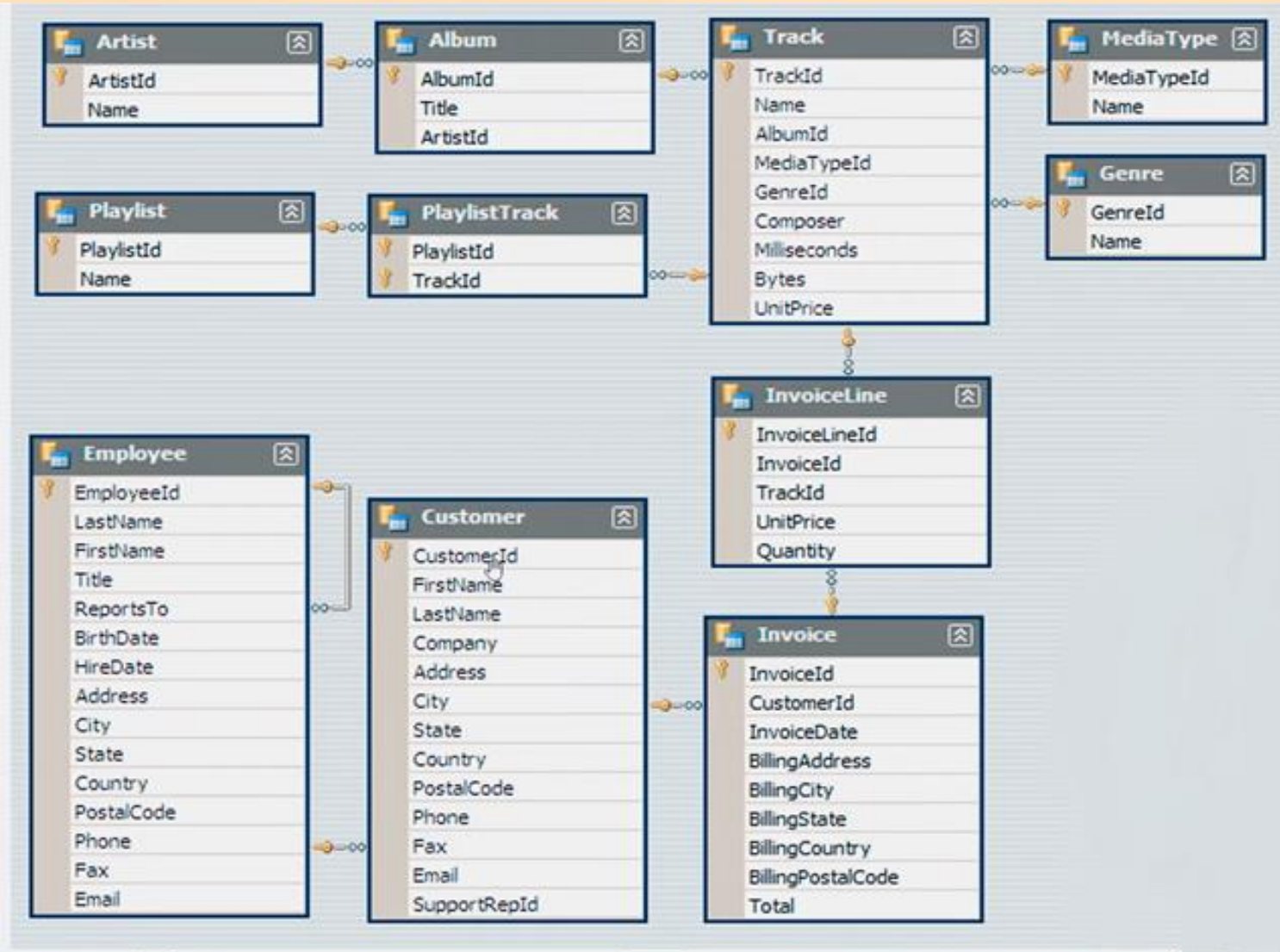


The background is a light orange and yellow gradient with various musical instruments and notes scattered around. In the top left, there's a violin. To its right is a trumpet. On the left side, there's a saxophone. In the bottom left, there's a xylophone with colorful keys. In the bottom right, there's a drum with two drumsticks. Several musical notes are floating around the instruments.

Music Store Data Analysis Using SQL

Schema of the Model



QUESTION 1:

Who is the senior most employee based on job title?

Query And Output:

The screenshot shows a database management tool interface. On the left is a sidebar with a tree view of databases. The 'music_database' database is selected, showing its contents: Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrapper, Languages, Publications, Schemas, Subscriptions, and postgres. The main area is divided into two tabs: 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query:

```
1 SELECT title, last_name, first_name
2 FROM employee
3 ORDER BY levels DESC
4 LIMIT 1
5
```

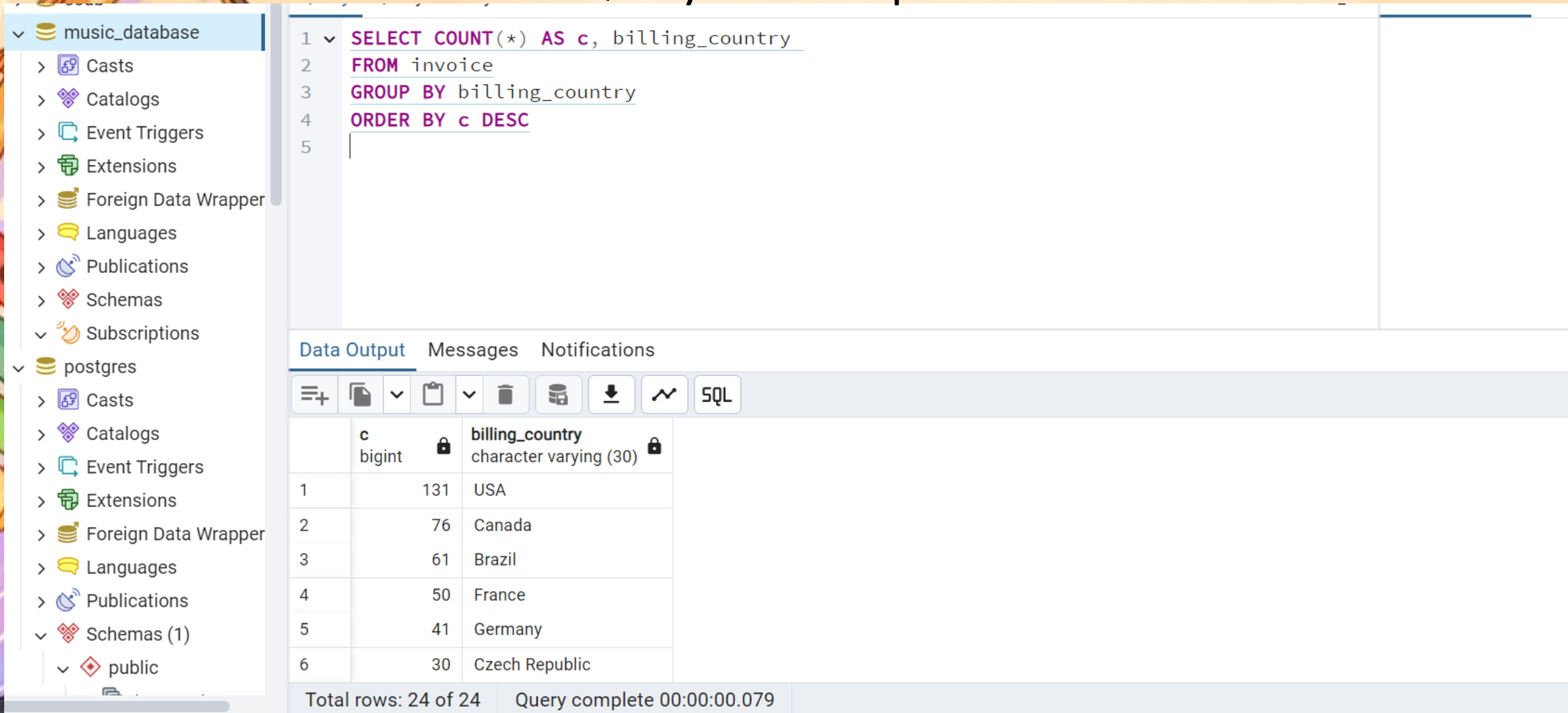
Below the query editor is a toolbar with icons for query execution, saving, and other functions. The 'Data Output' tab is active, showing the results of the query in a table format:

	title character varying (50)	last_name character	first_name character
1	Senior General Manager	Madan	Mohan

QUESTION 2:

Which countries have the most Invoices?

Query And Output:



The screenshot shows a database management interface with a sidebar on the left containing a tree view of the database structure. The main area displays a SQL query and its results.

Database Structure (Left Sidebar):

- music_database
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrapper
 - Languages
 - Publications
 - Schemas
 - Subscriptions
- postgres
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrapper
 - Languages
 - Publications
 - Schemas (1)
 - public

SQL Query:

```
1 SELECT COUNT(*) AS c, billing_country
2 FROM invoice
3 GROUP BY billing_country
4 ORDER BY c DESC
5
```

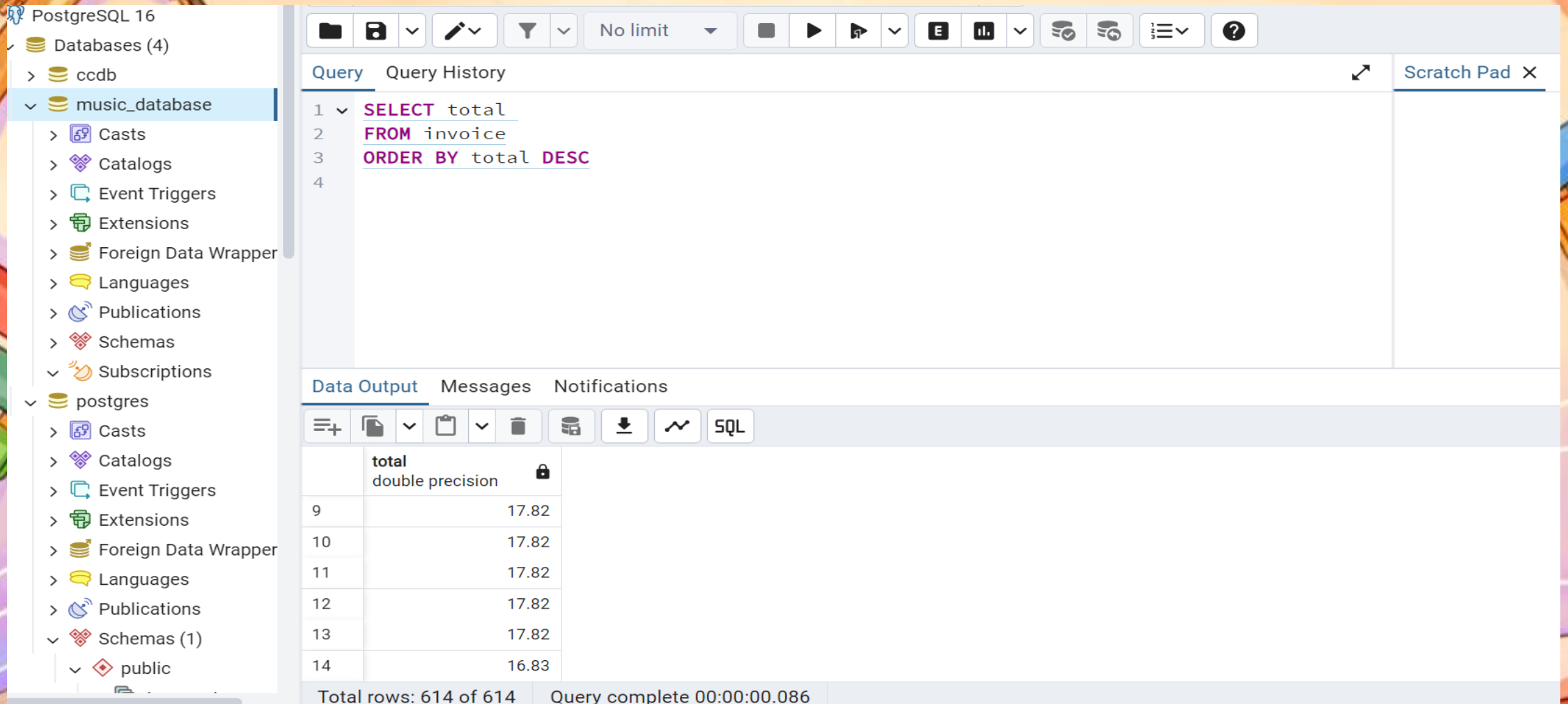
Data Output:

	c	billing_country
1	131	USA
2	76	Canada
3	61	Brazil
4	50	France
5	41	Germany
6	30	Czech Republic

Footer: Total rows: 24 of 24 | Query complete 00:00:00.079

QUESTION 3: What are top 3 values of total invoice?

Query And Output:



The screenshot shows the PostgreSQL 16 interface. On the left, the 'Databases (4)' list includes 'ccdb', 'music_database', and 'postgres'. The 'music_database' is selected, showing its contents: 'Casts', 'Catalogs', 'Event Triggers', 'Extensions', 'Foreign Data Wrapper', 'Languages', 'Publications', 'Schemas', and 'Subscriptions'. The 'postgres' database is also expanded, showing 'Casts', 'Catalogs', 'Event Triggers', 'Extensions', 'Foreign Data Wrapper', 'Languages', 'Publications', 'Schemas (1)', and 'public'.

The main area displays a SQL query in the 'Query' tab:

```
1 SELECT total
2 FROM invoice
3 ORDER BY total DESC
4
```

The 'Data Output' tab shows the results of the query. The output is a table with two columns: 'total' (double precision) and a lock icon. The results are sorted in descending order of total value.

	total	
	double precision	
9	17.82	
10	17.82	
11	17.82	
12	17.82	
13	17.82	
14	16.83	

The status bar at the bottom indicates 'Total rows: 614 of 614' and 'Query complete 00:00:00.086'.

QUESTION 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 totals

Query And Output:

The screenshot shows a database management interface with a sidebar on the left listing databases and schemas. The main area displays a SQL query and its output.

Query:

```
SELECT billing_city, SUM(total) AS InvoiceTotal
FROM invoice
GROUP BY billing_city
ORDER BY InvoiceTotal DESC
LIMIT 1;
```

Data Output:

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

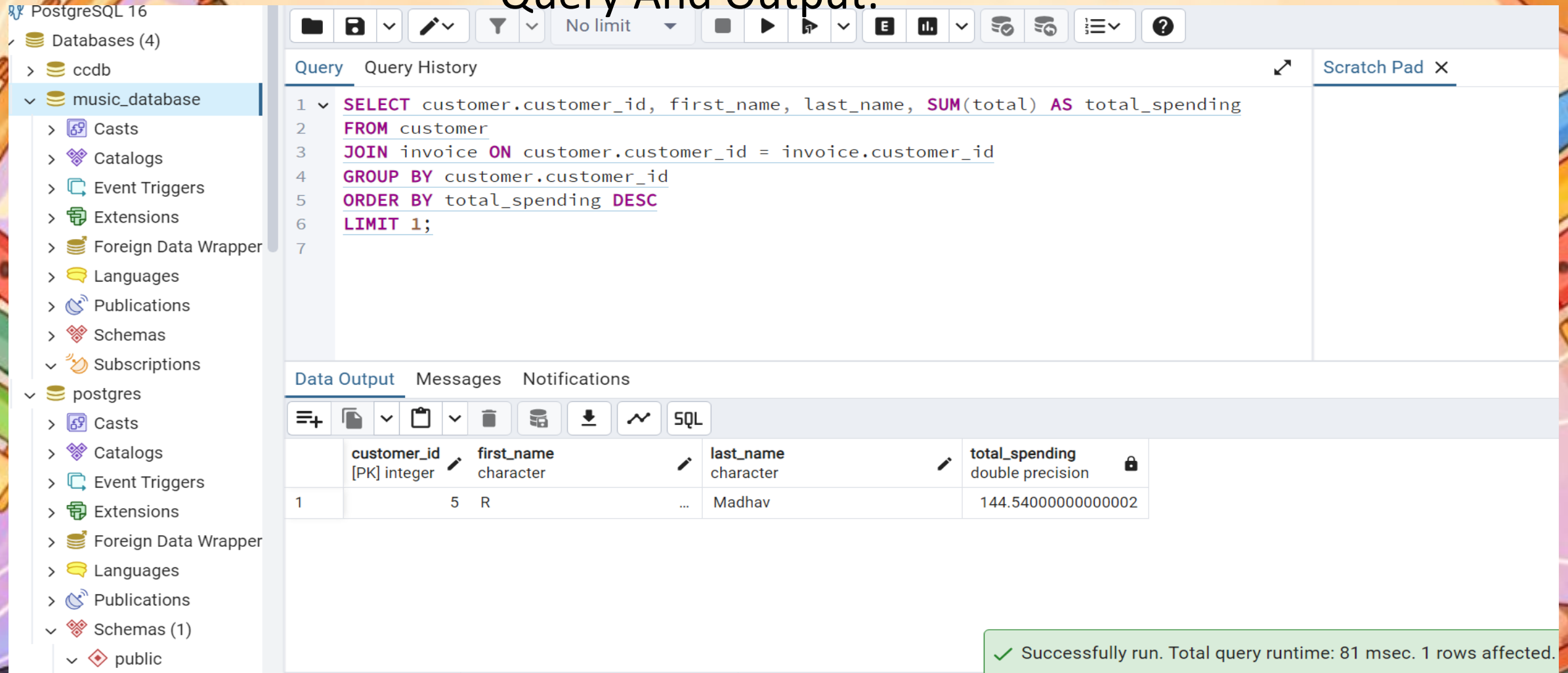
Total rows: 1 of 1 Query complete 00:00:00.109

QUESTION 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.

Query And Output:



The screenshot shows the PostgreSQL 16 interface. On the left, the 'Databases (4)' sidebar is expanded to show 'music_database'. The main query editor displays the following SQL query:

```
1 SELECT customer.customer_id, first_name, last_name, SUM(total) AS total_spending
2 FROM customer
3 JOIN invoice ON customer.customer_id = invoice.customer_id
4 GROUP BY customer.customer_id
5 ORDER BY total_spending DESC
6 LIMIT 1;
7
```

Below the query editor, the 'Data Output' tab is active, showing a table with the query results:

	customer_id [PK] integer	first_name character	last_name character	total_spending double precision
1	5	R	Madhav	144.54000000000002

At the bottom right, a green status bar indicates: ✓ Successfully run. Total query runtime: 81 msec. 1 rows affected.

QUESTION 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.

Query And Output:

The screenshot shows a database management interface. On the left is a sidebar with a tree view of database objects: FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, and Tables (11). The 'Tables (11)' folder is expanded, showing a list of tables: album, artist, customer, employee, genre, invoice, invoice_line, media_type, playlist, playlist_track, and track. The main area is divided into two panes. The top pane, titled 'Query', contains an SQL query:

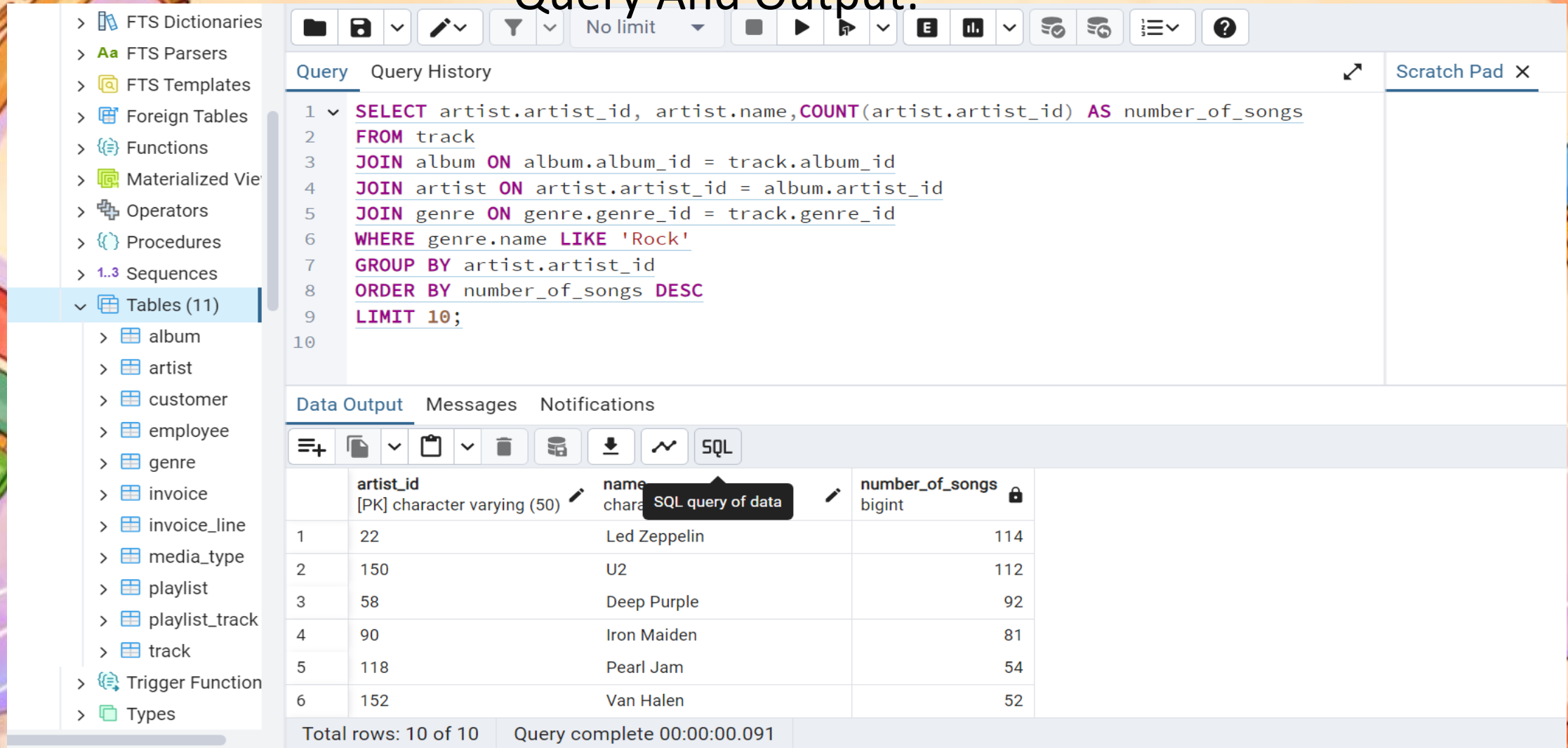
```
1 SELECT DISTINCT email, first_name, last_name
2 FROM customer
3 JOIN invoice ON customer.customer_id = invoice.customer_id
4 JOIN invoice_line ON invoice.invoice_id = invoice_line.invoice_id
5 WHERE track_id IN(
6     SELECT track_id FROM track
7     JOIN genre ON track.genre_id = genre.genre_id
8     WHERE genre.name LIKE 'Rock'
9 )
10 ORDER BY email;
```

 The bottom pane, titled 'Data Output', shows the results of the query in a table. The table has four columns: email, first_name, last_name, and an unlabeled column. The data is ordered alphabetically by email. The first six rows are visible.

	email character varying (50)	first_name character	last_name character	
1	aaronmitchell@yahoo.ca	Aaron	Mitchell	...
2	alero@uol.com.br	Alexandre	Rocha	...
3	astrid.gruber@apple.at	Astrid	Gruber	...
4	bjorn.hansen@yahoo.no	Bjørn	Hansen	...
5	camille.bernard@yahoo.fr	Camille	Bernard	...
6	daan_peeters@apple.be	Daan	Peeters	...

QUESTION 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.

Query And Output:



The screenshot shows a database management interface. On the left is a sidebar with a tree view of database objects: FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, and Tables (11). The 'Tables (11)' folder is expanded, showing tables: album, artist, customer, employee, genre, invoice, invoice_line, media_type, playlist, playlist_track, and track.

The main area has a 'Query' tab selected. It contains an SQL query to select the top 10 rock artists by the number of songs. The query is as follows:

```
1 SELECT artist.artist_id, artist.name, COUNT(artist.artist_id) AS number_of_songs
2 FROM track
3 JOIN album ON album.album_id = track.album_id
4 JOIN artist ON artist.artist_id = album.artist_id
5 JOIN genre ON genre.genre_id = track.genre_id
6 WHERE genre.name LIKE 'Rock'
7 GROUP BY artist.artist_id
8 ORDER BY number_of_songs DESC
9 LIMIT 10;
```

Below the query editor is the 'Data Output' tab, which displays the results of the query in a table. The table has three columns: artist_id, name, and number_of_songs. The results show the top 6 artists (out of 10 total rows).

	artist_id [PK] character varying (50)	name character varying (100)	number_of_songs bigint
1	22	Led Zeppelin	114
2	150	U2	112
3	58	Deep Purple	92
4	90	Iron Maiden	81
5	118	Pearl Jam	54
6	152	Van Halen	52

At the bottom of the interface, it shows 'Total rows: 10 of 10' and 'Query complete 00:00:00.091'.

QUESTION 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.

Query And Output:

The screenshot shows a database management interface with a sidebar on the left containing a tree view of database objects: FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, and Tables (11). The 'Tables (11)' folder is expanded, showing a list of tables: album, artist, customer, employee, genre, invoice, invoice_line, media_type, playlist, playlist_track, and track. The 'track' table is selected.

The main area displays an SQL query in the 'Query' tab:

```
1 SELECT name, milliseconds
2 FROM track
3 WHERE milliseconds > (
4     SELECT AVG(milliseconds) AS avg_track_length
5     FROM track )
6 ORDER BY milliseconds DESC;
```

Below the query editor, the 'Data Output' tab shows the results of the query. The results are displayed in a table with two columns: 'name' (character varying (150)) and 'milliseconds' (integer). The results are ordered by milliseconds in descending order.

	name	milliseconds
1	Occupation / Precipice	5286953
2	Through a Looking Glass	5088838
3	Greetings from Earth, Pt. 1	2960293
4	The Man With Nine Lives	2956998
5	Battlestar Galactica, Pt. 2	2956081
6	Battlestar Galactica, Pt. 1	2952702
7	Murder On the Rising Star	2935894
8	Battlestar Galactica, Pt. 3	2927802
9	Take the Celestra	2927677
10	Fire In Grass	2826502

The background of the slide is a light orange color with various musical instruments and notes scattered around. In the top left, there is a violin. To its right is a trumpet. On the left side, there is a saxophone. In the bottom left, there is a xylophone. In the bottom right, there is a drum with two drumsticks. There are also several musical notes floating around. The text is centered on the slide.

QUESTION 9:

Find how much amount spent by each customer on artists? Write a query to return customer name, artist name and total spent

Steps to Solve:

First, find which artist has earned the most according to the InvoiceLines. Now use this artist to find which customer spent the most on this artist. For this query, we need to use the Invoice, Invoice_Line, Track, Customer, Album, and Artist tables.

We need to use the Invoice_Line table to find out how many of each product was purchased, and then multiply this by the price for each artist

Query:

The screenshot shows a database query editor interface. On the left is a sidebar with a tree view of database objects: FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, and Tables (11). The 'Tables' folder is expanded, showing tables like album, artist, customer, employee, genre, invoice, invoice_line, media_type, playlist, playlist_track, and track. The main editor displays an SQL query with line numbers 1 through 20. A tooltip 'Execute script' with 'F5' is visible over the query text. The bottom of the interface includes tabs for 'Query', 'Query History', 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with columns 'customer_id', 'first_name', and 'last_name'. A status bar at the bottom indicates 'Total rows: 43 of 43' and 'Query complete 00:00:00.076'. A green notification box on the right says 'Successfully run. Total query runtime: 76 msec. 43 rows affected'.

```
1 WITH best_selling_artist AS (  
2     SELECT artist.artist_id AS artist_id, artist.name AS artist_name, SUM(invoice_li  
3     FROM invoice_line  
4     JOIN track ON track.track_id = invoice_line.track_id  
5     JOIN album ON album.album_id = track.album_id  
6     JOIN artist ON artist.artist_id = album.artist_id  
7     GROUP BY 1  
8     ORDER BY 3 DESC  
9     LIMIT 1  
10 )  
11 SELECT c.customer_id, c.first_name, c.last_name, bsa.artist_name, SUM(il.unit_price*  
12 FROM invoice i  
13 JOIN customer c ON c.customer_id = i.customer_id  
14 JOIN invoice_line il ON il.invoice_id = i.invoice_id  
15 JOIN track t ON t.track_id = il.track_id  
16 JOIN album alb ON alb.album_id = t.album_id  
17 JOIN best_selling_artist bsa ON bsa.artist_id = alb.artist_id  
18 GROUP BY 1,2,3,4  
19 ORDER BY 5 DESC;  
20
```

Successfully run. Total query runtime: 76 msec. 43 rows affected

Total rows: 43 of 43 Query complete 00:00:00.076

Output:

> Aa FTS Parsers

> FTS Templates

> Foreign Tables

> Functions

> Materialized View

> Operators

> Procedures

> 1..3 Sequences

Tables (11)

album

artist

customer

employee

genre

invoice

invoice_line

media_type

playlist

playlist_track

track

Trigger Function

Types

Data OutputMessagesNotifications

≡+

📄

▼

📋

▼

🗑

🗄

⬇

📈

SQL

	customer_id integer	first_name character	last_name character	artist_name character varying (120)	amount_spent double precision
1	46	Hugh	O'Reilly	Queen	27.719999999999985
2	38	Niklas	Schröder	Queen	18.81
3	3	François	Tremblay	Queen	17.82
4	34	João	Fernandes	Queen	16.830000000000002
5	53	Phil	Hughes	Queen	11.88
6	41	Marc	Dubois	Queen	11.88
7	47	Lucas	Mancini	Queen	10.89
8	33	Ellie	Sullivan	Queen	10.89
9	20	Dan	Miller	Queen	3.96
10	5	R	Madhav	Queen	3.96
11	23	John	Gordon	Queen	2.9699999999999998
12	54	Steve	Murray	Queen	2.9699999999999998
13	31	Martha	Silk	Queen	2.9699999999999998
14	16	Frank	Harris	Queen	1.98
15	17	Jack	Smith	Queen	1.98
16	24	Frank	Ralston	Queen	1.98

Total rows: 43 of 43Query complete 00:00:00.076Ln 1, Col 30



QUESTION 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.

Steps to Solve:

There are two parts in question-

First :most popular music genre

Second :need data at country level.

Query:

Query Query History Execute script F5

```

1 WITH popular_genre AS
2 (
3     SELECT COUNT(invoice_line.quantity) AS purchases, customer.country, genre.name,
4           ROW_NUMBER() OVER(PARTITION BY customer.country ORDER BY COUNT(invoice_line.quantity)) AS rowno
5     FROM invoice_line
6     JOIN invoice ON invoice.invoice_id = invoice_line.invoice_id
7     JOIN customer ON customer.customer_id = invoice.customer_id
8     JOIN track ON track.track_id = invoice_line.track_id
9     JOIN genre ON genre.genre_id = track.genre_id
10    GROUP BY 2,3,4
11    ORDER BY 2 ASC, 1 DESC
12 )
13 SELECT * FROM popular_genre WHERE RowNo <= 1
14 
```

Data Output Messages Notifications

	purchases bigint	country character varying (50)	name character varying (120)	genre_id character varying (50)	rowno bigint
1	17	Argentina	Alternative & Punk	4	1
2	34	Australia	Rock	1	1
3	40	Austria	Rock	1	

Total rows: 24 of 24 Query complete 00:00:00.092

✓ Successfully run. Total query runtime: 92 msec. 24 rows affected.

Output:

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized View

Operators

Procedures

1.3 Sequences

Tables (11)

album

artist

customer

employee

genre

invoice

invoice_line

media_type

playlist

playlist_track

track

Trigger Function

Types

No limit

Data Output

Messages

Notifications

</

The background of the slide is a light orange color with various musical instruments and notes scattered around. In the top left, there is a violin. To its right is a trumpet. On the left side, there is a saxophone. In the bottom left, there is a xylophone. In the bottom right, there is a drum with two drumsticks. There are also several musical notes floating around. The text is centered on the slide.

QUESTION 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.

Steps to Solve:

There are two parts in question-

First :find the most spent on music for each country.

Second : filter the data for respective customers.

Query And Output:

- > FTS Dictionaries
- > Aa FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > 1.3 Sequences
- ✓ Tables (11)
 - > album
 - > artist
 - > customer
 - > employee
 - > genre
 - > invoice
 - > invoice_line
 - > media_type
 - > playlist
 - > playlist_track
 - > track
 - > Trigger Function
 - > Types

Query Query History

```
1 WITH Customer_with_country AS (  
2     SELECT customer.customer_id,first_name,last_name,billing_country,SUM(total) AS  
3     ROW_NUMBER() OVER(PARTITION BY billing_country ORDER BY SUM(total) DESC) AS Row  
4     FROM invoice  
5     JOIN customer ON customer.customer_id = invoice.customer_id  
6     GROUP BY 1,2,3,4  
7     ORDER BY 4 ASC,5 DESC)  
8 SELECT * FROM Customer_with_country WHERE RowNo <= 1  
9
```

Data Output Messages Notifications

SQL

	customer_id integer	first_name character	last_name character	billing_country character varying (30)	total_spending double precision	rowno bigint
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	Luís	Gonçalves	Brazil	108.89999999999998	1
6	3	François	Tremblay			
7	57	Luís	Bois			

✓ Successfully run. Total query runtime: 75 msec. 24 rows affected. ✕

Total rows: 24 of 24

Query complete 00:00:00.075

Ln 9, Co