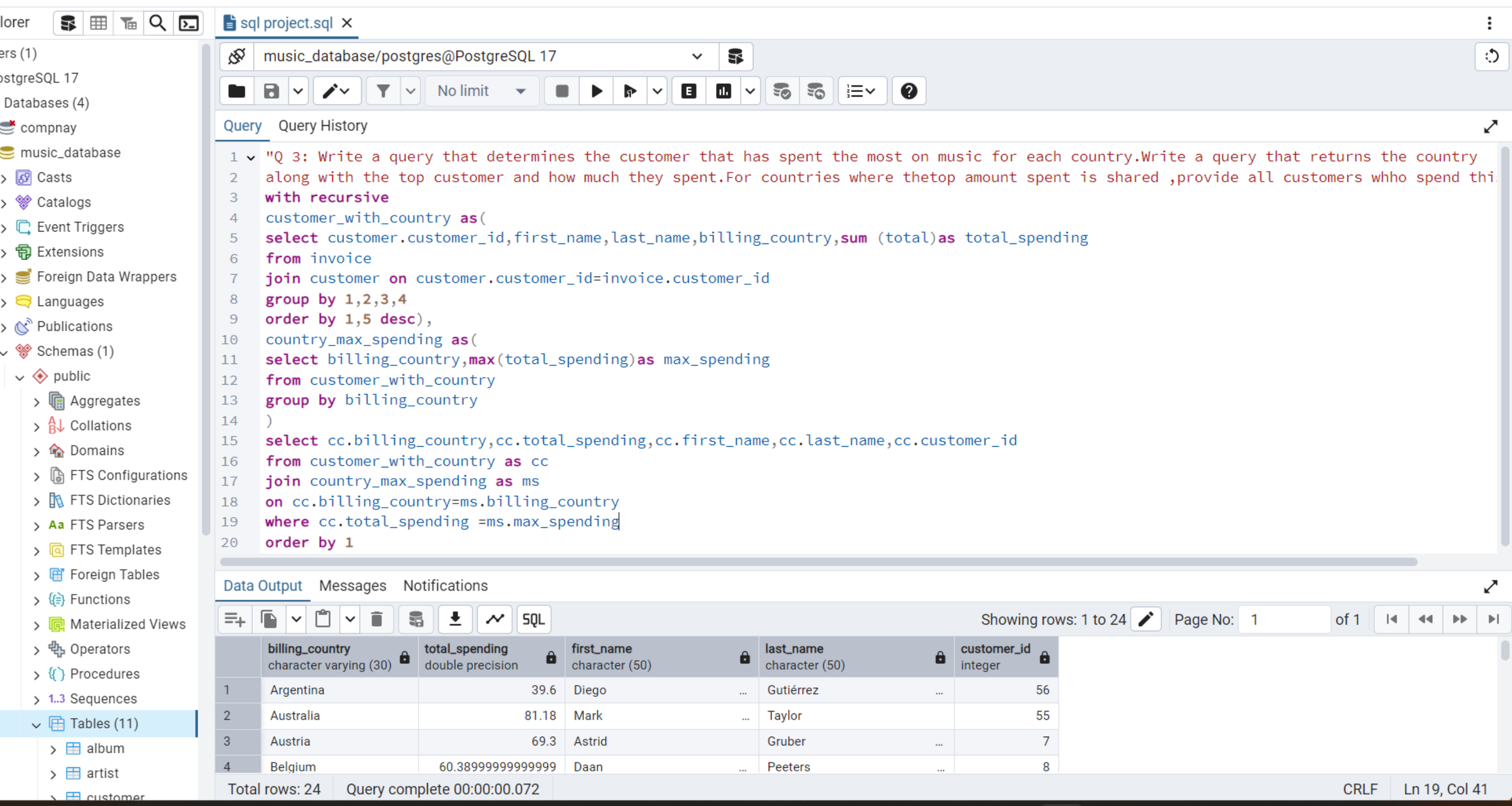
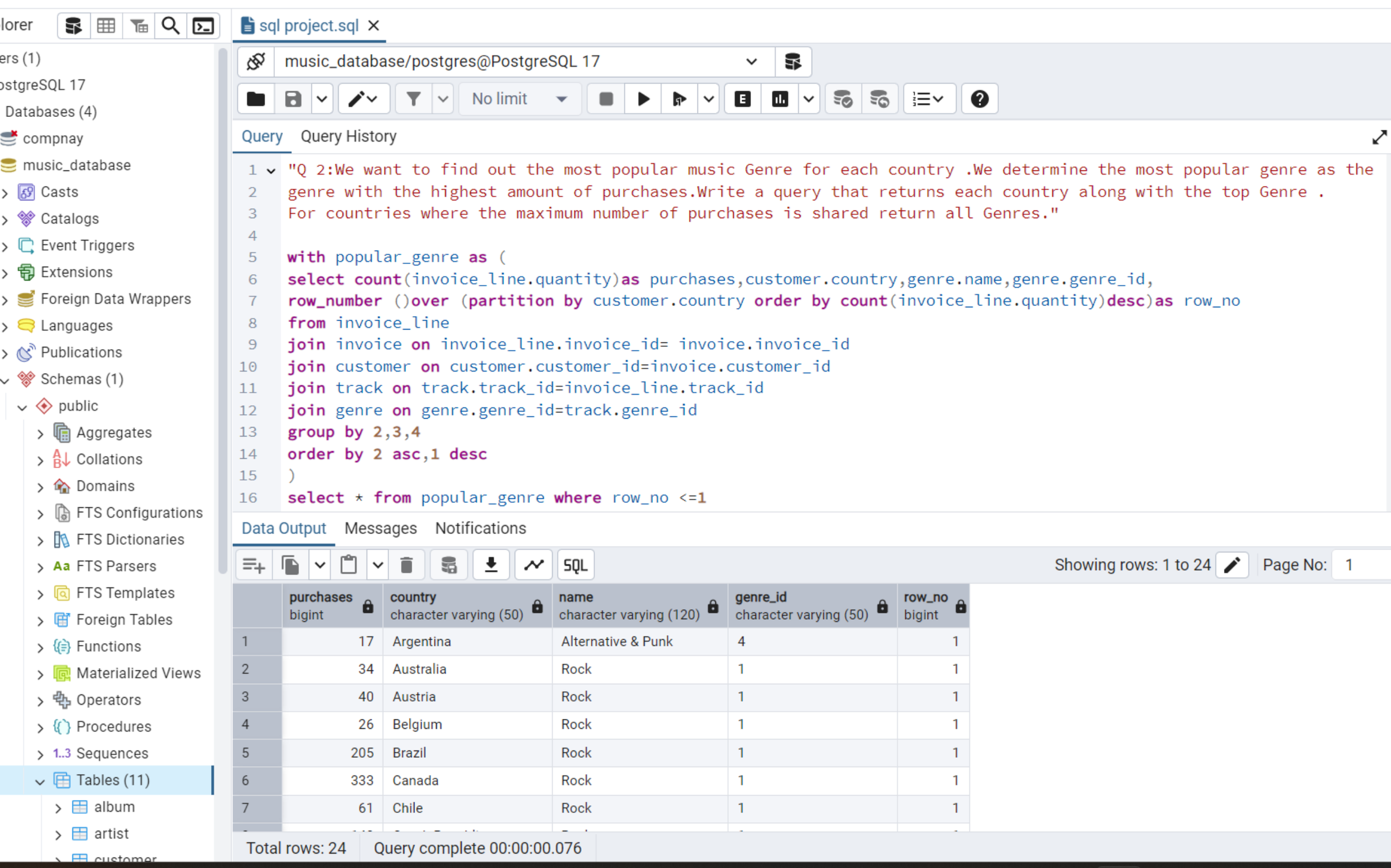
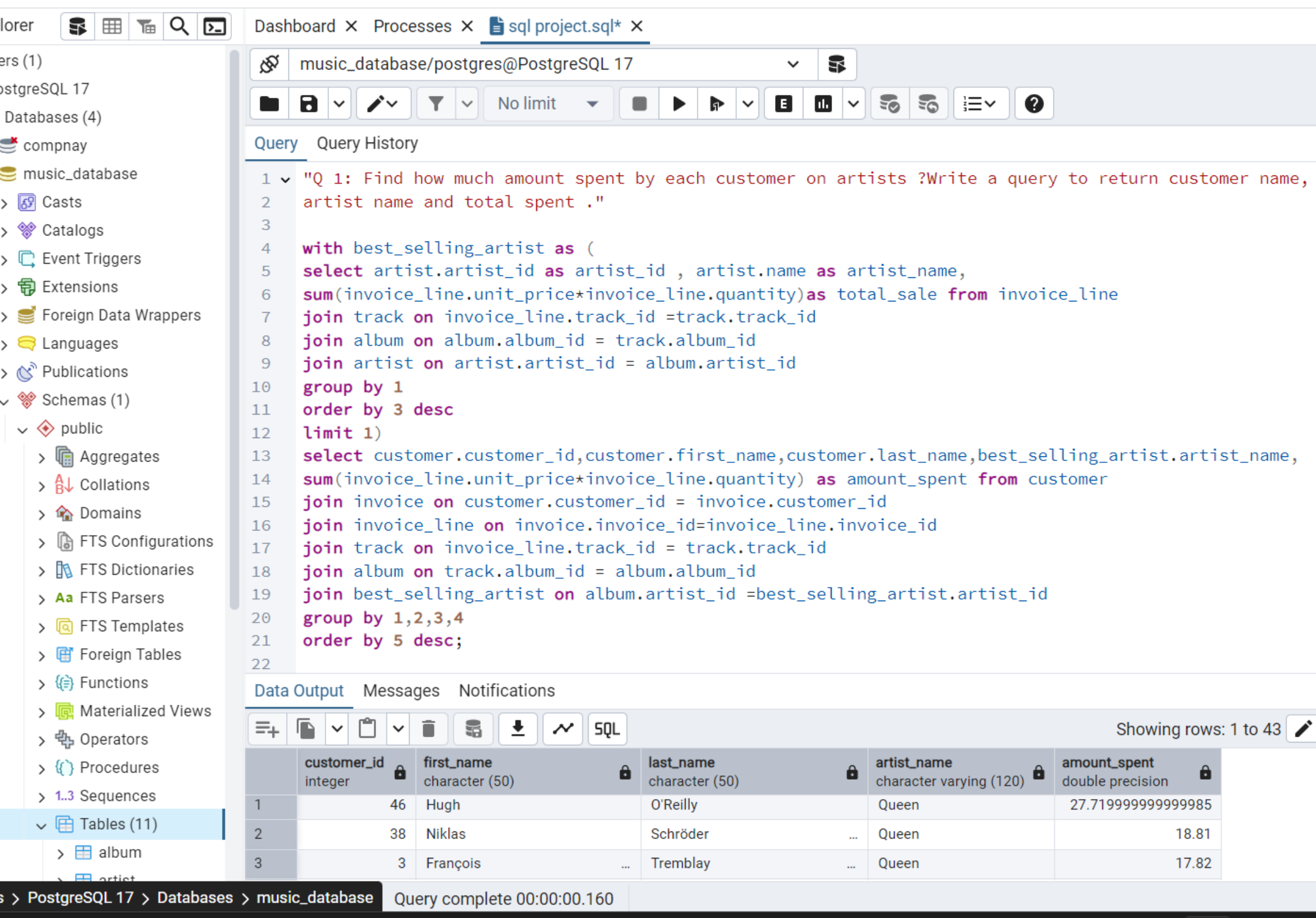

sql project.sql
music_database/postgres@PostgreSQL 17
No limit
Query
Query History
"Q 3: Write a query that determines the customer that has spent the most on music for each country.Write a query that returns the country
along with the top customer and how much they spent.For countries where thetop amount spent is shared ,provide all customers whho spend thi
with recursive
customer_with_country as(
select customer.customer_id,first_name,last_name,billing_country,sum (total)as total_spending
from invoice
join customer on customer.customer_id=invoice.customer_id
group by 1,2,3,4
order by 1,5 desc),
country_max_spending as(
select billing_country,max(total_spending)as max_spending
from customer_with_country
group by billing_country
)
select cc.billing_country,cc.total_spending,cc.first_name,cc.last_name,cc.customer_id
from customer_with_country as cc
join country_max_spending as ms
on cc.billing_country=ms.billing_country
where cc.total_spending =ms.max_spending
order by 1
Data Output
Messages
Notifications
SQL
Showing rows: 1 to 24
Page No: 1
of 1
billing_country character varying (30)
total_spending double precision
first_name character (50)
last_name character (50)
customer_id integer
1 Argentina 39.6 Diego Gutiérrez 56
2 Australia 81.18 Mark Taylor 55
3 Austria 69.3 Astrid Gruber 7
4 Belgium 60.38999999999999 Daan Peeters 8
Total rows: 24
Query complete 00:00:00.072
CRLF
Ln 19, Col 41
Databases (4)
compnay
music_database
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (11)
album
artist


sql project.sql
music_database/postgres@PostgreSQL 17
No limit
Query
Query History
"Q 2: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."
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 row_no
from invoice_line
join invoice on invoice_line.invoice_id= invoice.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 row_no <=1
Data Output
Messages
Notifications
SQL
Showing rows: 1 to 24
Page No: 1
purchases bigint
country character varying (50)
name character varying (120)
genre_id character varying (50)
row_no bigint
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
6 333 Canada Rock 1 1
7 61 Chile Rock 1 1
Total rows: 24
Query complete 00:00:00.076
Databases (4)
compnay
music_database
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (11)
album
artist


Dashboard
Processes
sql project.sql*
music_database/postgres@PostgreSQL 17
No limit
Query
Query History
"Q 1: Find how much amount spent by each customer on artists ?Write a query to return customer name, artist name and total spent ."
with best_selling_artist as (
select artist.artist_id as artist_id , artist.name as artist_name,
sum(invoice_line.unit_price*invoice_line.quantity)as total_sale from invoice_line
join track on invoice_line.track_id =track.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.customer_id,customer.first_name,customer.last_name,best_selling_artist.artist_name,
sum(invoice_line.unit_price*invoice_line.quantity) as amount_spent from customer
join invoice on customer.customer_id = invoice.customer_id
join invoice_line on invoice.invoice_id=invoice_line.invoice_id
join track on invoice_line.track_id = track.track_id
join album on track.album_id = album.album_id
join best_selling_artist on album.artist_id =best_selling_artist.artist_id
group by 1,2,3,4
order by 5 desc;
Data Output
Messages
Notifications
SQL
Showing rows: 1 to 43
customer_id integer
first_name character (50)
last_name character (50)
artist_name character varying (120)
amount_spent double precision
1 46 Hugh O'Reilly Queen 27.71999999999985
2 38 Niklas Schröder Queen 18.81
3 3 François Tremblay Queen 17.82
compnay
music_database
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (11)
album
PostgreSQL 17 > Databases > music_database
Query complete 00:00:00.160

music_database/postgres@PostgreSQL 17

No limit

Query Query History

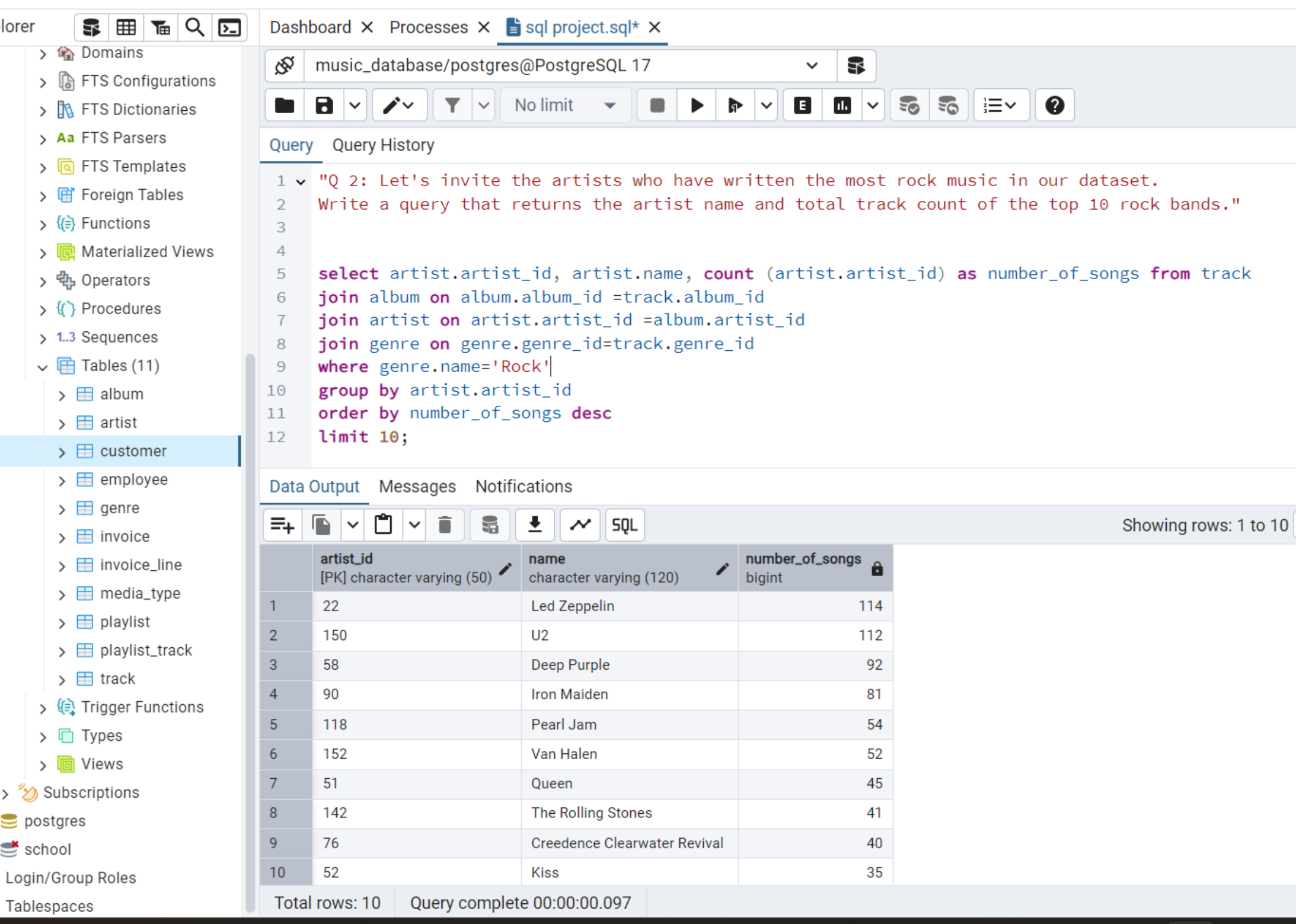
```
1  "Q 3: Return all the track names that have a song length longer than the average song lenth.Return the name and
2  milliseconds for each track .Order by the song length with the longest songs listed first."
3
4  select milliseconds , name
5  from track
6  where milliseconds>(select avg(milliseconds)as average_milliseconds
7  from track)
8  order by milliseconds desc
```

Data Output Messages Notifications

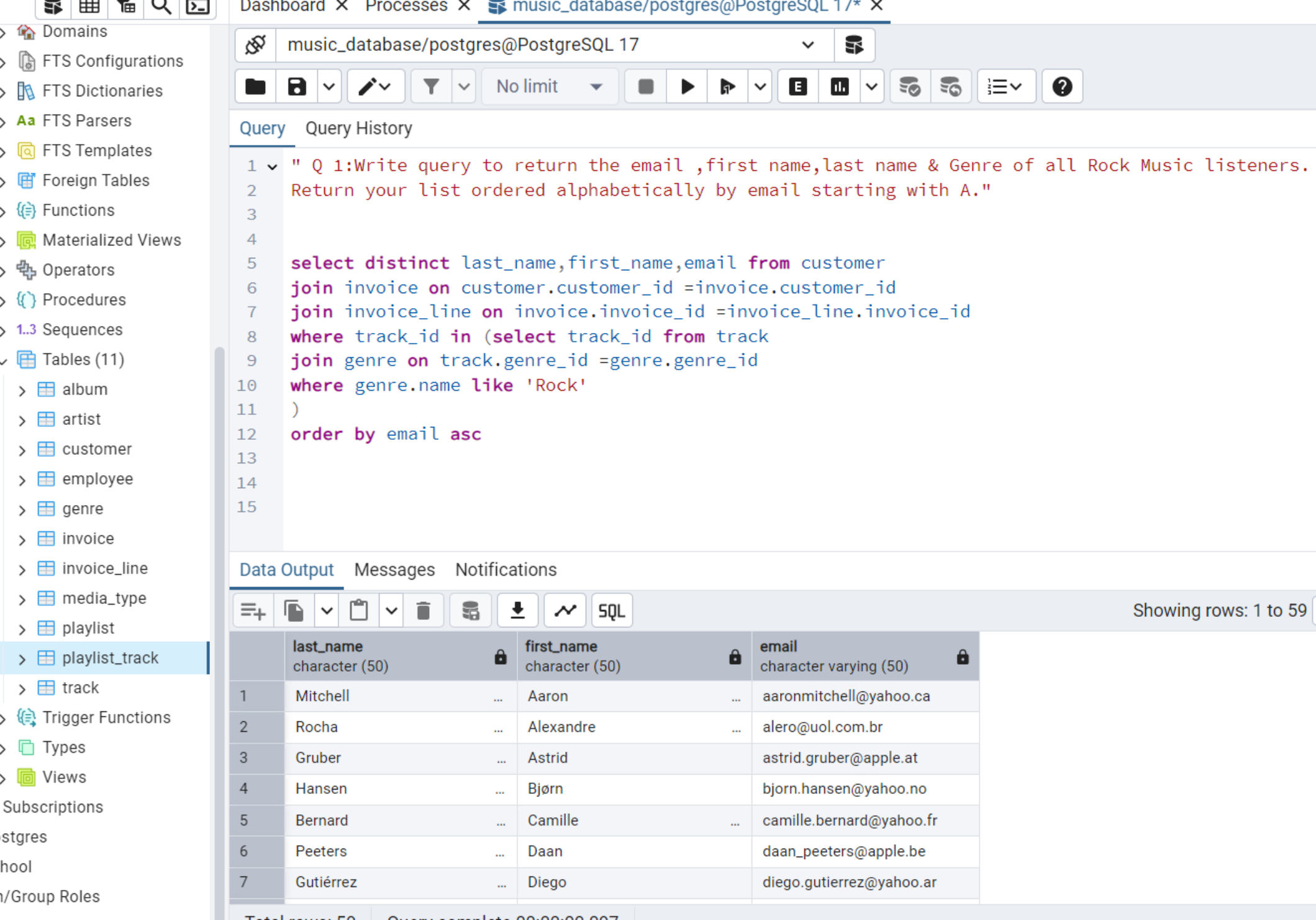
Showing rows: 1 to 494 Page No: 1

	milliseconds integer	name character varying (150)
1	5286953	Occupation / Precipice
2	5088838	Through a Looking Glass
3	2960293	Greetings from Earth, Pt. 1
4	2956998	The Man With Nine Lives
5	2956081	Battlestar Galactica, Pt. 2
6	2952702	Battlestar Galactica, Pt. 1
7	2935894	Murder On the Rising Star
8	2927802	Battlestar Galactica, Pt. 3
9	2927677	Take the Celestra
10	2926593	Fire In Space

Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (11)
album
artist
customer
employee
genre
invoice
invoice_line
media_type
playlist
playlist_track
track
Trigger Functions
Types
Views
Subscriptions
stgres
hool
/Group Roles



	artist_id [PK] character varying (50)	name character varying (120)	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
7	51	Queen	45
8	142	The Rolling Stones	41
9	76	Creedence Clearwater Revival	40
10	52	Kiss	35


Dashboard
Processes
music_database/postgres@PostgreSQL 17*
music_database/postgres@PostgreSQL 17
No limit
Query
Query History
" Q 1: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 last_name,first_name,email 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 like 'Rock'
)
order by email asc
Data Output
Messages
Notifications
SQL
Showing rows: 1 to 59
last_name character (50)
first_name character (50)
email character varying (50)
1 Mitchell Aaron aaronmitchell@yahoo.ca
2 Rocha Alexandre alero@uol.com.br
3 Gruber Astrid astrid.gruber@apple.at
4 Hansen Bjørn bjorn.hansen@yahoo.no
5 Bernard Camille camille.bernard@yahoo.fr
6 Peeters Daan daan_peeters@apple.be
7 Gutiérrez Diego diego.gutierrez@yahoo.ar
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (11)
album
artist
customer
employee
genre
invoice
invoice_line
media_type
playlist
playlist_track
track
Trigger Functions
Types
Views
Subscriptions

music_database/postgres@PostgreSQL 17

No limit

Query Query History

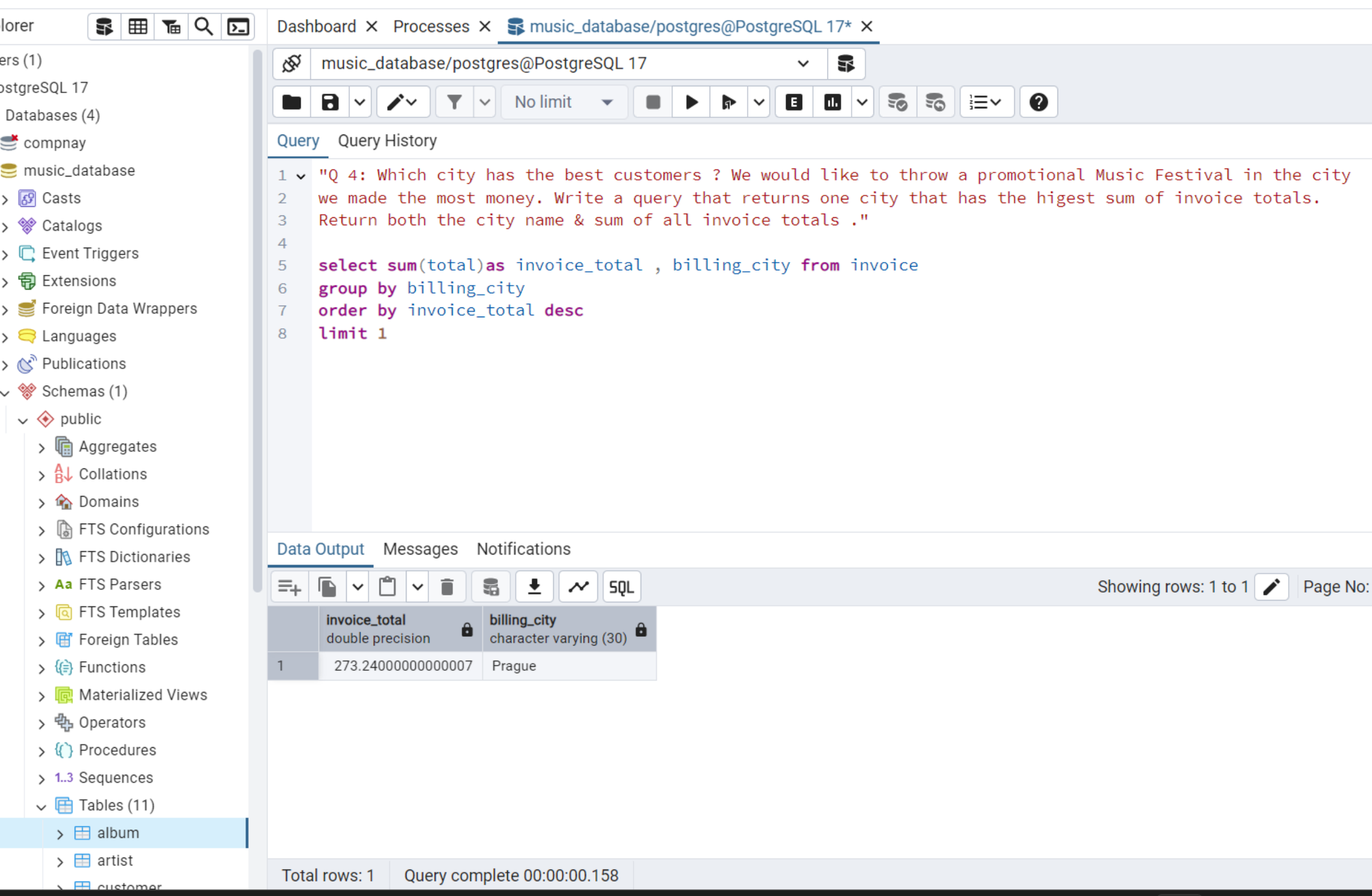
```
1  "Q 5: Who is the best customer ? The cussstomer who has spent the most money will be declared the best customer.
2  Write a query that returns the person who has spent the most money ."
3
4  select sum(invoice.total)as total,customer.customer_id,customer.first_name,customer. last_name from customer
5  join invoice on
6  customer.customer_id =invoice.customer_id
7  group by customer.customer_id
8  order by total desc
9  limit 1
```

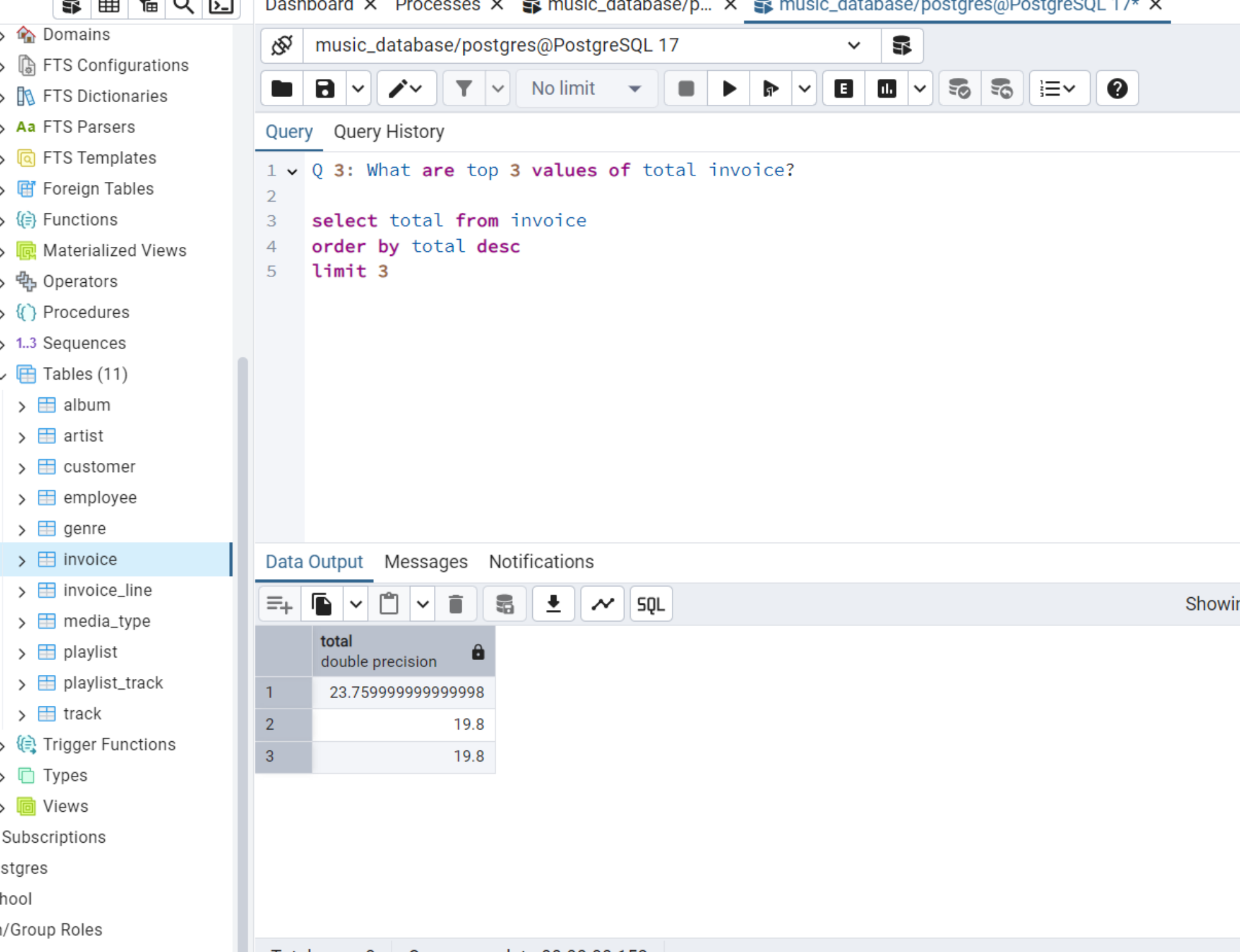
Data Output Messages Notifications

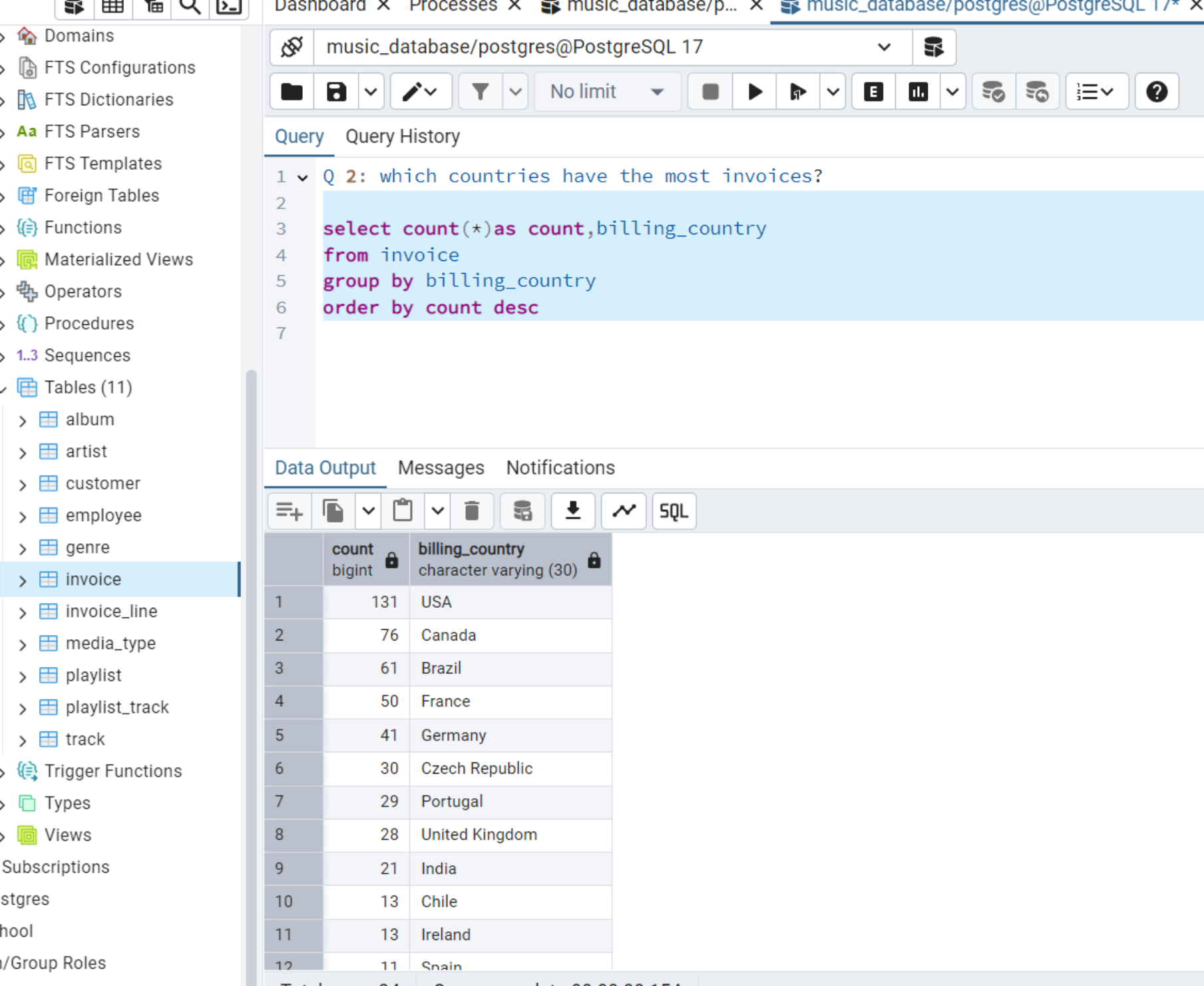
Showing rows: 1 to 1 Page No: 1

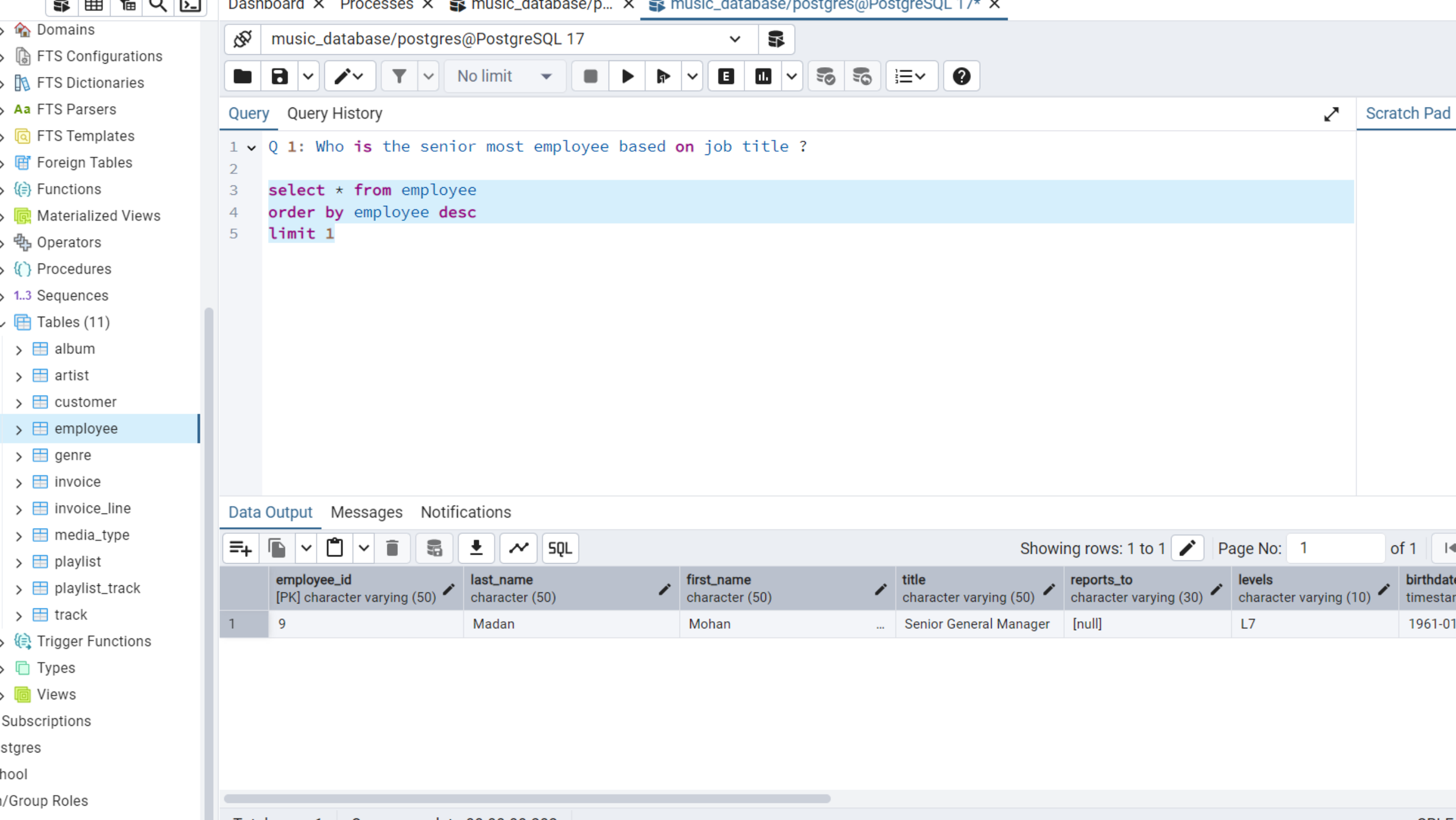
	total double precision	customer_id [PK] integer	first_name character (50)	last_name character (50)
1	144.54000000000002	5	R ...	Madhav

- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (11)
  - album
  - artist
  - customer
  - employee
  - genre
  - invoice
  - invoice_line
  - media_type
  - playlist
  - playlist_track
  - track
- Trigger Functions
- Types
- Views
- Subscriptions
- stgres
- hool
- /Group Roles


Dashboard
Processes
music_database/postgres@PostgreSQL 17*
music_database/postgres@PostgreSQL 17
No limit
Query
Query History
"Q 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 higest sum of invoice totals. Return both the city name & sum of all invoice totals ."
select sum(total)as invoice_total , billing_city from invoice
group by billing_city
order by invoice_total desc
limit 1
Data Output
Messages
Notifications
SQL
Showing rows: 1 to 1
Page No:
invoice_total
double precision
billing_city
character varying (30)
1
273.24000000000007
Prague
Total rows: 1
Query complete 00:00:00.158
Databases (4)
compnay
music_database
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (11)
album
artist


Dashboard
Processes
music_database/p...
music_database/postgres@PostgreSQL 17*
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (11)
album
artist
customer
employee
genre
invoice
invoice_line
media_type
playlist
playlist_track
track
Trigger Functions
Types
Views
Subscriptions
stgres
hool
/Group Roles
music_database/postgres@PostgreSQL 17
No limit
Query
Query History
1 Q 3: What are top 3 values of total invoice?
2
3 select total from invoice
4 order by total desc
5 limit 3
Data Output
Messages
Notifications
SQL
Showir
total
double precision
1 23.759999999999998
2 19.8
3 19.8


Dashboard
Processes
music_database/p...
music_database/postgres@PostgreSQL 17*
music_database/postgres@PostgreSQL 17
No limit
Query
Query History
Q 2: which countries have the most invoices?
select count(*)as count,billing_country
from invoice
group by billing_country
order by count desc
Data Output
Messages
Notifications
SQL
count
bigint
billing_country
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
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (11)
album
artist
customer
employee
genre
invoice
invoice_line
media_type
playlist
playlist_track
track
Trigger Functions
Types
Views
Subscriptions
stgres
hool
/Group Roles


Dashboard
Processes
music_database/p...
music_database/postgres@PostgreSQL 17*
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (11)
album
artist
customer
employee
genre
invoice
invoice_line
media_type
playlist
playlist_track
track
Trigger Functions
Types
Views
Subscriptions
stgres
hool
/Group Roles
music_database/postgres@PostgreSQL 17
No limit
Query
Query History
Scratch Pad
1 Q 1: Who is the senior most employee based on job title ?
2
3 select * from employee
4 order by employee desc
5 limit 1
Data Output
Messages
Notifications
SQL
Showing rows: 1 to 1
Page No: 1
of 1
employee_id
[PK] character varying (50)
last_name
character (50)
first_name
character (50)
title
character varying (50)
reports_to
character varying (30)
levels
character varying (10)
birthdat
timestar
1
9
Madan
Mohan
...
Senior General Manager
[null]
L7
1961-01