



# MUSIC STORE ANALYSIS

The SQL Music Store Analysis project aims to explore and analyze a music store database using SQL queries. The objective is to extract meaningful insights from the data related to customers, sales, artists, albums, and genres.

**Ranjitha KL**  
[ranjitha.aradhya.944@gmail.com](mailto:ranjitha.aradhya.944@gmail.com)

# TABLE OF CONTENTS

Sl.no	Topic	Page Number
1	Objective	2
2	Schema Diagram	3
3	Solved Problems	4 - 9
4	Overall Analysis	10

# OBJECTIVE

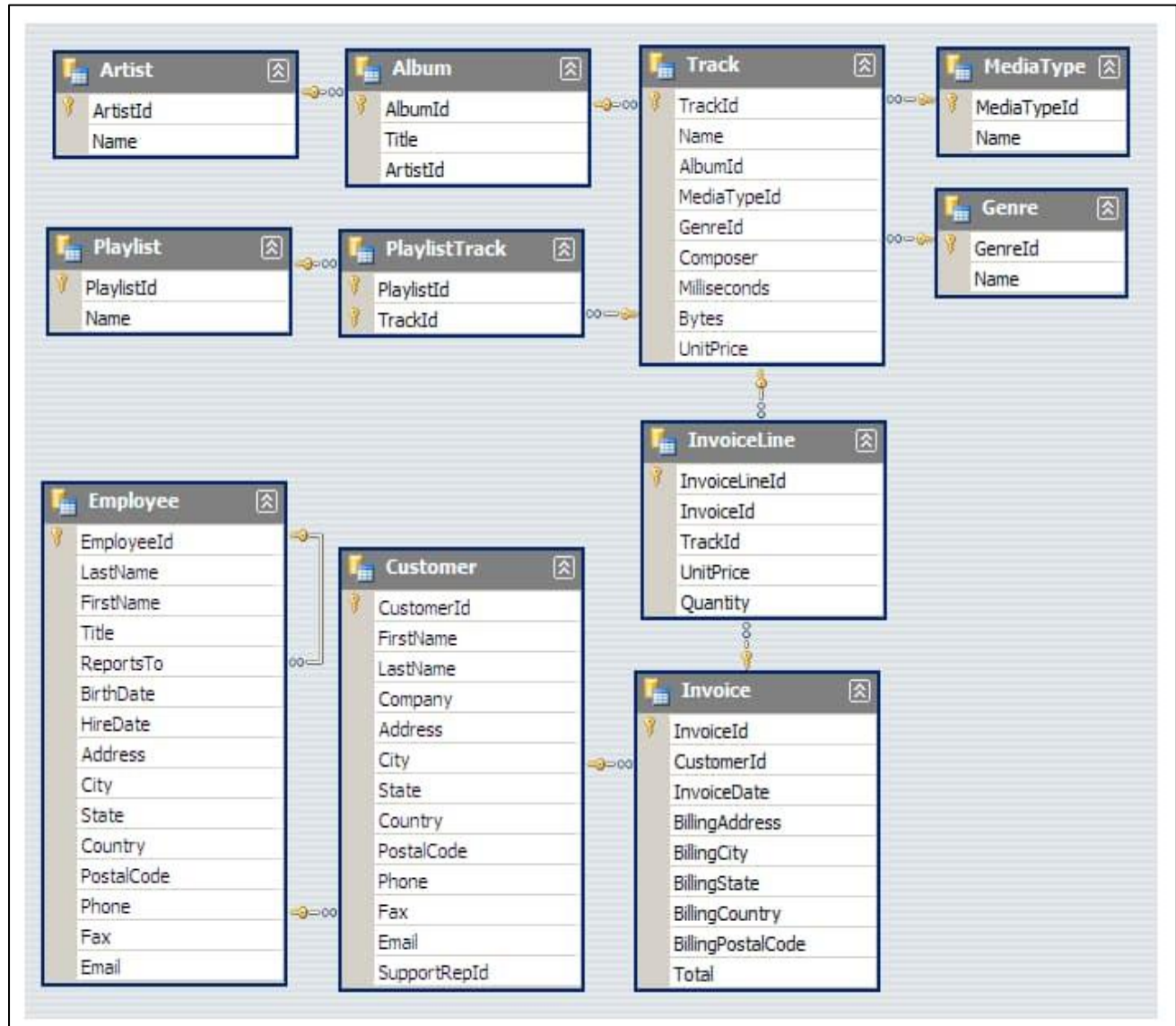
The SQL Music Store Analysis project aims to explore and analyze a music store database using SQL queries. The objective is to extract meaningful insights from the data related to customers, sales, artists, albums, and genres.

The analysis includes:

- Identifying the top-selling artists and albums
- Analyzing customer purchase behavior
- Finding the most popular music genres
- Understanding revenue trends and sales distribution
- Evaluating the performance of different store locations

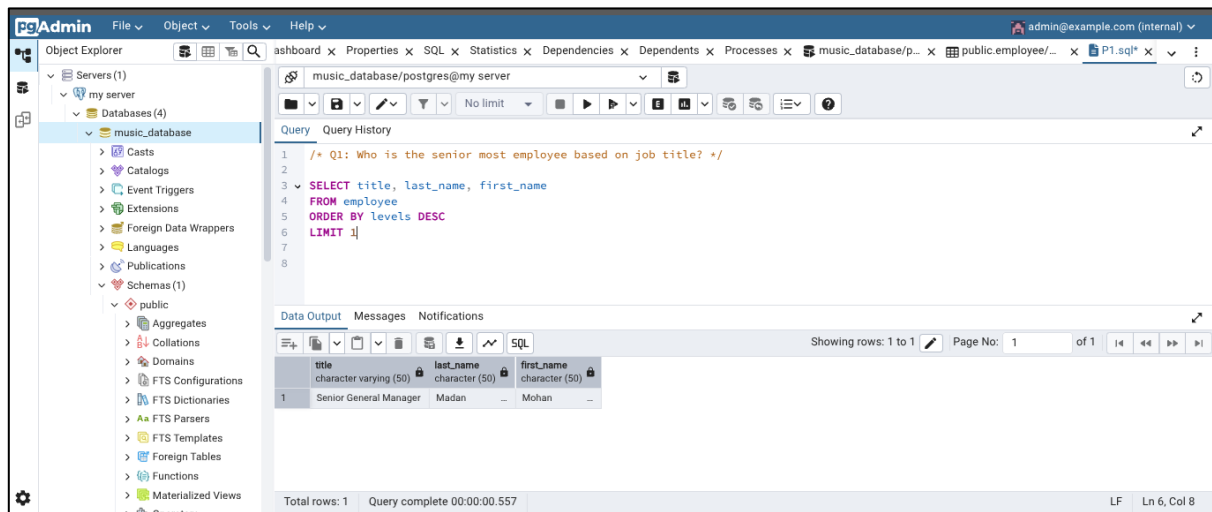
By leveraging SQL queries, this project helps in making data-driven decisions to optimize sales, improve customer engagement, and enhance business strategies for the music store.

# MUSIC PLAYLIST DATABASE SCHEMA



# MUSIC STORE ANALYSIS

## 1. Who is the senior most employee based on job title?

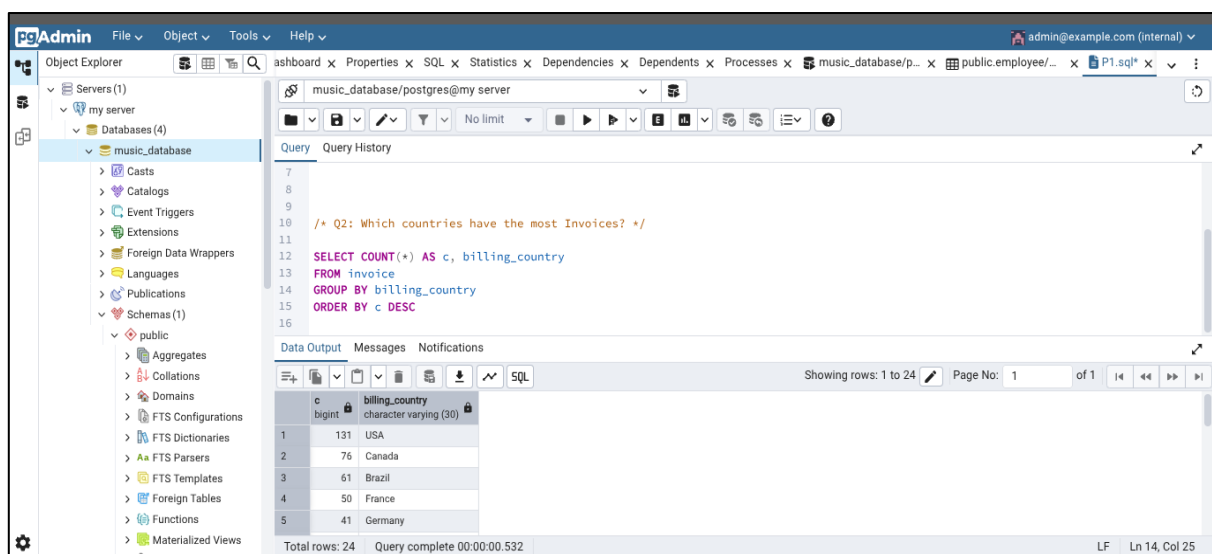


The screenshot shows the PgAdmin interface with a SQL query executed. The query is: `/* Q1: Who is the senior most employee based on job title? */  
SELECT title, last_name, first_name  
FROM employee  
ORDER BY levels DESC  
LIMIT 1`. The result set shows one row: 

title	last_name	first_name
Senior General Manager	Madan	Mohan

. The status bar indicates 'Total rows: 1' and 'Query complete 00:00:00.557'.

## 2. Which countries have the most Invoices?

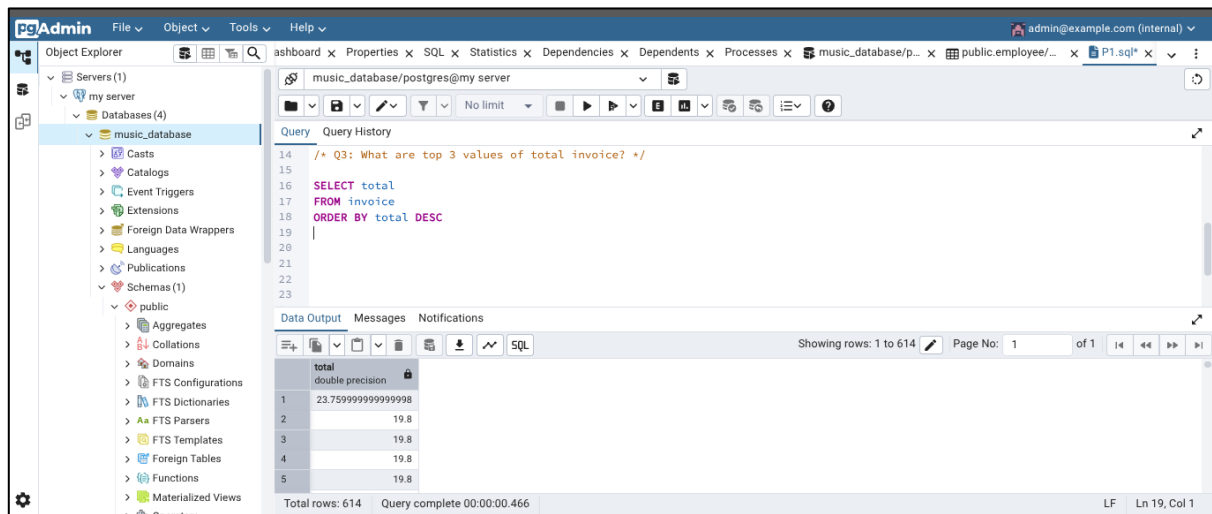


The screenshot shows the PgAdmin interface with a SQL query executed. The query is: `/* Q2: Which countries have the most Invoices? */  
SELECT COUNT(*) AS c, billing_country  
FROM invoice  
GROUP BY billing_country  
ORDER BY c DESC`. The result set shows five rows: 

c	billing_country
131	USA
76	Canada
61	Brazil
50	France
41	Germany

. The status bar indicates 'Total rows: 24' and 'Query complete 00:00:00.532'.

### 3. What are top 3 values of total invoice?

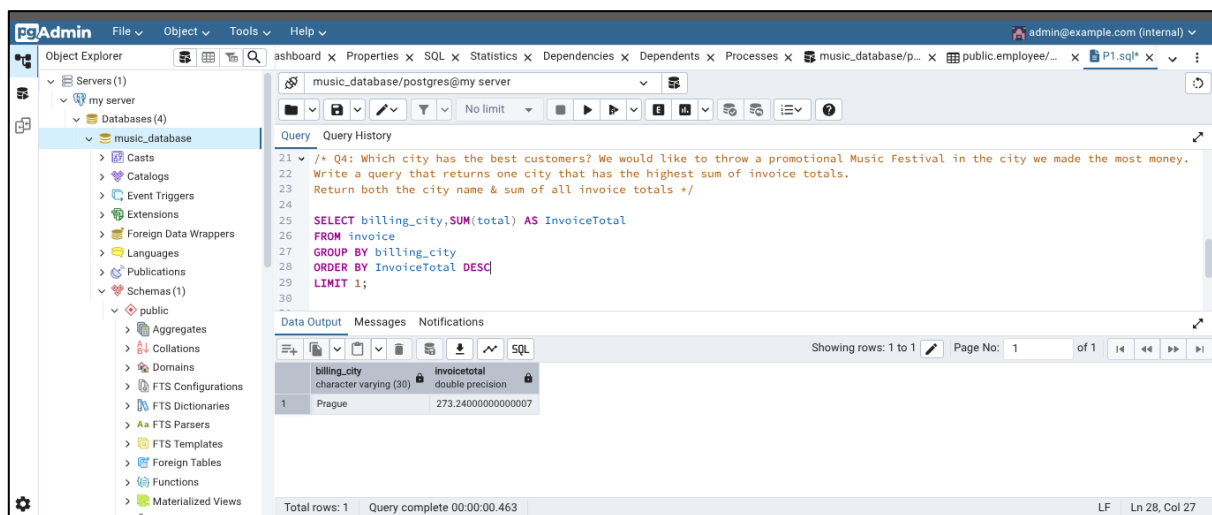


The screenshot shows the pgAdmin interface with a SQL query executed. The query is: `/* Q3: What are top 3 values of total invoice? */  
SELECT total  
FROM invoice  
ORDER BY total DESC`. The results are displayed in a table with 5 rows, showing the top 3 values of the total invoice.

	total	double precision
1	23.759999999999998	
2	19.8	
3	19.8	
4	19.8	
5	19.8	

Total rows: 614 Query complete 00:00:00.466

### 4. Which city has the best customers? We would like to throw a promotional Music Festival in the city we made the most money. Write a query that returns one city that has the highest sum of invoice totals. Return both the city name & sum of all invoice total



The screenshot shows the pgAdmin interface with a SQL query executed. The query is: `/* Q4: Which city has the best customers? We would like to throw a promotional Music Festival in the city we made the most money. Write a query that returns one city that has the highest sum of invoice totals. Return both the city name & sum of all invoice totals */  
SELECT billing_city, SUM(total) AS InvoiceTotal  
FROM invoice  
GROUP BY billing_city  
ORDER BY InvoiceTotal DESC  
LIMIT 1;`. The results are displayed in a table with 1 row, showing the city with the highest sum of invoice totals.

	billing_city	character varying (30)	invoicetotal	double precision
1	Prague		273.240000000000007	

Total rows: 1 Query complete 00:00:00.463

5. Who is the best customer? The customer who has spent the most money will be declared the best customer. Write a query that returns the person who has spent the most money.

The screenshot shows the PgAdmin interface with a SQL query executed. The query is as follows:

```
31 /* Q5: Who is the best customer? The customer who has spent the most money will be declared the best customer.
32 Write a query that returns the person who has spent the most money.*/
33
34 SELECT customer.customer_id, first_name, last_name, SUM(total) AS total_spending
35 FROM customer
36 JOIN invoice ON customer.customer_id = invoice.customer_id
37 GROUP BY customer.customer_id
38 ORDER BY total_spending DESC
39 LIMIT 1;
```

The data output shows one row:

customer_id	first_name	last_name	total_spending
1	S	Madhav	144.54000000000002

6. Write query to return the email, first name, last name, & Genre of all Rock Music listeners. Return your list ordered alphabetically by email starting with A.

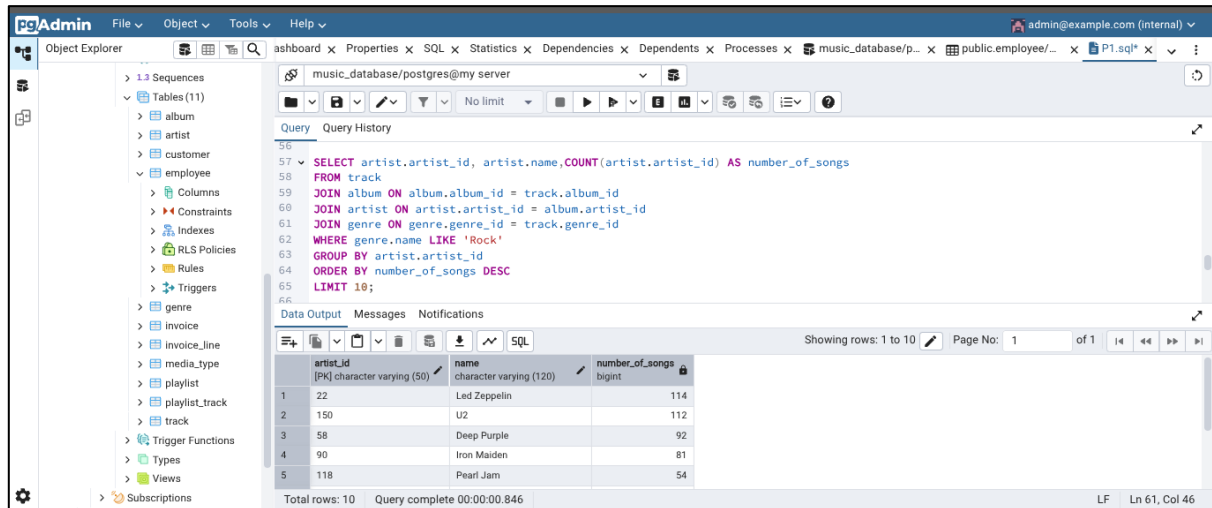
The screenshot shows the PgAdmin interface with a SQL query executed. The query is as follows:

```
42 SELECT DISTINCT email, first_name, last_name
43 FROM customer as c
44 JOIN invoice as i ON c.customer_id = i.customer_id
45 JOIN invoice_line as il ON i.invoice_id = il.invoice_id
46 WHERE track_id IN(
47     SELECT track_id FROM track as t
48     JOIN genre as g ON t.genre_id = g.genre_id
49     WHERE g.name LIKE 'Rock')
50 ORDER BY email;
```

The data output shows five rows:

email	first_name	last_name
aaronmitchell@yahoo.ca	Aaron	Mitchell
alero@uol.com.br	Alexandre	Rocha
astrid.gruber@apple.at	Astrid	Gruber
bjorn.hansen@yahoo.no	Bjorn	Hansen
camille.bernard@yahoo.fr	Camille	Bernard

7. Let's invite the artists who have written the most rock music in our dataset. Write a query that returns the Artist name and total track count of the top 10 rock bands.

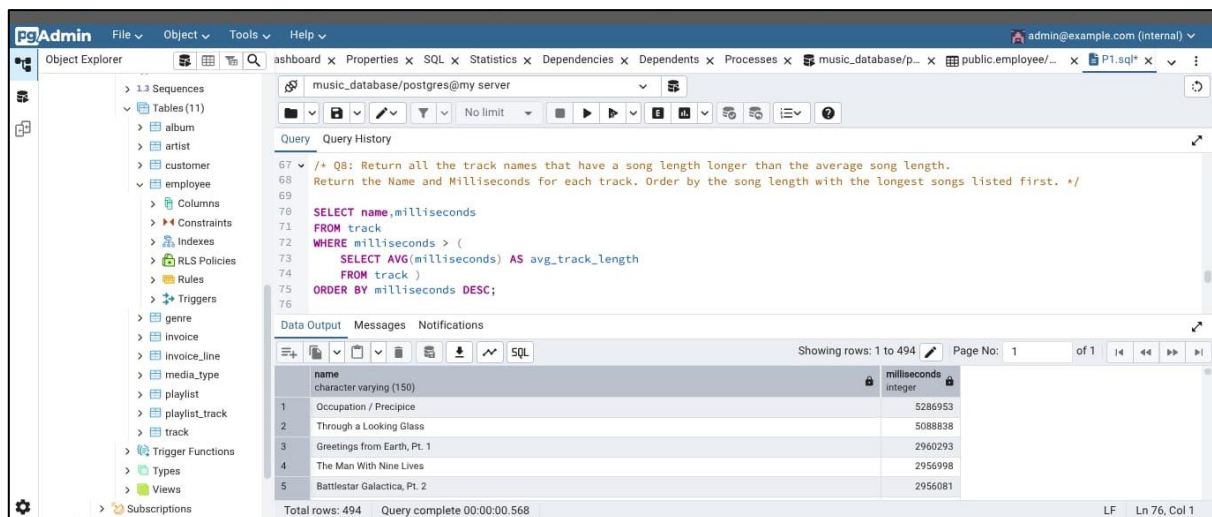


The screenshot shows the PgAdmin interface with a SQL query executed. The query selects the artist name and the count of tracks for rock artists, ordered by the count in descending order, limited to 10 results.

```
56 SELECT artist.artist_id, artist.name, COUNT(artist.artist_id) AS number_of_songs
57 FROM track
58 JOIN album ON album.album_id = track.album_id
59 JOIN artist ON artist.artist_id = album.artist_id
60 JOIN genre ON genre.genre_id = track.genre_id
61 WHERE genre.name LIKE 'Rock'
62 GROUP BY artist.artist_id
63 ORDER BY number_of_songs DESC
64 LIMIT 10;
```

artist_id	name	number_of_songs
22	Led Zeppelin	114
150	U2	112
58	Deep Purple	92
90	Iron Maiden	81
118	Pearl Jam	54

8. Return all the track names that have a song length longer than the average song length. Return the Name and Milliseconds for each track. Order by the song length with the longest songs listed first.



The screenshot shows the PgAdmin interface with a SQL query executed. The query selects track names and milliseconds for tracks longer than the average song length, ordered by milliseconds in descending order.

```
67 /* Q8: Return all the track names that have a song length longer than the average song length.
68 Return the Name and Milliseconds for each track. Order by the song length with the longest songs listed first. */
69
70 SELECT name, milliseconds
71 FROM track
72 WHERE milliseconds > (
73     SELECT AVG(milliseconds) AS avg_track_length
74     FROM track )
75 ORDER BY milliseconds DESC;
```

name	milliseconds
Occupation / Precipice	5286953
Through a Looking Glass	5088838
Greetings from Earth, Pt. 1	2960293
The Man With Nine Lives	2956998
Battlestar Galactica, Pt. 2	2956081



9. Return all the track names that have a song length longer than the average song length. Return the Name and Milliseconds for each track. Order by the song length with the longest songs listed first.

pgAdmin interface showing a SQL query and its results. The query is:

```
WITH best_selling_artist AS (
SELECT artist_id AS artist_id, artist.name AS artist_name, SUM(invoice_line.unit_price*invoice_line.quantity) AS total_sales
FROM invoice_line
JOIN track ON track.track_id = invoice_line.track_id
JOIN album ON album.album_id = track.album_id
JOIN artist ON artist.artist_id = album.artist_id
GROUP BY 1
ORDER BY 3 DESC
LIMIT 1 )
SELECT customer_id, first_name, last_name, best_selling_artist.artist_name, SUM(il.unit_price*il.quantity) AS amount_spent
FROM invoice i
JOIN customer c ON c.customer_id = i.customer_id
JOIN invoice_line il ON il.invoice_id = i.invoice_id
JOIN track t ON t.track_id = il.track_id
JOIN album alb ON alb.album_id = t.album_id
JOIN best_selling_artist bsa ON bsa.artist_id = alb.artist_id
GROUP BY 1,2,3,4
ORDER BY 5 DESC;
```

The results show 5 rows of data:

customer_id	first_name	last_name	artist_name	amount_spent
1	46	Hugh	O'Reilly	27.719999999999985
2	38	Niklas	Schröder	18.81
3	3	François	Tremblay	17.82
4	34	João	Fernandes	16.830000000000002
5	53	Phil	Hughes	11.88

pgAdmin interface showing a SQL query and its results. The query is:

```
WITH best_selling_artist AS (
SELECT artist_id AS artist_id, artist.name AS artist_name, SUM(invoice_line.unit_price*invoice_line.quantity) AS total_sales
FROM invoice_line
JOIN track ON track.track_id = invoice_line.track_id
JOIN album ON album.album_id = track.album_id
JOIN artist ON artist.artist_id = album.artist_id
GROUP BY 1
ORDER BY 3 DESC
LIMIT 1 )
SELECT customer_id, first_name, last_name, best_selling_artist.artist_name, SUM(il.unit_price*il.quantity) AS amount_spent
FROM invoice i
JOIN customer c ON c.customer_id = i.customer_id
JOIN invoice_line il ON il.invoice_id = i.invoice_id
JOIN track t ON t.track_id = il.track_id
JOIN album alb ON alb.album_id = t.album_id
JOIN best_selling_artist bsa ON bsa.artist_id = alb.artist_id
GROUP BY 1,2,3,4
ORDER BY 5 DESC;
```

The results show 5 rows of data:

customer_id	first_name	last_name	artist_name	amount_spent
1	46	Hugh	O'Reilly	27.719999999999985
2	38	Niklas	Schröder	18.81
3	3	François	Tremblay	17.82
4	34	João	Fernandes	16.830000000000002
5	53	Phil	Hughes	11.88

10. We want to find out the most popular music Genre for each country. We determine the most popular genre as the genre with the highest amount of purchases. Write a query that returns each country along with the top Genre. For countries where the maximum number of purchases is shared return all Genres.

The screenshot shows the PgAdmin interface with a SQL query executed. The query uses a window function to rank genres by purchase count within each country. The results table shows the top genre for each country, with ties for Argentina and Brazil.

```

WITH popular_genre AS (
    SELECT COUNT(invoice_line.quantity) AS purchases, customer.country, genre.name, genre.genre_id,
    ROW_NUMBER() OVER(PARTITION BY customer.country ORDER BY COUNT(invoice_line.quantity) DESC) AS RowNo
    FROM invoice_line
    JOIN invoice ON invoice.invoice_id = invoice_line.invoice_id
    JOIN customer ON customer.customer_id = invoice.customer_id
    JOIN track ON track.track_id = invoice_line.track_id
    JOIN genre ON genre.genre_id = track.genre_id
    GROUP BY 2,3,4 ORDER BY 2 ASC, 1 DESC )
SELECT * FROM popular_genre WHERE RowNo <= 1

```

	purchases	country	name	genre_id	rowno
1	17	Argentina	Alternative & Punk	4	1
2	34	Australia	Rock	1	1
3	40	Austria	Rock	1	1
4	26	Belgium	Rock	1	1
5	205	Brazil	Rock	1	1

Total rows: 24 Query complete 00:00:00.450

11. Write a query that determines the customer that has spent the most on music for each country. Write a query that returns the country along with the top customer and how much they spent. For countries where the top amount spent is shared, provide all customers who spent this amount.

The screenshot shows the PgAdmin interface with a SQL query executed. The query uses a window function to rank customers by total spending within each country. The results table shows the top customer for each country, with ties for Argentina, Australia, Austria, and Belgium.

```

WITH Customer_with_country AS (
    SELECT customer.customer_id, first_name, last_name, billing_country, SUM(total) AS total_spending,
    ROW_NUMBER() OVER(PARTITION BY billing_country ORDER BY SUM(total) DESC) AS RowNo
    FROM invoice
    JOIN customer ON customer.customer_id = invoice.customer_id
    GROUP BY 1,2,3,4
    ORDER BY 4 ASC, 5 DESC)
SELECT * FROM Customer_with_country WHERE RowNo <= 1

```

	customer_id	first_name	last_name	billing_country	total_spending	rowno
1	56	Diego	Gutiérrez	Argentina	39.6	1
2	55	Mark	Taylor	Australia	81.18	1
3	7	Astrid	Gruber	Austria	69.3	1
4	8	Daan	Peeters	Belgium	60.38999999999999	1
5	1	Luis	Gonçalves	Brazil	108.89999999999998	1

Total rows: 24 Query complete 00:00:00.487

# OVERALL ANALYSIS

The SQL Music Store Analysis project explores a music store database using SQL queries to extract meaningful insights related to customers, sales, artists, albums, and genres. The analysis is divided into key sections, each addressing important business questions and trends.

## Key Findings

- Employee Hierarchy
- Sales Insights
- Customer Behavior
- Music Preferences & Popularity
- Track Analysis
- Regional Trends

## Business Impact

By leveraging SQL-based data analysis, this project helps the music store make informed, data-driven decisions. Insights gained can be used for:

- Optimizing sales by focusing on high-performing regions and customers.
- Enhancing customer engagement through targeted marketing campaigns.
- Strategic business growth by understanding genre popularity and revenue streams.