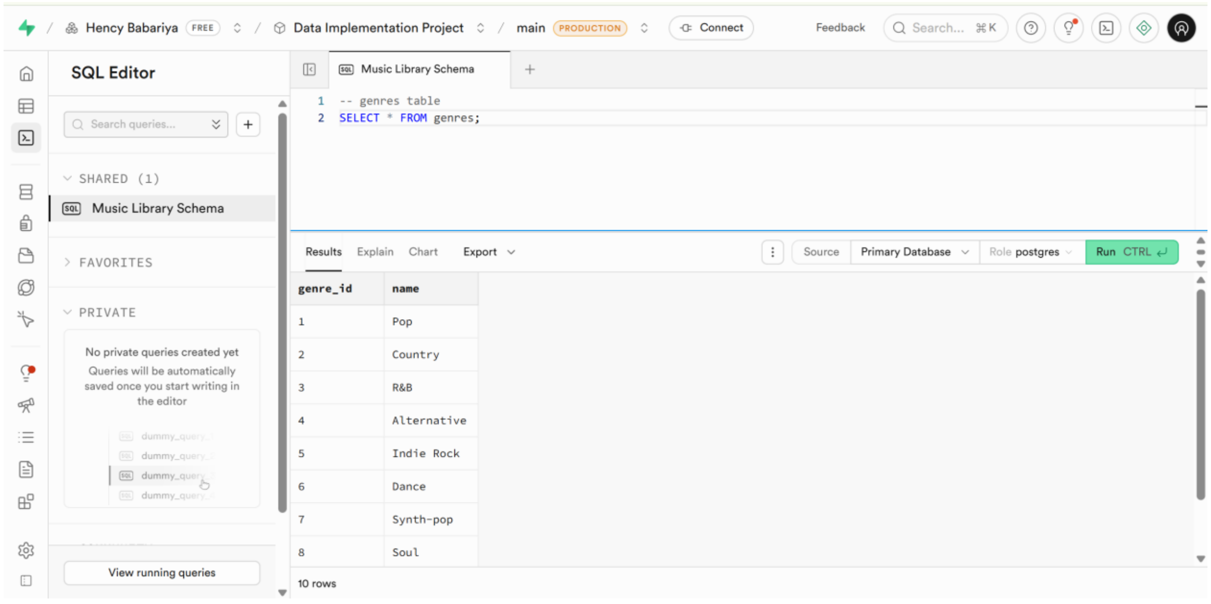


```
-- artists table
SELECT * FROM artists;
```

artist_id	name	country
1	Taylor Swift	USA
2	Morgan Wallen	USA
3	Sabrina Carpenter	USA
4	Billie Eilish	USA
5	Olivia Rodrigo	USA
6	The Weeknd	Canada
7	Dua Lipa	UK
8	Alex Warren	USA

10 rows

4. Genres Table

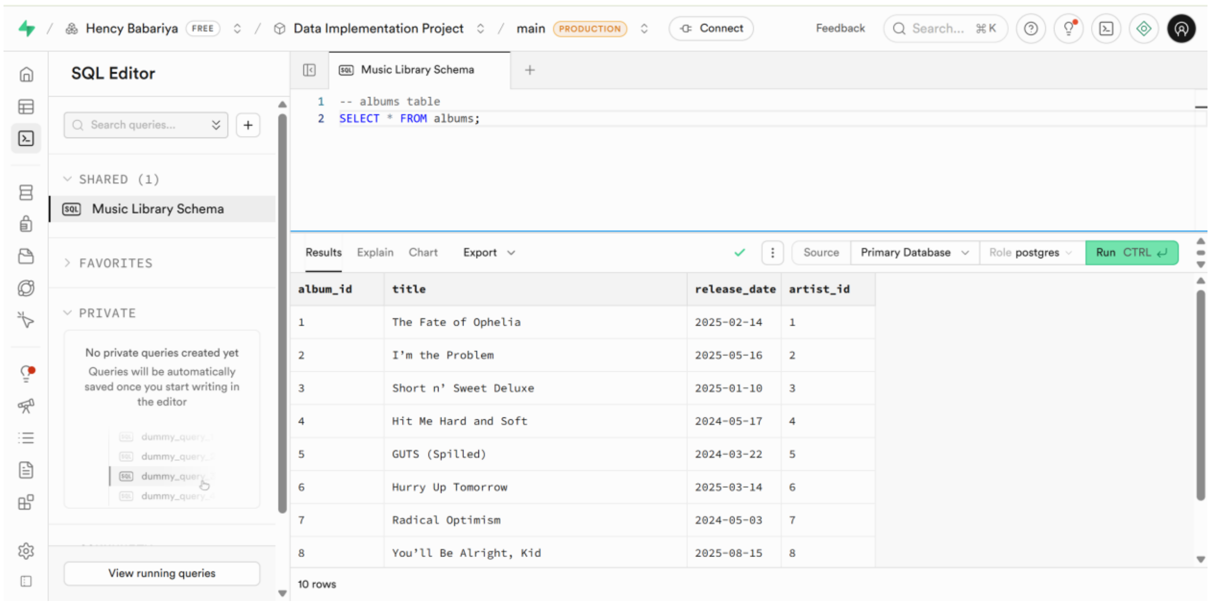


```
-- genres table
SELECT * FROM genres;
```

genre_id	name
1	Pop
2	Country
3	R&B
4	Alternative
5	Indie Rock
6	Dance
7	Synth-pop
8	Soul

10 rows

5. Albums Table

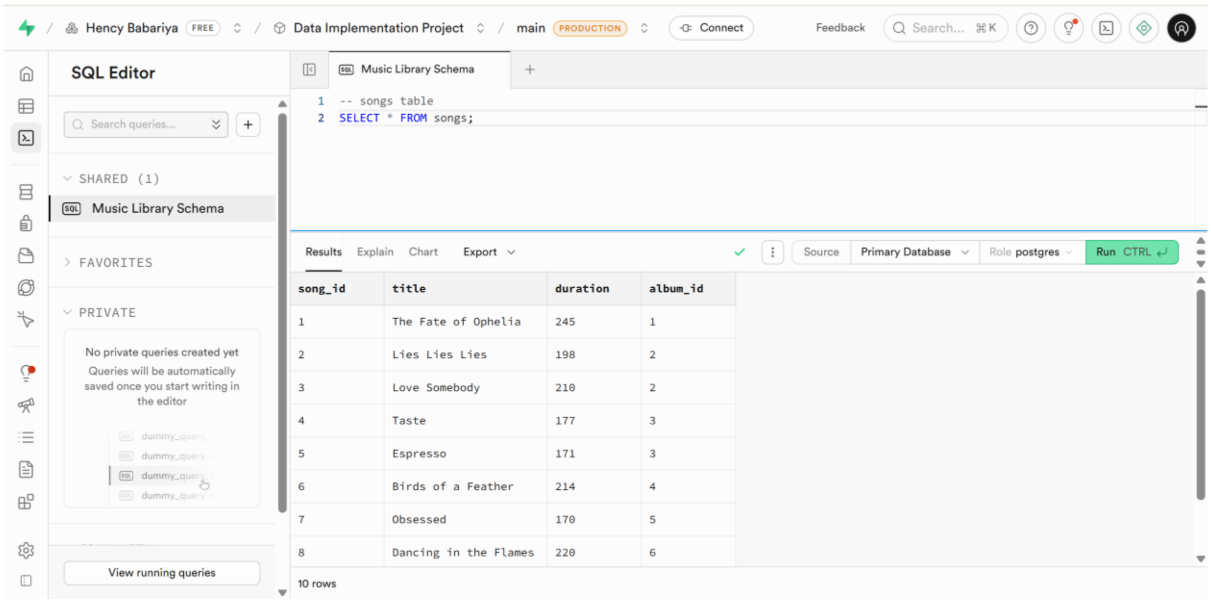


```
-- albums table
SELECT * FROM albums;
```

album_id	title	release_date	artist_id
1	The Fate of Ophelia	2025-02-14	1
2	I'm the Problem	2025-05-16	2
3	Short n' Sweet Deluxe	2025-01-10	3
4	Hit Me Hard and Soft	2024-05-17	4
5	GUTS (Spilled)	2024-03-22	5
6	Hurry Up Tomorrow	2025-03-14	6
7	Radical Optimism	2024-05-03	7
8	You'll Be Alright, Kid	2025-08-15	8

10 rows

6. Songs Table

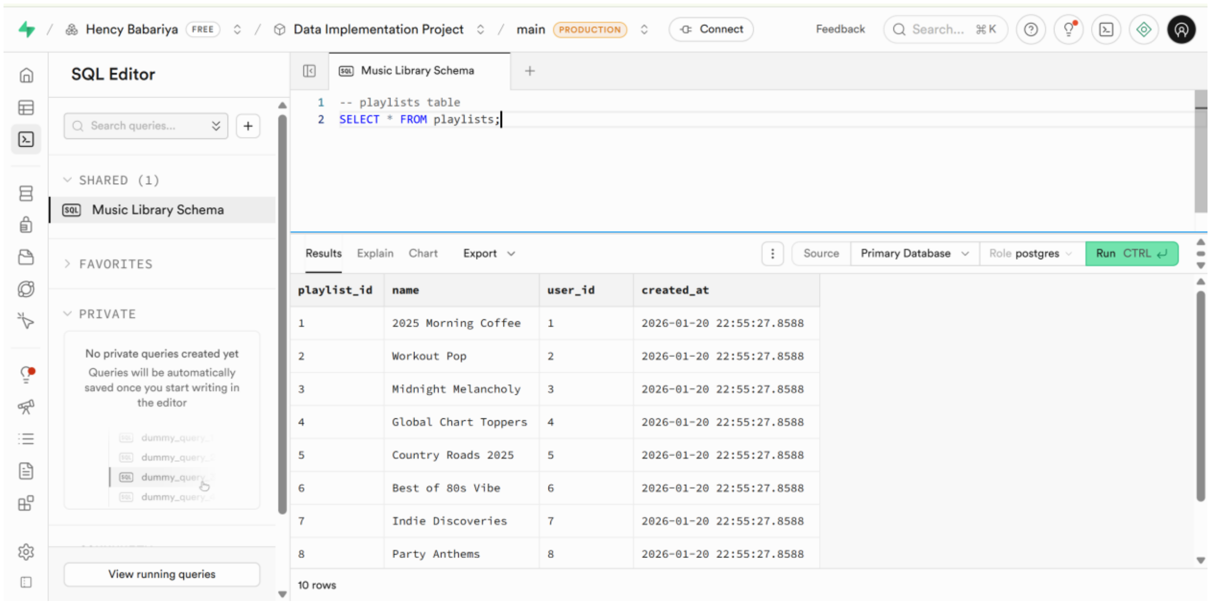


```
-- songs table
SELECT * FROM songs;
```

song_id	title	duration	album_id
1	The Fate of Ophelia	245	1
2	Lies Lies Lies	198	2
3	Love Somebody	210	2
4	Taste	177	3
5	Espresso	171	3
6	Birds of a Feather	214	4
7	Obsessed	170	5
8	Dancing in the Flames	220	6

10 rows

7. Playlists Table



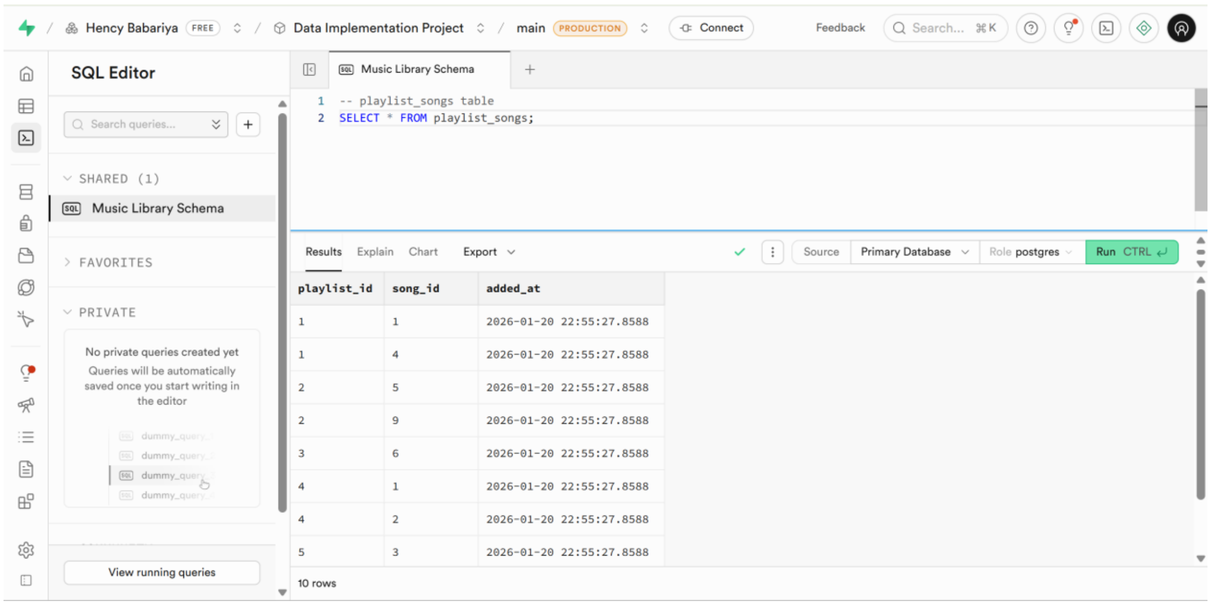
No private queries created yet Queries will be automatically saved once you start writing in the editor

```sql
-- playlists table
SELECT * FROM playlists;
```

playlist_id	name	user_id	created_at
1	2025 Morning Coffee	1	2026-01-20 22:55:27.8588
2	Workout Pop	2	2026-01-20 22:55:27.8588
3	Midnight Melancholy	3	2026-01-20 22:55:27.8588
4	Global Chart Toppers	4	2026-01-20 22:55:27.8588
5	Country Roads 2025	5	2026-01-20 22:55:27.8588
6	Best of 80s Vibe	6	2026-01-20 22:55:27.8588
7	Indie Discoveries	7	2026-01-20 22:55:27.8588
8	Party Anthems	8	2026-01-20 22:55:27.8588

10 rows

8. playlist_songs Table



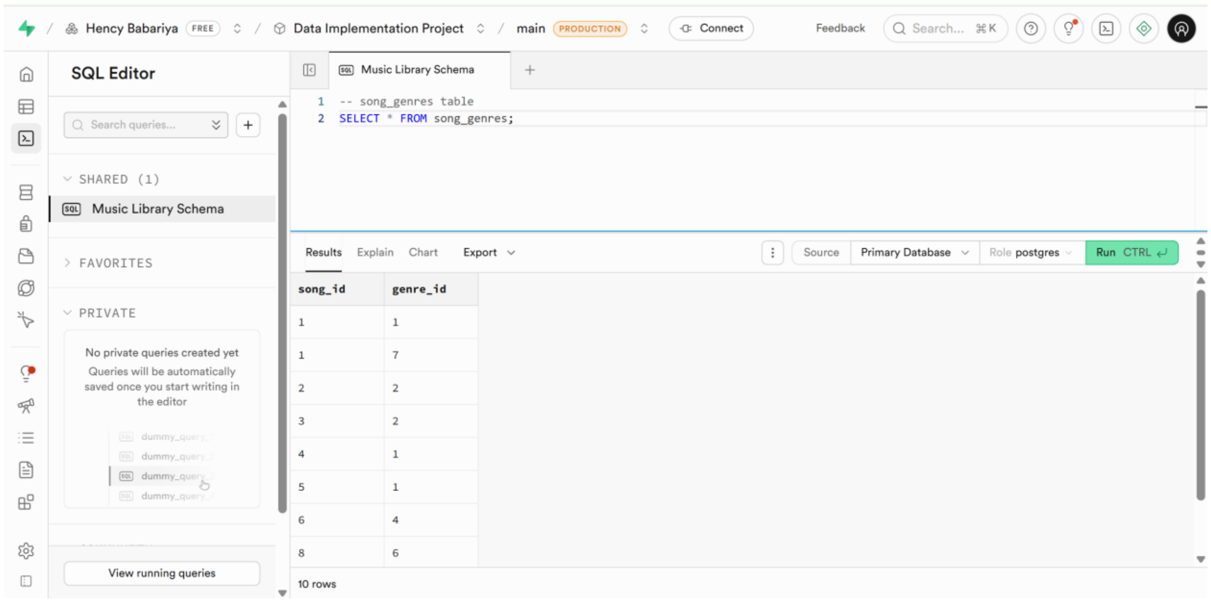
No private queries created yet Queries will be automatically saved once you start writing in the editor

```sql
-- playlist_songs table
SELECT * FROM playlist_songs;
```

playlist_id	song_id	added_at
1	1	2026-01-20 22:55:27.8588
1	4	2026-01-20 22:55:27.8588
2	5	2026-01-20 22:55:27.8588
2	9	2026-01-20 22:55:27.8588
3	6	2026-01-20 22:55:27.8588
4	1	2026-01-20 22:55:27.8588
4	2	2026-01-20 22:55:27.8588
5	3	2026-01-20 22:55:27.8588

10 rows

9. song_genres Table

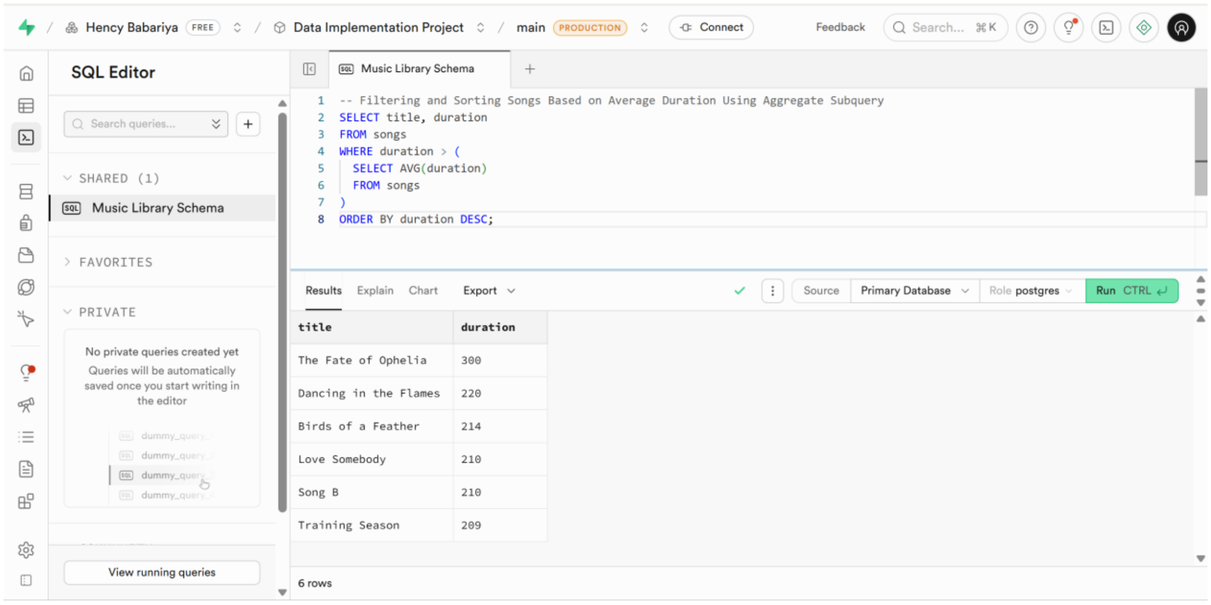

Hency Babariya FREE
Data Implementation Project
main PRODUCTION
Connect
Feedback
SQL Editor
Music Library Schema
SHARED (1)
FAVORITES
PRIVATE
No private queries created yet Queries will be automatically saved once you start writing in the editor
View running queries
Results Explain Chart Export
Source Primary Database Role postgres Run CTRL

```sql
-- song_genres table
SELECT * FROM song_genres;
```

song_id	genre_id
1	1
1	7
2	2
3	2
4	1
5	1
6	4
8	6

10 rows

10. Filtering Songs Based on Average Duration Using Aggregate Subquery

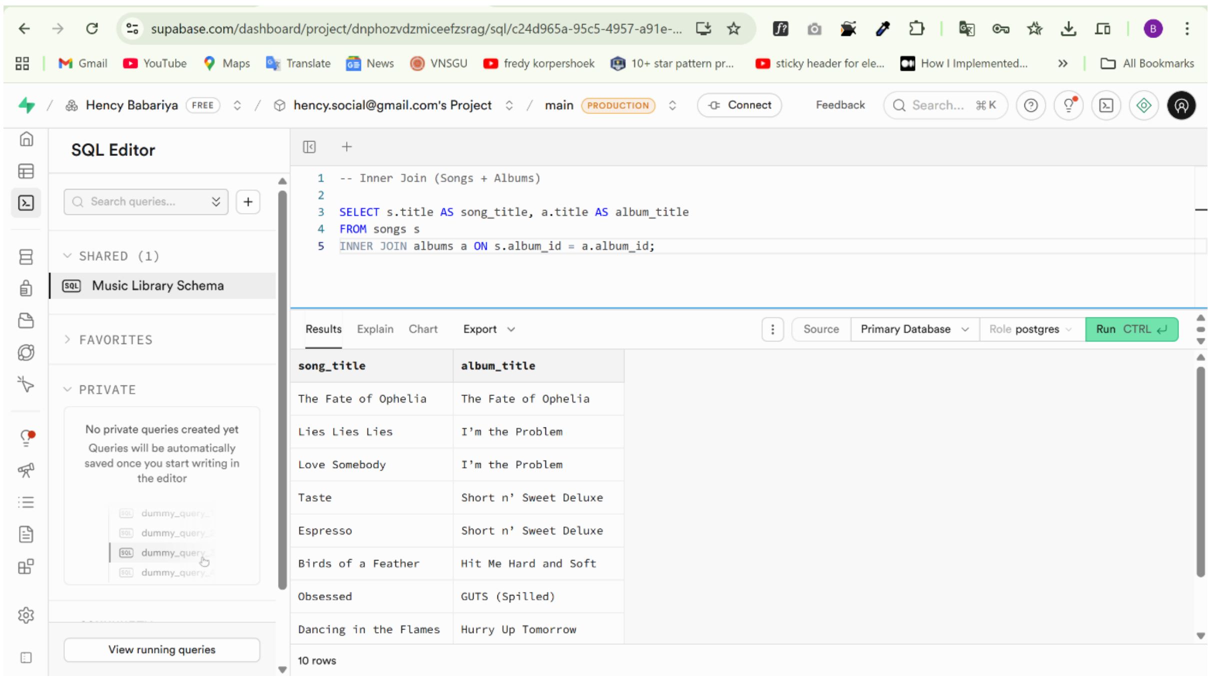

Hency Babariya FREE
Data Implementation Project
main PRODUCTION
Connect
Feedback
SQL Editor
Music Library Schema
SHARED (1)
FAVORITES
PRIVATE
No private queries created yet Queries will be automatically saved once you start writing in the editor
View running queries
Results Explain Chart Export
Source Primary Database Role postgres Run CTRL

```sql
-- Filtering and Sorting Songs Based on Average Duration Using Aggregate Subquery
SELECT title, duration
FROM songs
WHERE duration > (
  SELECT AVG(duration)
  FROM songs
)
ORDER BY duration DESC;
```

title	duration
The Fate of Ophelia	300
Dancing in the Flames	220
Birds of a Feather	214
Love Somebody	210
Song B	210
Training Season	209

6 rows

11. Inner Join (Songs + Albums)

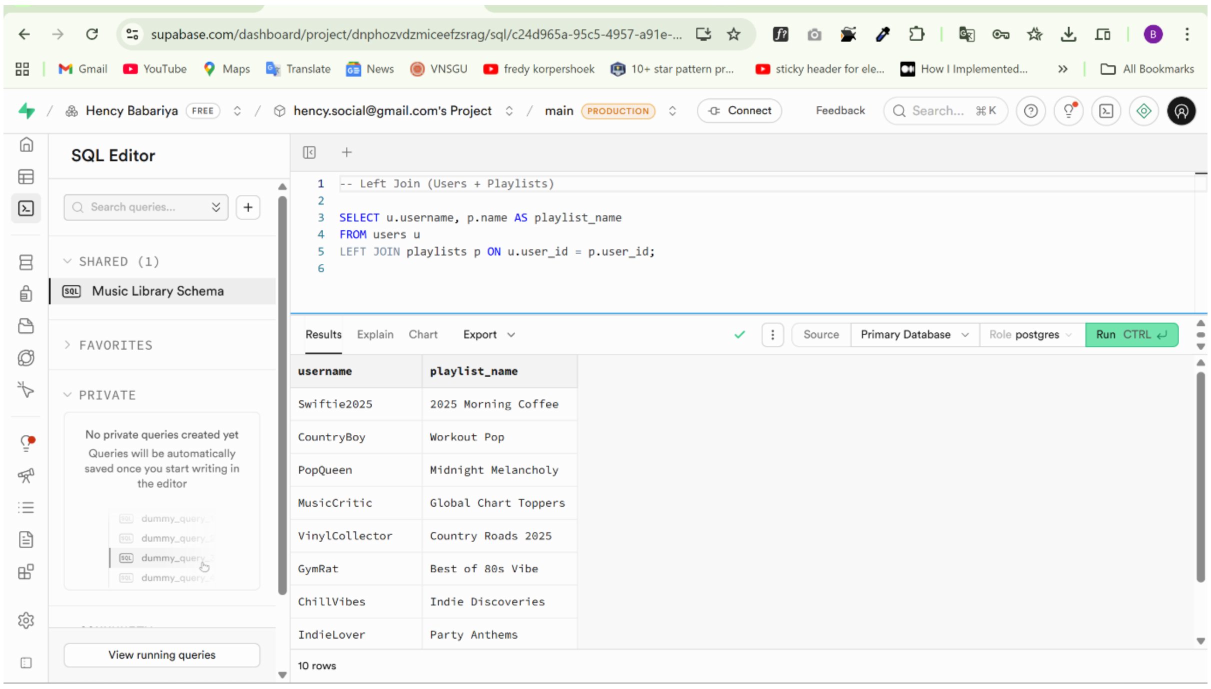


```sql
-- Inner Join (Songs + Albums)

SELECT s.title AS song_title, a.title AS album_title
FROM songs s
INNER JOIN albums a ON s.album_id = a.album_id;
```

song_title	album_title
The Fate of Ophelia	The Fate of Ophelia
Lies Lies Lies	I'm the Problem
Love Somebody	I'm the Problem
Taste	Short n' Sweet Deluxe
Espresso	Short n' Sweet Deluxe
Birds of a Feather	Hit Me Hard and Soft
Obsessed	GUTS (Spilled)
Dancing in the Flames	Hurry Up Tomorrow

10 rows

12. Left Join (Users + Playlists)

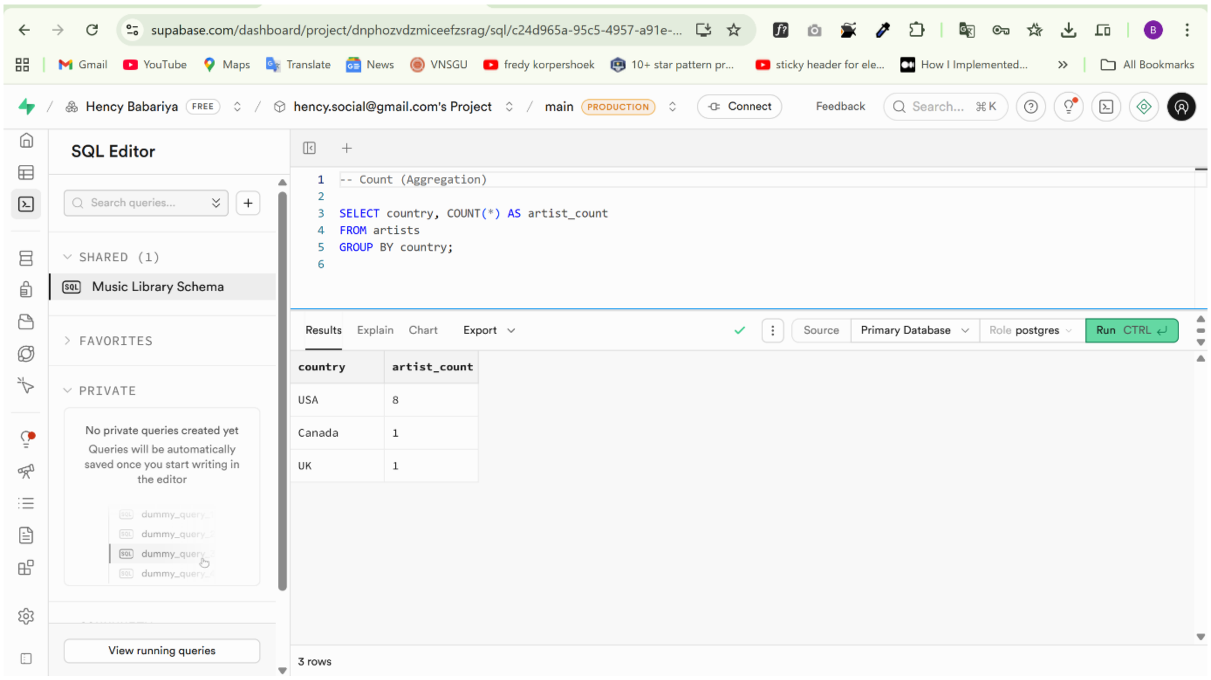


```sql
-- Left Join (Users + Playlists)

SELECT u.username, p.name AS playlist_name
FROM users u
LEFT JOIN playlists p ON u.user_id = p.user_id;
```

username	playlist_name
Swiftie2025	2025 Morning Coffee
CountryBoy	Workout Pop
PopQueen	Midnight Melancholy
MusicCritic	Global Chart Toppers
VinylCollector	Country Roads 2025
GymRat	Best of 80s Vibe
ChillVibes	Indie Discoveries
IndieLover	Party Anthems

10 rows

13. Count (Aggregation)

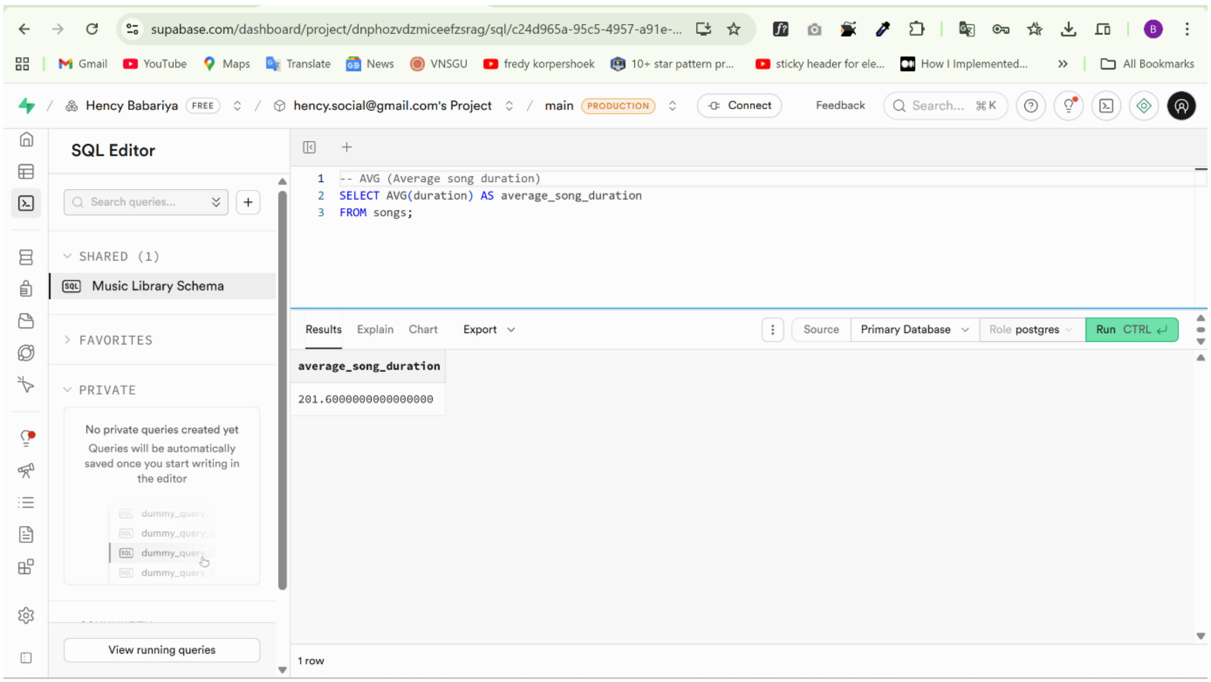


```sql
-- Count (Aggregation)

SELECT country, COUNT(*) AS artist_count
FROM artists
GROUP BY country;
```

```
country   artist_count
USA       8
Canada    1
UK        1
```

3 rows

14. AVG (Average song duration)

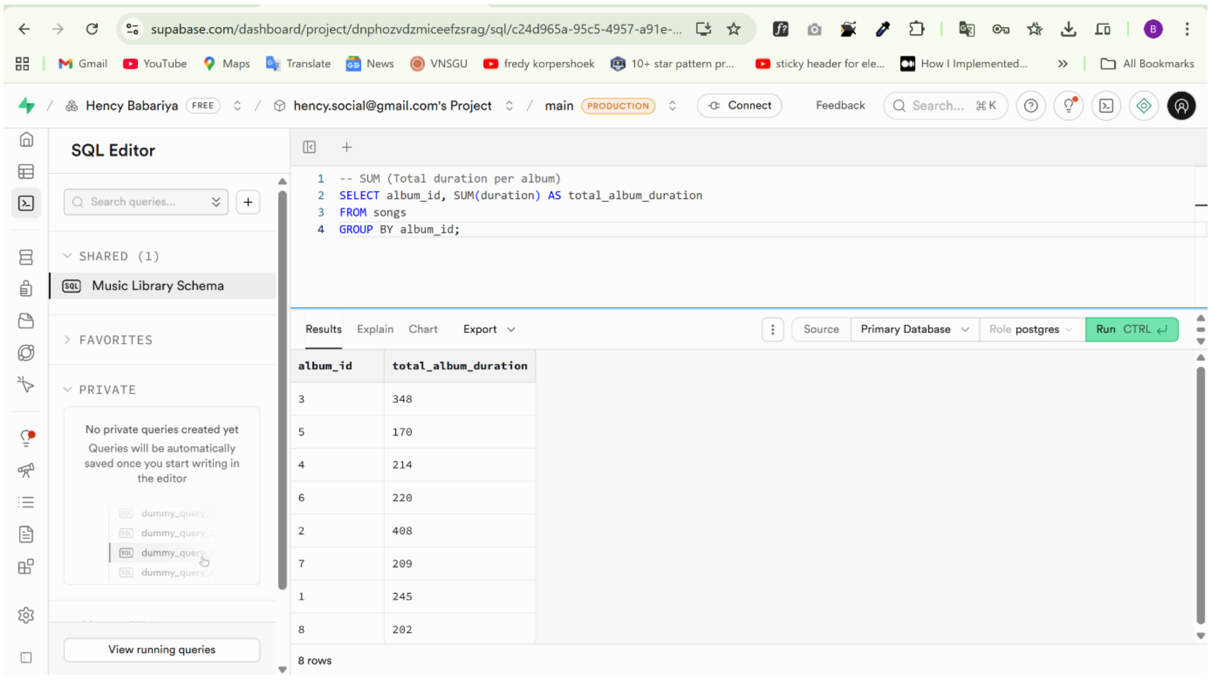


```sql
-- AVG (Average song duration)
SELECT AVG(duration) AS average_song_duration
FROM songs;
```

```
average_song_duration
201.6000000000000000
```

1 row

15. SUM (Total duration per album)

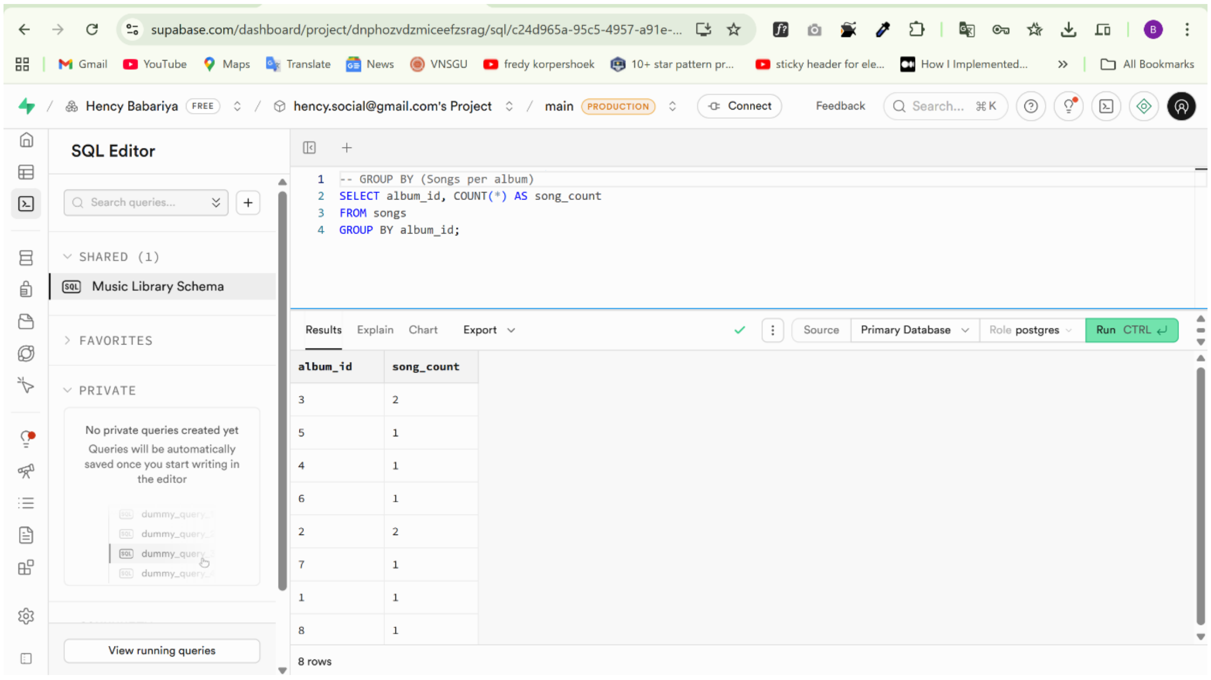


```
-- SUM (Total duration per album)
SELECT album_id, SUM(duration) AS total_album_duration
FROM songs
GROUP BY album_id;
```

album_id	total_album_duration
3	348
5	170
4	214
6	220
2	408
7	209
1	245
8	202

8 rows

16. GROUP BY (Songs per album)

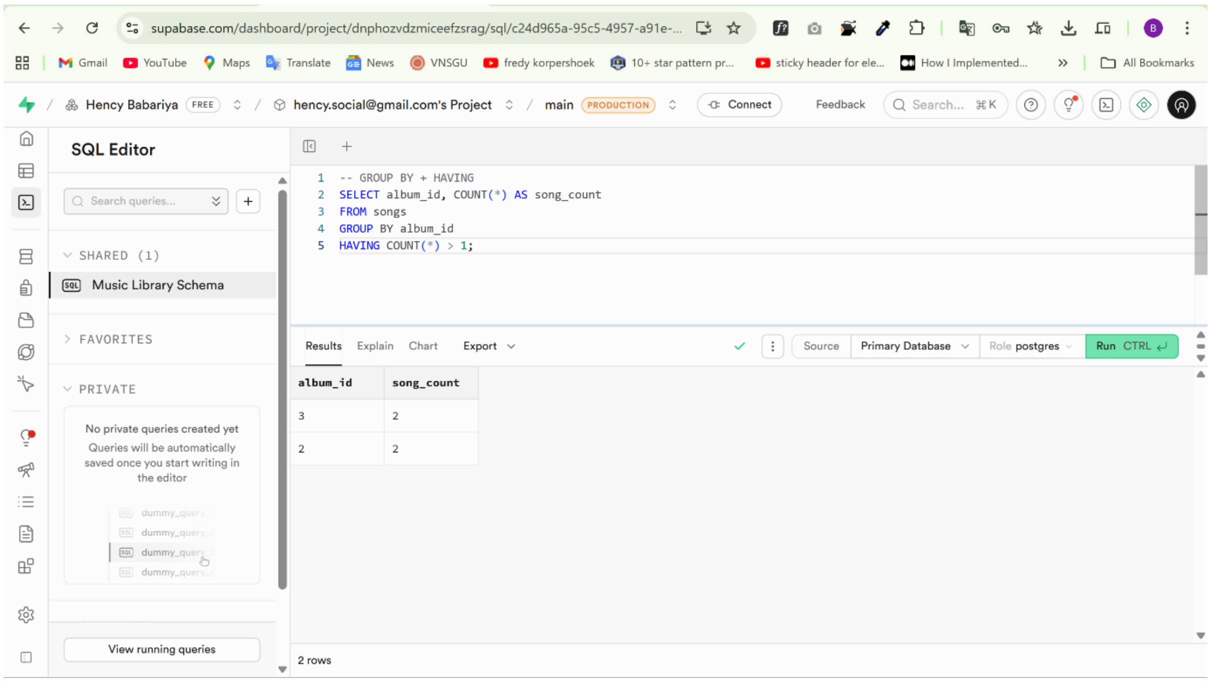


```
-- GROUP BY (Songs per album)
SELECT album_id, COUNT(*) AS song_count
FROM songs
GROUP BY album_id;
```

album_id	song_count
3	2
5	1
4	1
6	1
2	2
7	1
1	1
8	1

8 rows

17. GROUP BY + HAVING

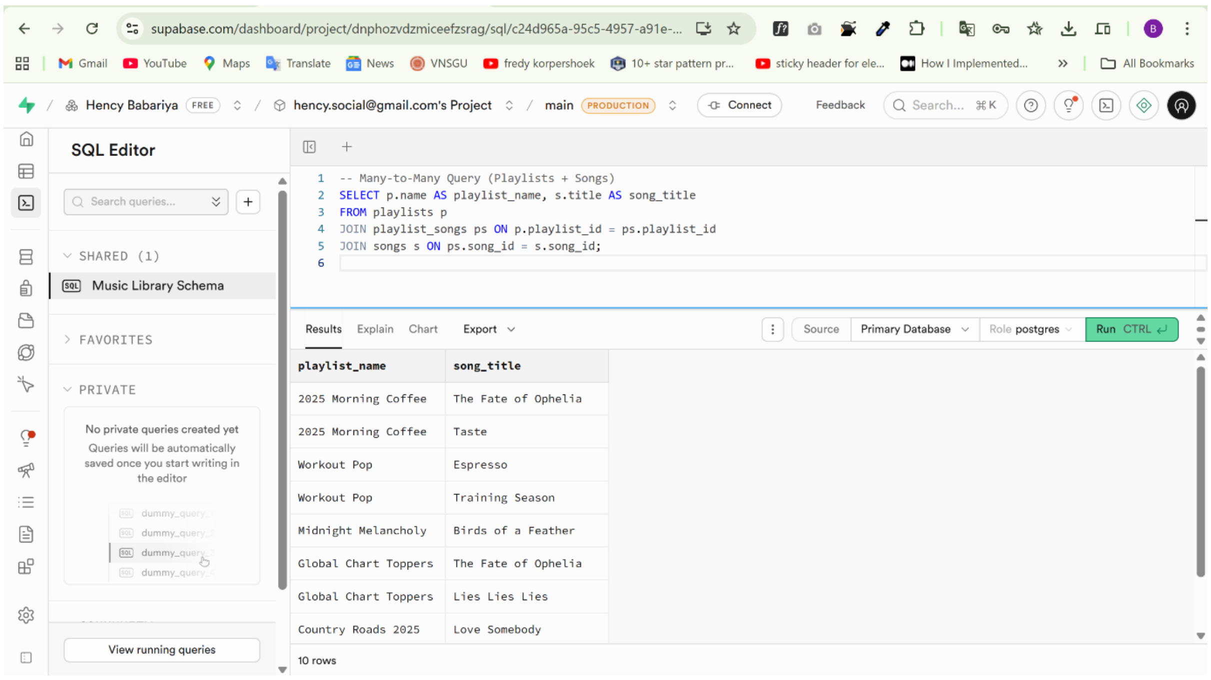

supabase.com/dashboard/project/dnphozvdzmiceefzsrag/sql/c24d965a-95c5-4957-a91e-...
Hency Babariya FREE
hency.social@gmail.com's Project
main PRODUCTION
Connect
Feedback
SQL Editor
SHARED (1)
Music Library Schema
FAVORITES
PRIVATE
No private queries created yet
Queries will be automatically saved once you start writing in the editor
View running queries
Results
Explain
Chart
Export
Source
Primary Database
Role postgres
Run CTRL

```sql
-- GROUP BY + HAVING
SELECT album_id, COUNT(*) AS song_count
FROM songs
GROUP BY album_id
HAVING COUNT(*) > 1;
```

album_id	song_count
3	2
2	2

2 rows

18. Many-to-Many Query (Playlists + Songs)

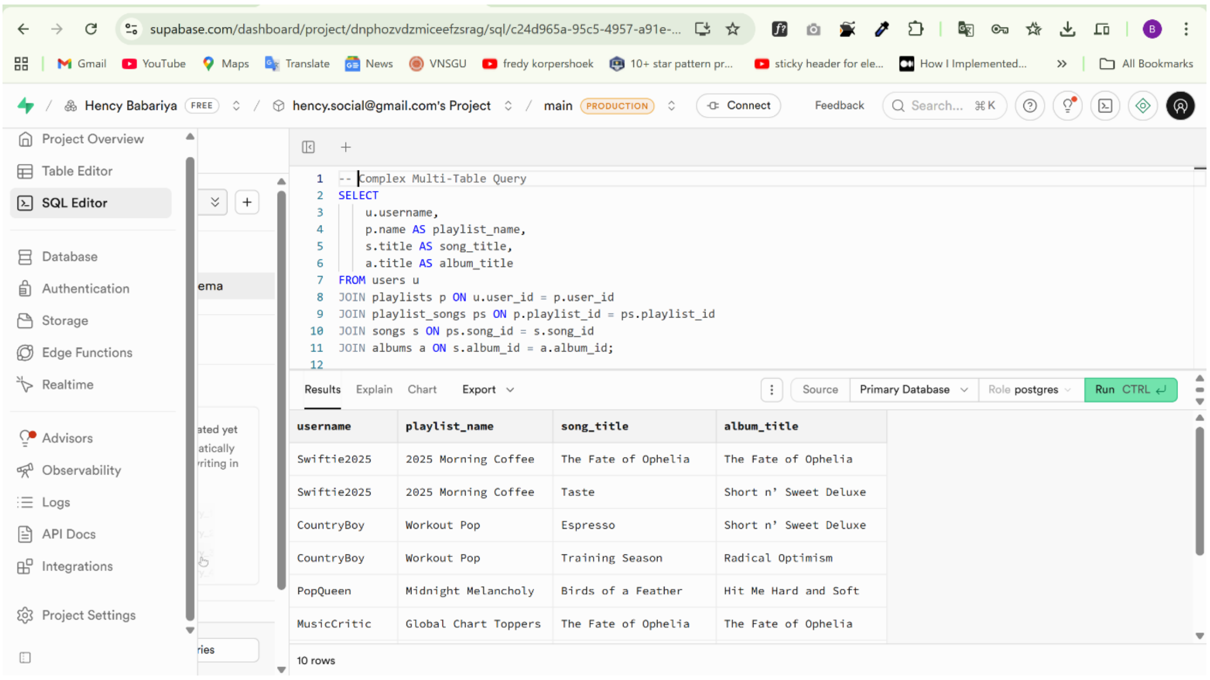

supabase.com/dashboard/project/dnphozvdzmiceefzsrag/sql/c24d965a-95c5-4957-a91e-...
Hency Babariya FREE
hency.social@gmail.com's Project
main PRODUCTION
Connect
Feedback
SQL Editor
SHARED (1)
Music Library Schema
FAVORITES
PRIVATE
No private queries created yet
Queries will be automatically saved once you start writing in the editor
View running queries
Results
Explain
Chart
Export
Source
Primary Database
Role postgres
Run CTRL

```sql
-- Many-to-Many Query (Playlists + Songs)
SELECT p.name AS playlist_name, s.title AS song_title
FROM playlists p
JOIN playlist_songs ps ON p.playlist_id = ps.playlist_id
JOIN songs s ON ps.song_id = s.song_id;

```

playlist_name	song_title
2025 Morning Coffee	The Fate of Ophelia
2025 Morning Coffee	Taste
Workout Pop	Espresso
Workout Pop	Training Season
Midnight Melancholy	Birds of a Feather
Global Chart Toppers	The Fate of Ophelia
Global Chart Toppers	Lies Lies Lies
Country Roads 2025	Love Somebody

10 rows

19. Complex Multi-Table Query



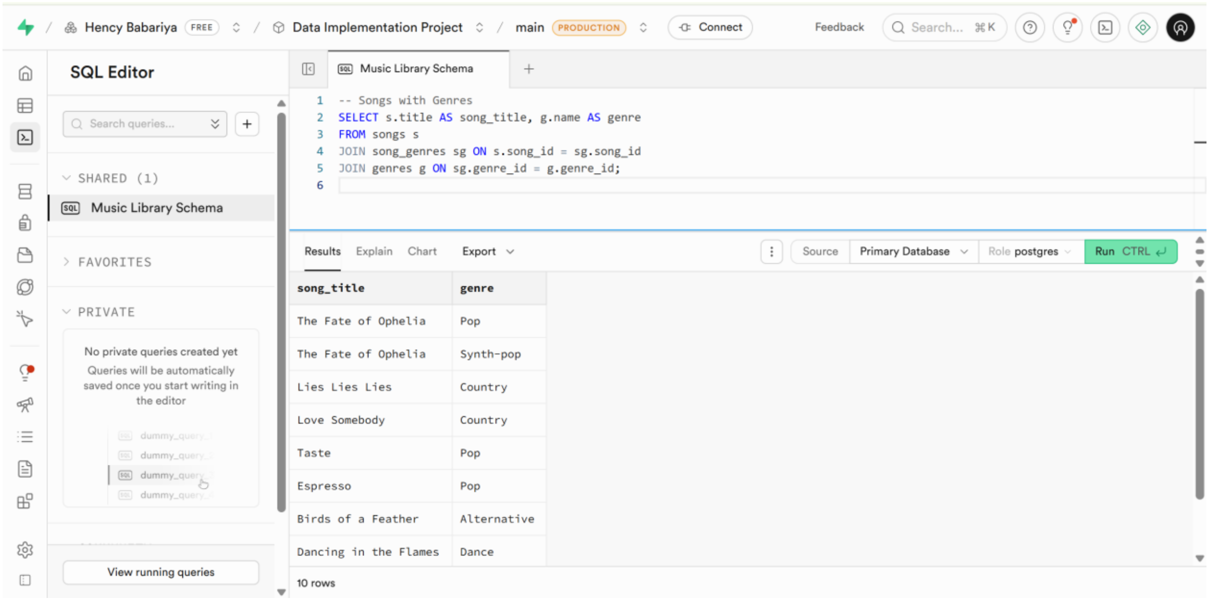
supabase.com/dashboard/project/dnphozvdzmiceefzsrag/sql/c24d965a-95c5-4957-a91e-...
Hency Babariya FREE / hency.social@gmail.com's Project / main PRODUCTION
Connect
Feedback
Project Overview
Table Editor
SQL Editor
Database
Authentication
Storage
Edge Functions
Realtime
Advisors
Observability
Logs
API Docs
Integrations
Project Settings
Results Explain Chart Export
Source Primary Database Role postgres Run CTRL

```sql
-- Complex Multi-Table Query
SELECT
    u.username,
    p.name AS playlist_name,
    s.title AS song_title,
    a.title AS album_title
FROM users u
JOIN playlists p ON u.user_id = p.user_id
JOIN playlist_songs ps ON p.playlist_id = ps.playlist_id
JOIN songs s ON ps.song_id = s.song_id
JOIN albums a ON s.album_id = a.album_id;
```

username	playlist_name	song_title	album_title
Swiftie2025	2025 Morning Coffee	The Fate of Ophelia	The Fate of Ophelia
Swiftie2025	2025 Morning Coffee	Taste	Short n' Sweet Deluxe
CountryBoy	Workout Pop	Espresso	Short n' Sweet Deluxe
CountryBoy	Workout Pop	Training Season	Radical Optimism
PopQueen	Midnight Melancholy	Birds of a Feather	Hit Me Hard and Soft
MusicCritic	Global Chart Toppers	The Fate of Ophelia	The Fate of Ophelia

10 rows

20. Songs with Genres



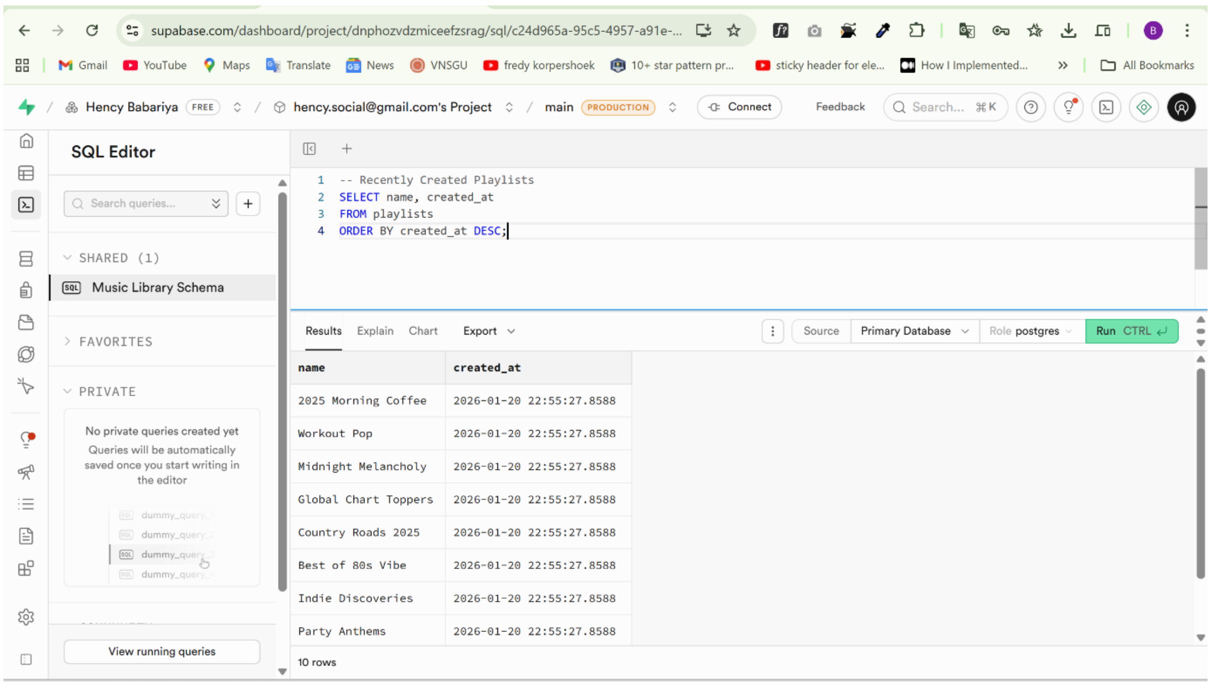
Hency Babariya FREE / Data Implementation Project / main PRODUCTION
Connect
Feedback
SQL Editor
Search queries...
SHARED (1)
Music Library Schema
FAVORITES
PRIVATE
No private queries created yet
Queries will be automatically saved once you start writing in the editor
View running queries
Results Explain Chart Export
Source Primary Database Role postgres Run CTRL

```sql
-- Songs with Genres
SELECT s.title AS song_title, g.name AS genre
FROM songs s
JOIN song_genres sg ON s.song_id = sg.song_id
JOIN genres g ON sg.genre_id = g.genre_id;
```

song_title	genre
The Fate of Ophelia	Pop
The Fate of Ophelia	Synth-pop
Lies Lies Lies	Country
Love Somebody	Country
Taste	Pop
Espresso	Pop
Birds of a Feather	Alternative
Dancing in the Flames	Dance

10 rows

21. Recently Created Playlists

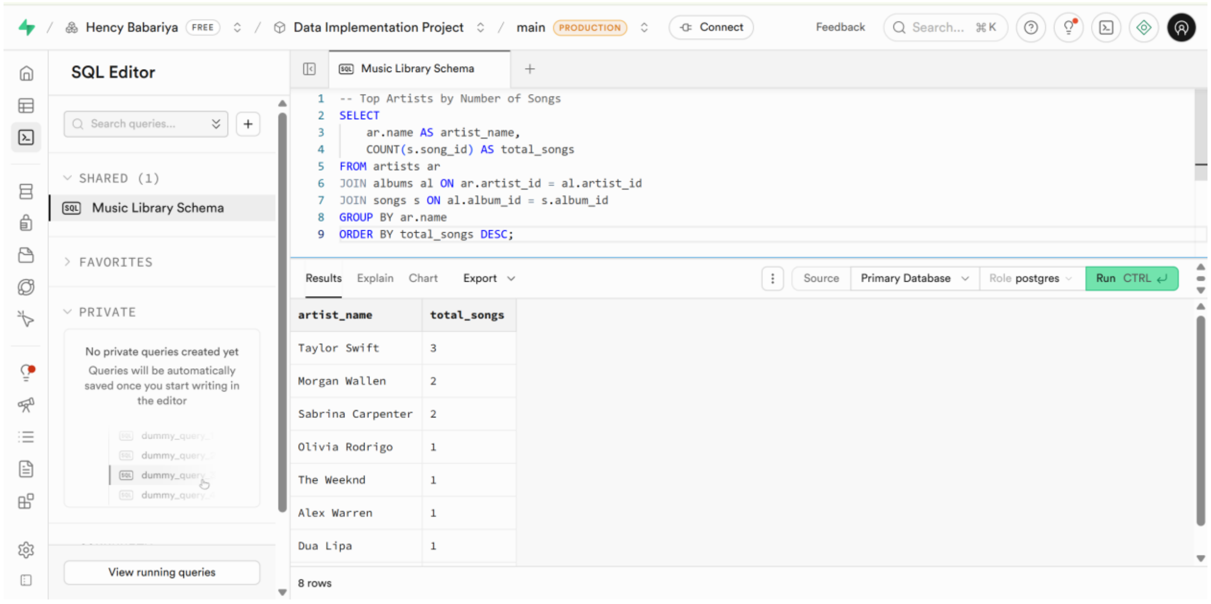


```sql
-- Recently Created Playlists
SELECT name, created_at
FROM playlists
ORDER BY created_at DESC;
```

name	created_at
2025 Morning Coffee	2026-01-20 22:55:27.8588
Workout Pop	2026-01-20 22:55:27.8588
Midnight Melancholy	2026-01-20 22:55:27.8588
Global Chart Toppers	2026-01-20 22:55:27.8588
Country Roads 2025	2026-01-20 22:55:27.8588
Best of 80s Vibe	2026-01-20 22:55:27.8588
Indie Discoveries	2026-01-20 22:55:27.8588
Party Anthems	2026-01-20 22:55:27.8588

10 rows

22. Top Artists by Number of Songs

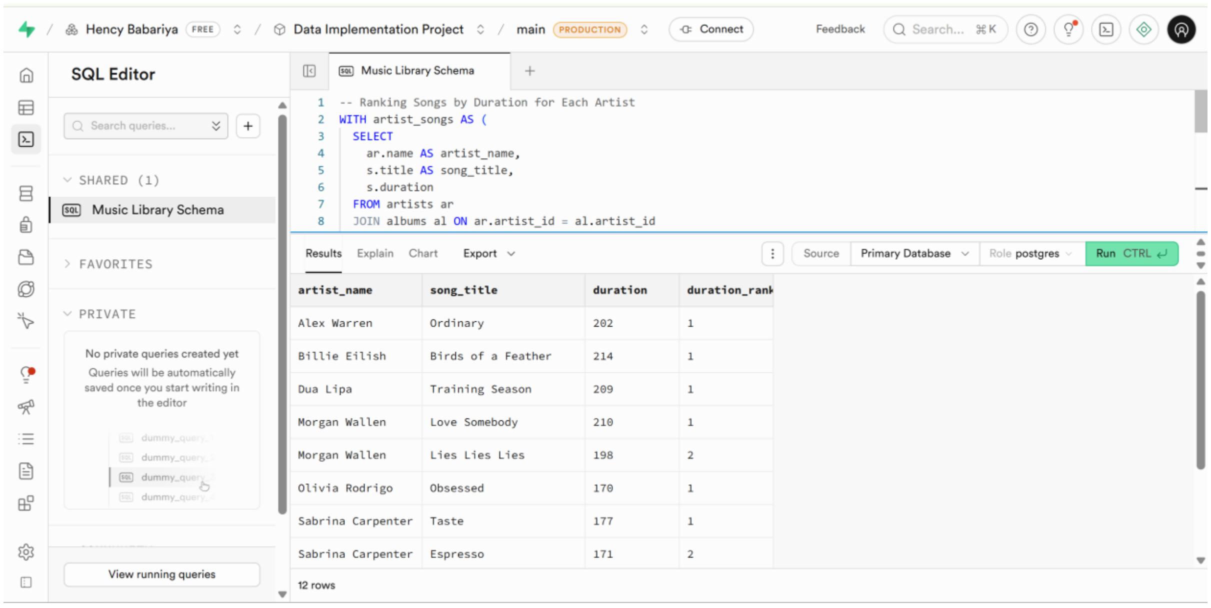


```sql
-- Top Artists by Number of Songs
SELECT
    ar.name AS artist_name,
    COUNT(s.song_id) AS total_songs
FROM artists ar
JOIN albums al ON ar.artist_id = al.artist_id
JOIN songs s ON al.album_id = s.album_id
GROUP BY ar.name
ORDER BY total_songs DESC;
```

artist_name	total_songs
Taylor Swift	3
Morgan Wallen	2
Sabrina Carpenter	2
Olivia Rodrigo	1
The Weeknd	1
Alex Warren	1
Dua Lipa	1

8 rows

23. Ranking Songs by Duration for Each Artist



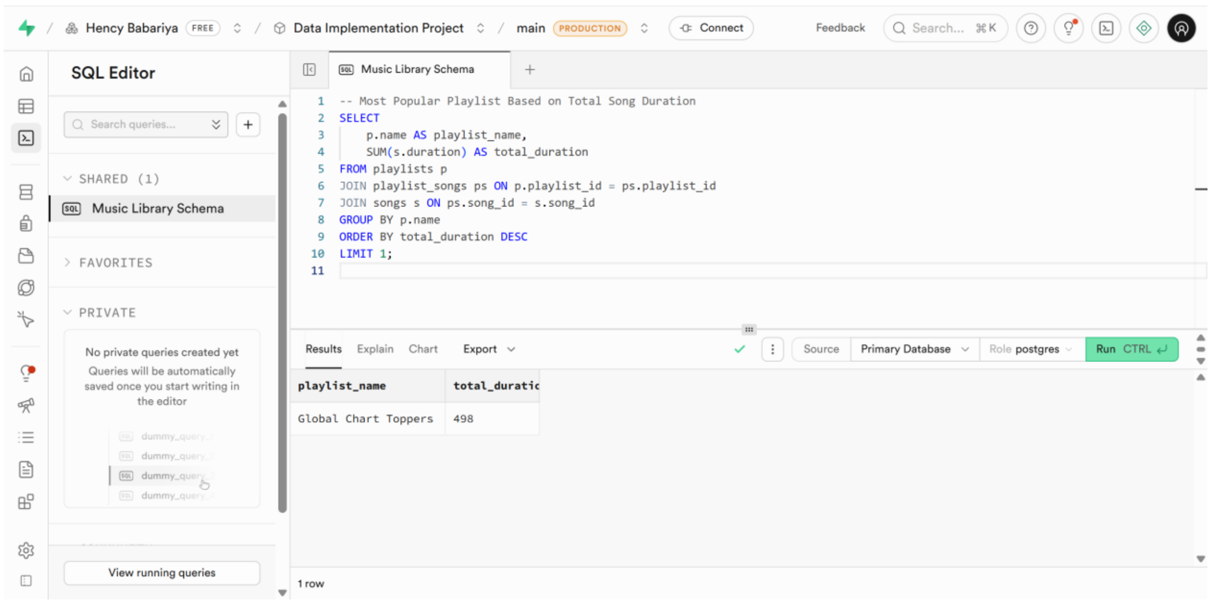
No private queries created yet
Queries will be automatically saved once you start writing in the editor

```sql
-- Ranking Songs by Duration for Each Artist
WITH artist_songs AS (
  SELECT
    ar.name AS artist_name,
    s.title AS song_title,
    s.duration
  FROM artists ar
  JOIN albums al ON ar.artist_id = al.artist_id
```

artist_name	song_title	duration	duration_rank
Alex Warren	Ordinary	202	1
Billie Eilish	Birds of a Feather	214	1
Dua Lipa	Training Season	209	1
Morgan Wallen	Love Somebody	210	1
Morgan Wallen	Lies Lies Lies	198	2
Olivia Rodrigo	Obsessed	170	1
Sabrina Carpenter	Taste	177	1
Sabrina Carpenter	Espresso	171	2

12 rows

24. Most Popular Playlist Based on Total Song Duration



No private queries created yet
Queries will be automatically saved once you start writing in the editor

```sql
-- Most Popular Playlist Based on Total Song Duration
SELECT
    p.name AS playlist_name,
    SUM(s.duration) AS total_duration
FROM playlists p
JOIN playlist_songs ps ON p.playlist_id = ps.playlist_id
JOIN songs s ON ps.song_id = s.song_id
GROUP BY p.name
ORDER BY total_duration DESC
LIMIT 1;
```

playlist_name	total_duratic
Global Chart Toppers	498

1 row