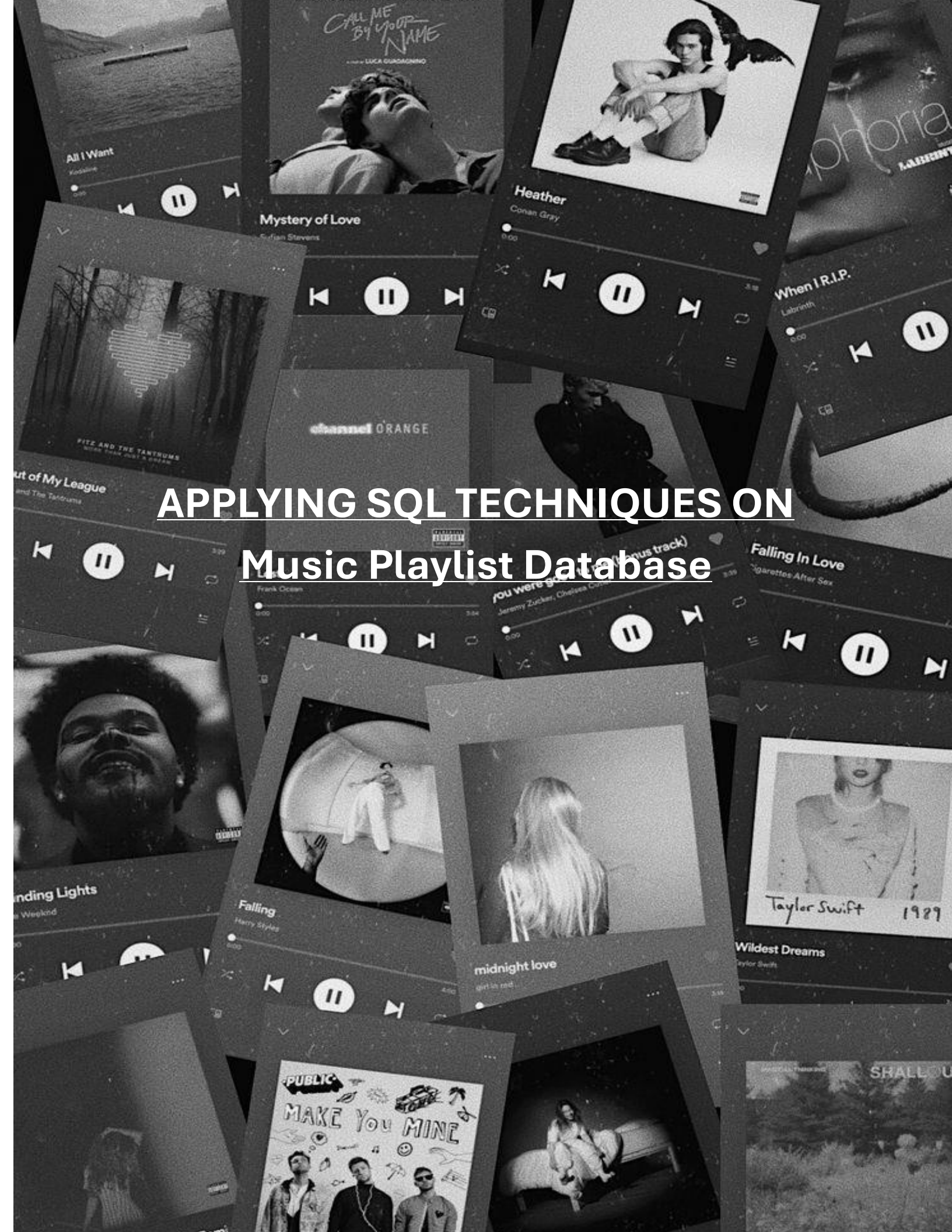
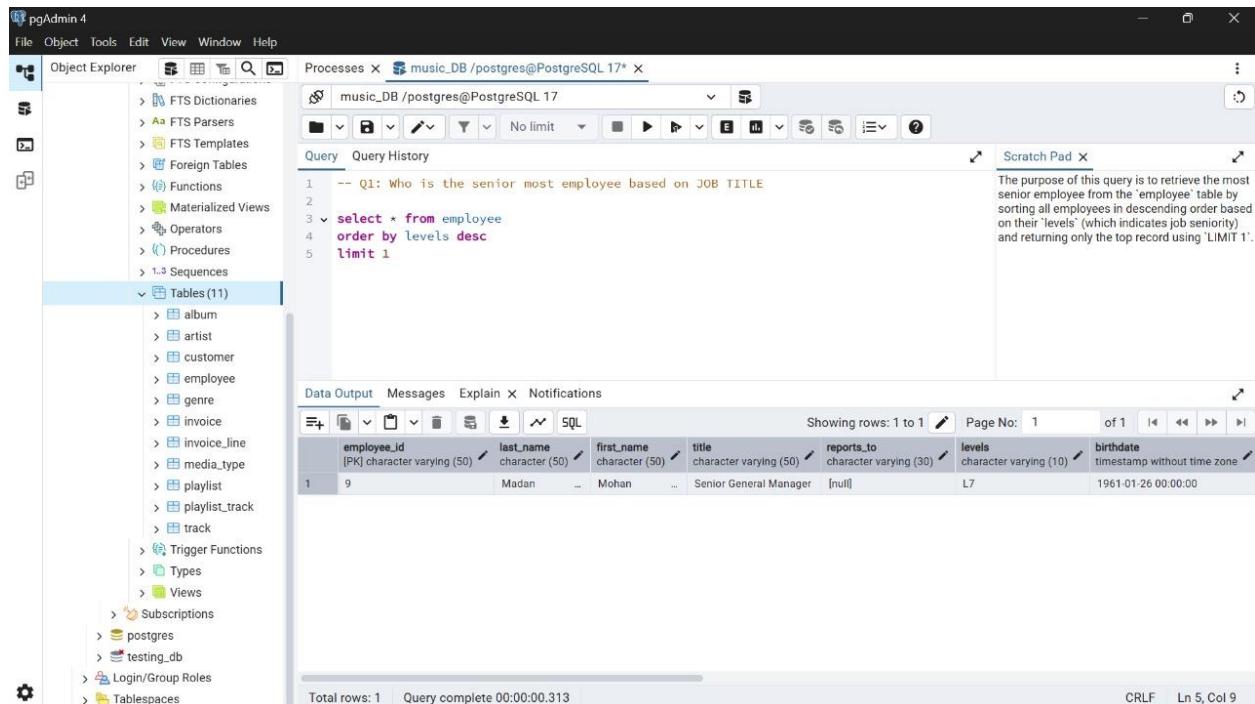


APPLYING SQL TECHNIQUES ON Music Playlist Database



Questions along with their Output, Syntax and reasoning...

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



The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer shows the database structure, with 'Tables (11)' expanded. The main pane displays a SQL query in the Query editor:

```
-- Q1: Who is the senior most employee based on JOB TITLE
1
2
3 select * from employee
4 order by levels desc
5 limit 1
```

The Data Output pane shows the result of the query as a table with 8 columns: employee_id, last_name, first_name, title, reports_to, levels, and birthdate. The output shows one row for the employee with ID 9, last name Madan, first name Mohan, title Senior General Manager, reports_to null, levels L7, and birthdate 1961-01-26 00:00:00.

employee_id	last_name	first_name	title	reports_to	levels	birthdate
9	Madan	Mohan	Senior General Manager	[null]	L7	1961-01-26 00:00:00

The status bar at the bottom indicates 'Total rows: 1' and 'Query complete 00:00:00.313'.

Explanation

- **SELECT * FROM employee**
→ Fetches all columns from the employee table.
- **ORDER BY levels DESC**
→ Sorts the records in **descending order** of the levels column.
→ Assumes that a **higher level (e.g., L7)** indicates a more senior position.
- **LIMIT 1**
→ Restricts the result to **only the top 1 record** after sorting.
→ This ensures only the **most senior employee** is shown.

2. Which countries have the most invoices?

The screenshot shows the pgAdmin 4 interface with a SQL query executed in the 'music_DB / postgres@PostgreSQL 17*' database. The query is as follows:

```
1 -- which country have most invoices
2
3 select count (*) as c, billing_country
4 from invoice
5 group by billing_country
6 order by c desc
```

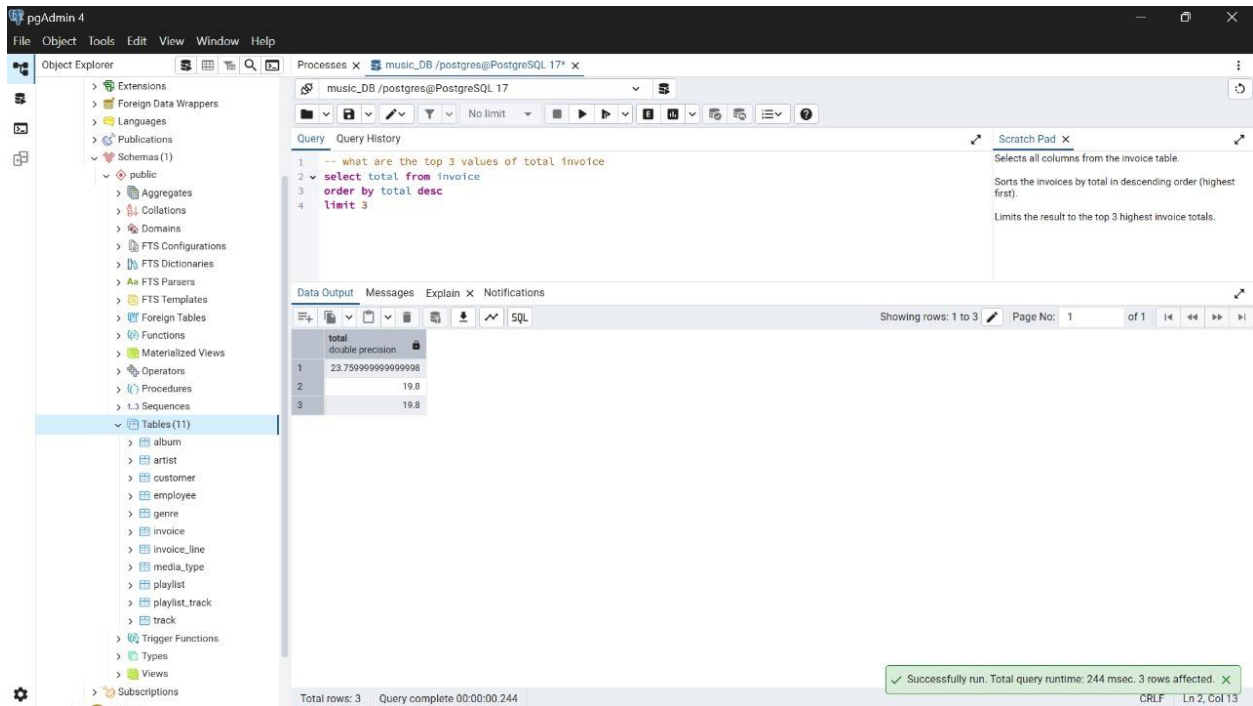
The results are displayed in the 'Data Output' tab, showing the top 16 countries by invoice count. The status bar indicates 'Total rows: 24' and 'Query complete 00:00:00.287'.

c	billing_country
131	USA
76	Canada
61	Brazil
50	France
41	Germany
30	Czech Republic
29	Portugal
28	United Kingdom
21	India
13	Chile
13	Ireland
11	Spain
11	Finland
10	Australia
10	Netherlands
10	Sweden

Output

Data Output		Messages	Explain X	Notifications
Showing rows: 1 to 24		Page No: 1	of 1	
c	billing_country			
bigint	character varying (30)			
1	131 USA			
2	76 Canada			
3	61 Brazil			
4	50 France			
5	41 Germany			
6	30 Czech Republic			
7	29 Portugal			
8	28 United Kingdom			
9	21 India			
10	13 Chile			
11	13 Ireland			
12	11 Spain			
13	11 Finland			
14	10 Australia			
15	10 Netherlands			
16	10 Sweden			
17	10 Poland			
18	10 Hungary			
19	10 Denmark			
20	9 Austria			
21	9 Norway			
22	9 Italy			
23	7 Belgium			
24	5 Argentina			
Total rows: 24		Query complete 00:00:00.287		CRLF Ln 6, Col 16

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



The screenshot shows the pgAdmin 4 interface with a SQL query executed in the 'music_DB / postgres@PostgreSQL 17' database. The query is:

```
1 -- what are the top 3 values of total invoice
2 select total from invoice
3 order by total desc
4 limit 3
```

The results are displayed in the 'Data Output' tab, showing the top 3 highest invoice totals:

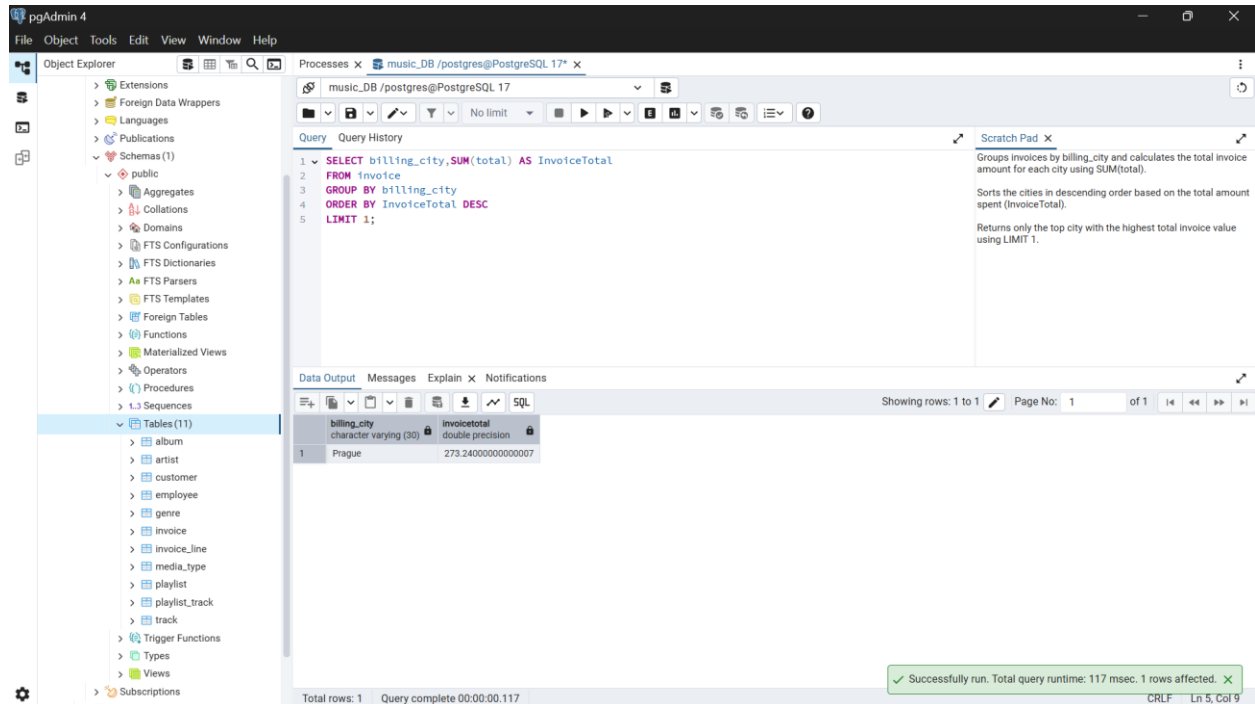
	total
1	23.759999999999998
2	19.8
3	19.8

A status message at the bottom right indicates: 'Successfully run. Total query runtime: 244 msec. 3 rows affected.'

Explanation

- **SELECT total FROM invoice**
→ Retrieves the total amount from all invoices.
- **ORDER BY total DESC**
→ Sorts the totals in **descending order** (highest to lowest).
- **LIMIT 3**
→ Returns only the **top 3 highest invoice totals**.

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.



The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer displays the database structure, with 'Tables (11)' expanded. The main query editor shows the following SQL query:

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

The 'Data Output' tab shows the results of the query:

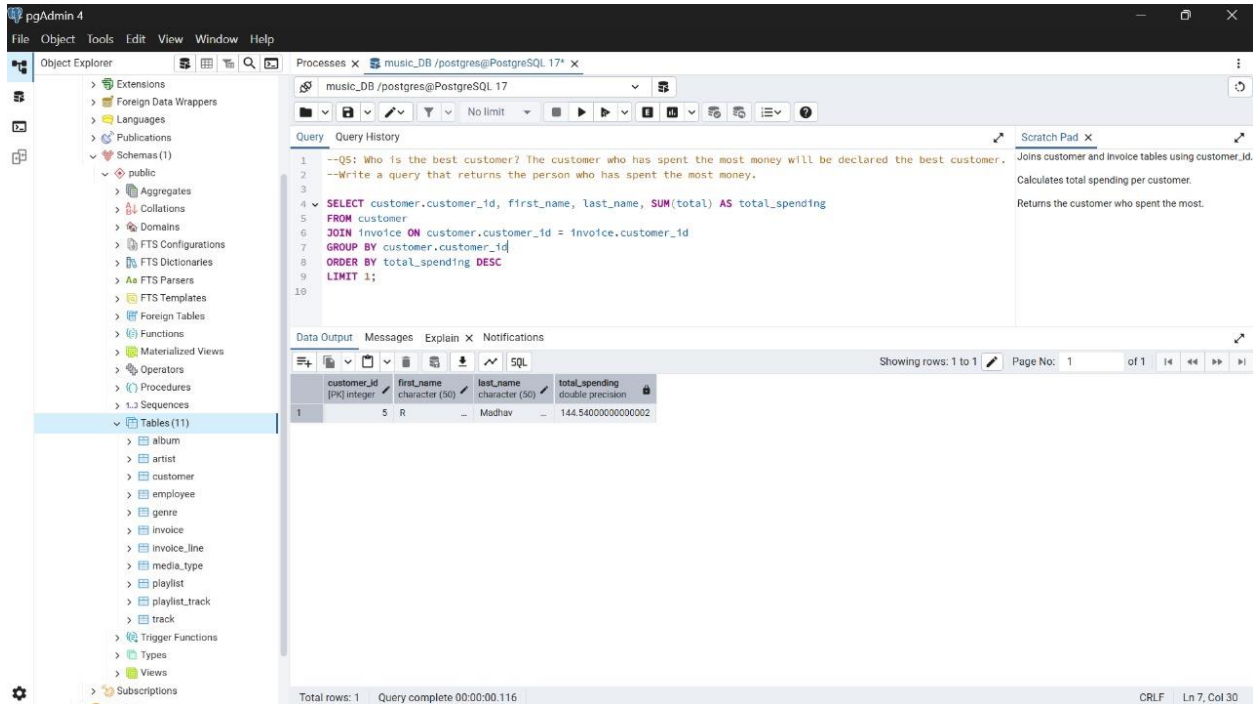
billing_city	InvoiceTotal
Prague	273.24000000000007

A status bar at the bottom indicates: 'Total rows: 1 Query complete 00:00:00.117' and a green message box says 'Successfully run. Total query runtime: 117 msec. 1 rows affected.'

Explanation

- **Groups invoices by billing_city** and calculates the **total invoice amount** for each city using SUM(total).
- **Sorts the cities** in descending order based on the total amount spent (InvoiceTotal).
- **Returns only the top city** with the highest total invoice value using LIMIT 1.

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 4 interface. On the left, the Object Explorer shows the 'public' schema with tables like 'album', 'artist', 'customer', 'employee', 'genre', 'invoice', 'invoice_line', 'media_type', 'playlist', 'playlist_track', 'track', 'trigger_functions', 'types', 'views', and 'subscriptions'. The 'customer' table is selected. The main window shows a SQL query in the 'Query' tab:

```
1 --Q5: Who is the best customer? The customer who has spent the most money will be declared the best customer.
2 --Write a query that returns the person who has spent the most money.
3
4 SELECT customer.customer_id, first_name, last_name, SUM(total) AS total_spending
5 FROM customer
6 JOIN invoice ON customer.customer_id = invoice.customer_id
7 GROUP BY customer.customer_id
8 ORDER BY total_spending DESC
9 LIMIT 1;
```

The 'Data Output' tab shows the results of the query:

customer_id	first_name	last_name	total_spending
5	R	Madhav	144.54000000000002

The status bar at the bottom indicates 'Total rows: 1' and 'Query complete 00:00:00.116'.

Explanation

- **SELECT customer.customer_id, first_name, last_name, SUM(total) AS total_spending**
→ Retrieves the customer's ID, first name, last name, and their **total spending** by summing up all invoice amounts.
- **FROM customer**
→ Data is being selected from the **customer** table.
- **JOIN invoice ON customer.customer_id = invoice.customer_id**
→ Performs an **inner join** between customer and invoice tables using the customer_id column.
→ This links each customer to their respective invoices.
- **GROUP BY customer.customer_id**
→ Groups all invoice records by customer to calculate **total spending per customer**.
- **ORDER BY total_spending DESC**
→ Sorts the results in **descending order** so the highest spender comes first.

- **LIMIT 1**

→ Returns **only the top customer** who has spent the most.

6. Write a 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 4 interface. On the left, the 'Object Explorer' shows the database structure. The main pane displays a SQL query in the 'Query' tab. The query is as follows:

```

--Q1: 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.
SELECT DISTINCT email, first_name, last_name
FROM customer
JOIN invoice ON customer.customer_id = invoice.customer_id
JOIN invoice_line ON invoice.invoice_id = invoice_line.invoice_id
WHERE track_id IN (
    SELECT track_id
    FROM track
    JOIN genre ON track.genre_id = genre.genre_id
    WHERE genre.name = 'Rock'
)
ORDER BY email;

```

The 'Data Output' tab shows the results of the query, displaying 13 rows of customer data who have purchased Rock music tracks. The columns are email, first_name, and last_name.

	email	first_name	last_name
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	Bjorn	Hansen
5	camille.bernard@yahoo.fr	Camille	Bernard
6	daan.peeters@apple.be	Daan	Peeters
7	diego.gutierrez@yahoo.ar	Diego	Gutierrez
8	dmiller@comcast.com	Dan	Miller
9	dominiquelefebvre@gmail.com	Dominique	Lefebvre
10	edfrancis@yahoo.ca	Edward	Francis
11	eduardo@woodstock.com.br	Eduardo	Martins
12	ellie.sullivan@shaw.ca	Ellie	Sullivan
13	emma.jones@hotmail.com	Emma	Jones

Total rows: 59 Query complete 00:00:00.129

Explanation

- **Joins** the track, album, artist, and genre tables to connect each track to its respective artist and genre.
- **Filters for Rock genre tracks** and **counts the number of Rock songs** per artist using COUNT().
- **Returns the top 10 artists** who have the most Rock songs, sorted in **descending order** by song count.

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 4 interface. On the left, the 'Object Explorer' pane shows the database structure, with 'Tables (11)' expanded. The 'track' table is selected. The main pane displays a SQL query in the 'Query' editor. The query is as follows:

```
-- Q2: 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.

SELECT artist.artist_id, artist.name, COUNT(artist.artist_id) AS number_of_songs
FROM track
JOIN album ON album.album_id = track.album_id
JOIN artist ON artist.artist_id = album.artist_id
JOIN genre ON genre.genre_id = track.genre_id
WHERE genre.name LIKE 'Rock'
GROUP BY artist.artist_id
ORDER BY number_of_songs DESC
LIMIT 10;
```

The 'Data Output' pane shows the results of the query, displaying 10 rows. The columns are 'artist_id', 'name', and 'number_of_songs'. The results are as follows:

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
152	Van Halen	52
51	Queen	45
142	The Rolling Stones	41
76	Credence Clearwater Revival	40
52	Kiss	35

At the bottom of the interface, a status bar indicates 'Total rows: 10' and 'Query complete 00:00:00.123'. A green message box at the bottom right states 'Successfully run. Total query runtime: 123 msec. 10 rows affected.'

- **SELECT artist.artist_id, artist.name, COUNT(artist.artist_id) AS number_of_songs**
→ Retrieves each **artist's ID, name**, and the **total number of Rock songs** associated with them.
- **FROM track**
→ Starts from the track table which contains information about each song.
- **JOIN album ON album.album_id = track.album_id**
→ Links each track to its respective album using album_id.
- **JOIN artist ON artist.artist_id = album.artist_id**
→ Links each album to its artist using artist_id.
- **JOIN genre ON genre.genre_id = track.genre_id**
→ Links each track to its genre.
- **WHERE genre.name LIKE 'Rock'**
→ Filters the tracks to include **only those with the genre 'Rock'**.
- **GROUP BY artist.artist_id**
→ Groups the data **by artist**, so that the number of Rock songs per artist can be counted.

- **ORDER BY number_of_songs DESC**
→ Sorts the result by **number of Rock songs**, highest first.
- **LIMIT 10**
→ Returns only the **top 10 artists** with the most Rock songs.