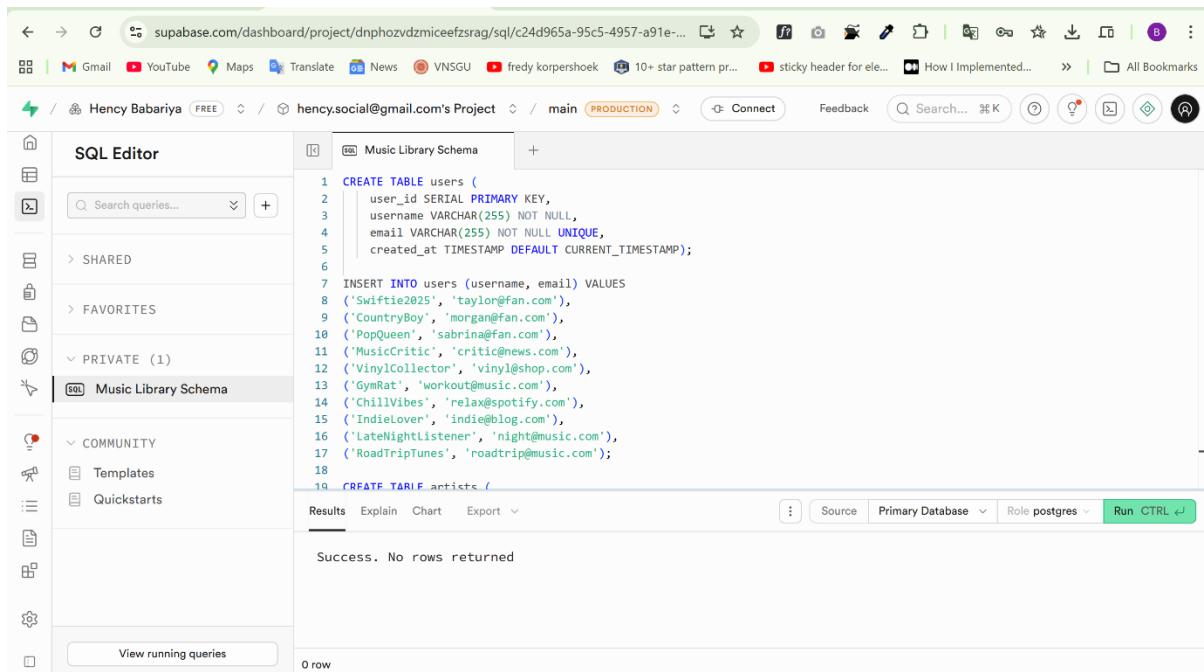


# Output of Section 2: Data Implementation

## 1. Successful creation of table and insertion of data

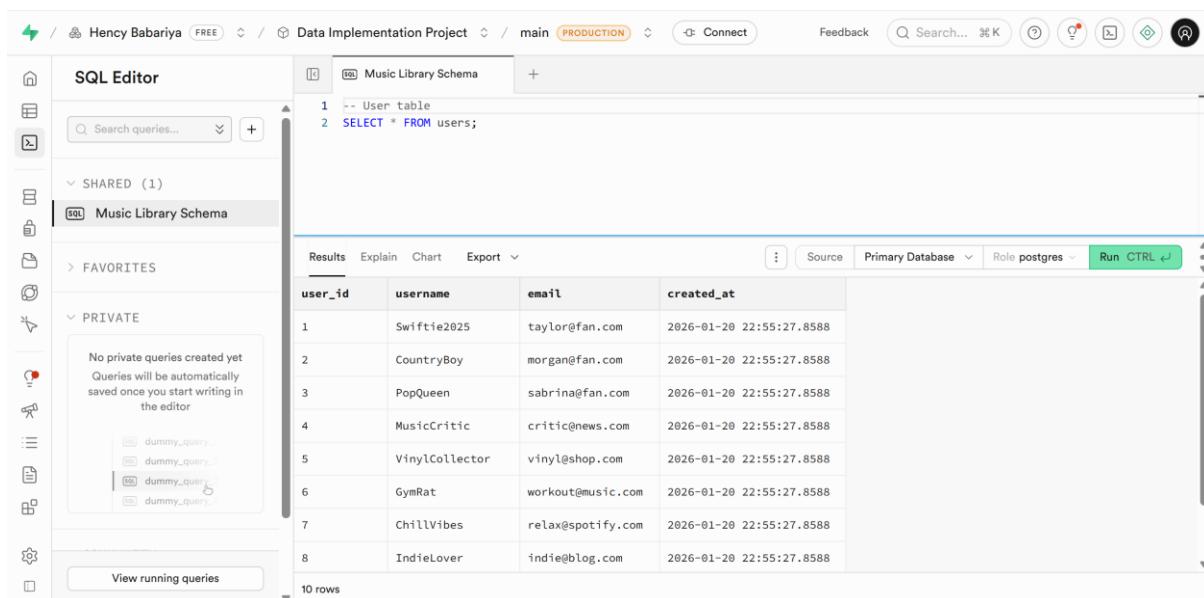


The screenshot shows the Supabase SQL Editor interface. On the left, there's a sidebar with project navigation (Shared, Favorites, Private), a schema tree (Music Library Schema selected), and a list of templates and quickstarts. The main area is the SQL Editor with the title "Music Library Schema". It contains the following SQL code:

```
1 CREATE TABLE users (
2     user_id SERIAL PRIMARY KEY,
3     username VARCHAR(255) NOT NULL,
4     email VARCHAR(255) NOT NULL UNIQUE,
5     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
6
7 INSERT INTO users (username, email) VALUES
8 ('Swiftie2025', 'taylor@fan.com'),
9 ('CountryBoy', 'morgan@fan.com'),
10 ('PopQueen', 'sabrina@fan.com'),
11 ('MusicCritic', 'critic@news.com'),
12 ('VinylCollector', 'vinyl@shop.com'),
13 ('GymRat', 'workout@music.com'),
14 ('ChillVibes', 'relax@spotify.com'),
15 ('IndieLover', 'indie@blog.com'),
16 ('LateNightListener', 'night@music.com'),
17 ('RoadTripTunes', 'roadtrip@music.com');
18
19 CREATE TABLE artists /
```

Below the code, the results pane shows "Success. No rows returned" and "0 row".

## 2. User Table



The screenshot shows the Supabase SQL Editor interface. The sidebar is similar to the previous one, with the "PRIVATE" section expanded to show "No private queries created yet" and "Queries will be automatically saved once you start writing in the editor". The SQL Editor has the title "Music Library Schema". It contains the following SQL code:

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

The results pane shows a table with the following data:

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

Below the table, it says "10 rows".

### 3. Artists Table

The screenshot shows a PostgreSQL database interface. On the left, there's a sidebar with icons for file operations, a search bar, and sections for 'SHARED' and 'PRIVATE' queries. The main area has tabs for 'SQL Editor' and 'Results'. In the SQL Editor tab, a query is run against the 'Music Library Schema':

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

The results tab displays the data from the 'artists' table:

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

The screenshot shows a PostgreSQL database interface, similar to the previous one. The sidebar and schema navigation are identical. The main area shows the results of a query on the 'genres' table:

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

The results tab displays the data from the 'genres' table:

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

The screenshot shows the pgAdmin interface with the following details:

- SQL Editor:** The code entered is:

```
-- albums table
SELECT * FROM albums;
```
- Results Table:** The results show 10 rows of data from the albums table, with columns: album\_id, title, release\_date, artist\_id.
- Data:**

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

The screenshot shows the pgAdmin interface with the following details:

- SQL Editor:** The code entered is:

```
-- songs table
SELECT * FROM songs;
```
- Results Table:** The results show 10 rows of data from the songs table, with columns: song\_id, title, duration, album\_id.
- Data:**

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

The screenshot shows a PostgreSQL database interface. On the left, there's a sidebar with icons for home, shared, favorites, and private queries. The main area has a SQL Editor tab open with the following code:

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

The results pane shows a table with the following data:

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

The screenshot shows a PostgreSQL database interface. On the left, there's a sidebar with icons for home, shared, favorites, and private queries. The main area has a SQL Editor tab open with the following code:

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

The results pane shows a table with the following data:

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

The screenshot shows the DataGrip interface with the SQL Editor tab selected. The code editor contains the following SQL query:

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

The results pane displays the data from the song\_genres table:

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

The screenshot shows the DataGrip interface with the SQL Editor tab selected. The code editor contains the following SQL query:

```
-- 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;
```

The results pane displays the filtered songs:

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)

The screenshot shows the Supabase SQL Editor interface. On the left, there's a sidebar with navigation links like Home, Shared (1), Favorites, and Private. The Private section indicates no private queries have been created yet. The main area has a SQL Editor with the following code:

```
1 -- Inner Join (Songs + Albums)
2
3 SELECT s.title AS song_title, a.title AS album_title
4 FROM songs s
5 INNER JOIN albums a ON s.album_id = a.album_id;
```

Below the editor is a results table with two columns: song\_title and album\_title. The data is as follows:

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

At the bottom, it says "10 rows".

## 12. Left Join (Users + Playlists)

The screenshot shows the Supabase SQL Editor interface. The sidebar and Private section are identical to the previous screenshot. The SQL Editor contains the following code:

```
1 -- Left Join (Users + Playlists)
2
3 SELECT u.username, p.name AS playlist_name
4 FROM users u
5 LEFT JOIN playlists p ON u.user_id = p.user_id;
6
```

The results table has two columns: username and playlist\_name. The data is as follows:

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

At the bottom, it says "10 rows".

## 13. Count (Aggregation)

The screenshot shows the Supabase SQL Editor interface. On the left, there's a sidebar with navigation links like Home, Shared (1), Favorites, and Private. The Private section notes that no private queries have been created yet. The main area is titled "SQL Editor" and contains the following SQL code:

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

Below the code, the results are displayed in a table:

country	artist_count
USA	8
Canada	1
UK	1

There are 3 rows in total.

## 14. AVG (Average song duration)

The screenshot shows the Supabase SQL Editor interface. The sidebar and Private section are identical to the previous screenshot. The main area is titled "SQL Editor" and contains the following SQL code:

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

Below the code, the results are displayed in a table:

average_song_duration
201.60000000000000

There is 1 row in total.

## 15. SUM (Total duration per album)

The screenshot shows the Supabase SQL Editor interface. On the left, there's a sidebar with navigation links like Home, Shared (1), Favorites, and Private. The Private section notes that no private queries have been created yet. The main area is titled "SQL Editor" and contains the following SQL code:

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

Below the code, the results are displayed in a table:

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)

The screenshot shows the Supabase SQL Editor interface. The sidebar and Private section are identical to the previous screenshot. The main area is titled "SQL Editor" and contains the following SQL code:

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

Below the code, the results are displayed in a table:

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

The screenshot shows the Supabase SQL Editor interface. On the left, there's a sidebar with icons for file operations, a search bar, and a list of shared and private queries. The main area has tabs for 'SQL Editor' and '+'. A code editor window contains the following SQL query:

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

Below the code editor is a results table with two rows:

album_id	song_count
3	2
2	2

The results panel includes tabs for 'Results', 'Explain', 'Chart', and 'Export'. It also shows connection details like 'Primary Database' set to 'postgres' and 'Role' set to 'postgres'.

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

The screenshot shows the Supabase SQL Editor interface. The sidebar and layout are similar to the previous screenshot. The code editor window contains the following SQL query:

```
1 -- Many-to-Many Query (Playlists + Songs)
2 SELECT p.name AS playlist_name, s.title AS song_title
3 FROM playlists p
4 JOIN playlist_songs ps ON p.playlist_id = ps.playlist_id
5 JOIN songs s ON ps.song_id = s.song_id;
6
```

Below the code editor is a results table with ten rows:

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

The results panel includes tabs for 'Results', 'Explain', 'Chart', and 'Export'. It also shows connection details like 'Primary Database' set to 'postgres' and 'Role' set to 'postgres'.

## 19. Complex Multi-Table Query

The screenshot shows the Supabase dashboard with the SQL Editor open. The sidebar on the left includes Project Overview, Table Editor, SQL Editor (selected), Database, Authentication, Storage, Edge Functions, and Realtime. The SQL Editor pane contains the following query:

```
1 -- Complex Multi-Table Query
2 SELECT
3     u.username,
4     p.name AS playlist_name,
5     s.title AS song_title,
6     a.title AS album_title
7 FROM users u
8 JOIN playlists p ON u.user_id = p.user_id
9 JOIN playlist_songs ps ON p.playlist_id = ps.playlist_id
10 JOIN songs s ON ps.song_id = s.song_id
11 JOIN albums a ON s.album_id = a.album_id;
12
```

The results pane shows a table with four columns: username, playlist\_name, song\_title, and album\_title. The data is as follows:

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

The screenshot shows the Supabase dashboard with the SQL Editor open. The sidebar on the left includes Project Overview, Table Editor, SQL Editor (selected), Database, Authentication, Storage, Edge Functions, and Realtime. The SQL Editor pane contains the following query:

```
1 -- Songs with Genres
2 SELECT s.title AS song_title, g.name AS genre
3 FROM songs s
4 JOIN song_genres sg ON s.song_id = sg.song_id
5 JOIN genres g ON sg.genre_id = g.genre_id;
6
```

The results pane shows a table with two columns: song\_title and genre. The data is as follows:

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

The screenshot shows the Supabase SQL Editor interface. On the left, there's a sidebar with navigation links like Home, Shared (1), Favorites, and Private. The Private section notes that no private queries have been created yet. The main area is titled "SQL Editor" and contains a code editor with the following SQL query:

```
1 -- Recently Created Playlists
2 SELECT name, created_at
3 FROM playlists
4 ORDER BY created_at DESC;
```

Below the code editor is a results table with the columns "name" and "created\_at". The data shows ten rows of playlists created on January 20, 2026, at 22:55:27.8588. The results are as follows:

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

The screenshot shows the Supabase SQL Editor interface. The sidebar is identical to the previous one, showing "Shared (1)" and "Music Library Schema". The Private section again notes that no private queries have been created yet. The main area is titled "SQL Editor" and contains the following SQL query:

```
1 -- Top Artists by Number of Songs
2 SELECT
3     ar.name AS artist_name,
4     COUNT(s.song_id) AS total_songs
5 FROM artists ar
6 JOIN albums al ON ar.artist_id = al.artist_id
7 JOIN songs s ON al.album_id = s.album_id
8 GROUP BY ar.name
9 ORDER BY total_songs DESC;
```

Below the code editor is a results table with the columns "artist\_name" and "total\_songs". The data shows eight rows of artists and their song counts. The results are as follows:

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

The screenshot shows a PostgreSQL database interface with the following details:

- SQL Editor:** Contains a query titled "Music Library Schema".

```
-- 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)
```
- Results:** Displays a table with four columns: `artist_name`, `song_title`, `duration`, and `duration_rank`. The data is as follows:

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

- Notes:** A message in the sidebar states: "No private queries created yet. Queries will be automatically saved once you start writing in the editor".

## 24. Most Popular Playlist Based on Total Song Duration

The screenshot shows a PostgreSQL database interface with the following details:

- SQL Editor:** Contains a query titled "Music Library Schema".

```
-- 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;
```
- Results:** Displays a table with two columns: `playlist_name` and `total_duration`. The data is as follows:

playlist_name	total_duration
Global Chart Toppers	498

- Notes:** A message in the sidebar states: "No private queries created yet. Queries will be automatically saved once you start writing in the editor".