




Music Store Data Analysis

Data Analysis Reveals Trends in Consumer Preferences and Seasonal Peaks...

Data Output Messages Notifications 

pgAdmin 4

FileObjectToolsHelp

Object Explorer

Servers (1)

PostgreSQL 16

Databases (4)

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

customer

employee

genre

invoice

Dashboard × Properties × SQL × Statistics × Dependencies × Dependents × Processes × music_database/postgres@PostgreSQL 16* ×

music_database/postgres@PostgreSQL 16

Query

Query History

1

2

3

4

--Q2 :: Which countries have the most Invoices?

select count(*) as s , billing_country from invoice group by billing_country order by s desc ;

Data Output

Messages

Notifications

s

bigint

billing_country

character varying (30)

1

2

3

4

5

6

7

8

9

10

11

12

131

76

61

50

41

30

29

28

21

13

13

11

USA

Canada

Brazil

France

Germany

Czech Republic

Portugal

United Kingdom

India

Chile

Ireland


Spain

✓ Successfully run. Total query runtime: 46 msec. 24 rows affected. ✕

Total rows: 24 of 24

Query complete 00:00:00.046

Ln 4, Col 95

	total double precision 
1	23.759999999999998
2	19.8
3	19.8
4	19.8
5	19.8
6	18.81
7	17.82
8	17.82
9	17.82
10	17.82

pgAdmin 4

FileObjectToolsHelp

Object Explorer

public

Aggregates

Collations

Domains

FTS Configurations

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized Views

Operators

Procedures

Sequences

Tables (11)

Trigger Functions

Types

Views

Subscriptions

online_store

os

postgres

DashboardPropertiesSQLStatisticsDependenciesDependentsProcessesmusic_database/postgres@PostgreSQL 16*

music_database/postgres@PostgreSQL 16

No limit

E

?

Query

Query History

1

2

3

4

5

6

7

8

/*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 sum(total) as total_invoice_value , billing_city from invoice group by billing_city order by total_invoice_value desc

Data Output

Messages

Notifications

+

📄

▼

🗑️

🗑️

📄

📉

	total_invoice_value double precision	billing_city character varying (30)
1	273.24000000000007	Prague
2	169.29	Mountain View
3	166.32	London
4	158.4	Berlin
5	151.47	Paris
6	129.69	São Paulo
7	114.83999999999997	Dublin
8	111.86999999999999	Delhi
9	108.89999999999998	São José dos Campos
10	106.91999999999999	Brasília
11	102.96000000000001	Lisbon
12	99.99	Bordeaux

✓ Successfully run. Total query runtime: 69 msec. 53 rows affected. ✕

Total rows: 53 of 53

Query complete 00:00:00.069

Ln 4, Col 61

pgAdmin 4

FileObjectToolsHelp

Object Explorer

public

Aggregates

Collations

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

online_store

os

postgres

DashboardPropertiesSQLStatisticsDependenciesDependentsProcessesmusic_database/postgres@PostgreSQL 16*

music_database/postgres@PostgreSQL 16

No limit

QueryQuery History

1

2

3

4

5

6

7

8

9

10

11

12

13

14

/*Q6 :: 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

first_name,

last_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;

Data OutputMessagesNotifications

first_name

last_name

email

character

character

character varying (50)

1

Aaron

Mitchell

aaronmitchell@yahoo.ca

2

Alexandre

Rocha

alero@uol.com.br

3

Astrid

Gruber

astrid.gruber@apple.at

4

Bjørn

Hansen

bjorn.hansen@yahoo.no

5

Camille

Bernard

camille.bernard@yahoo.fr

6

Daan

Peeters

daan_peeters@apple.be

7

Diego

Gutiérrez

diego.gutierrez@yahoo.ar

8

Dan

Miller

dmiller@comcast.com

9

Dominique

Lefebvre

dominiquelefebvre@gmail.c...

10

Edward

Francis

edfrancis@yachoo.ca

Successfully run. Total query runtime: 73 msec. 59 rows affected.

Total rows: 59 of 59

Query complete 00:00:00.073

Ln 5, Col 17

Data Output Messages Notifications

✓ Successfully run. Total query runtime: 72 msec. 59 rows affected. ✕

pgAdmin 4

FileObjectToolsHelp

Object Explorer

public

Aggregates

Collations

Domains

FTS Configurations

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized Views

Operators

Procedures

Sequences

Tables (11)

Trigger Functions

Types

Views

Subscriptions

online_store

os

postgres

DashboardPropertiesSQLStatisticsDependenciesDependentsProcessesmusic_database/postgres@PostgreSQL 16*

music_database/postgres@PostgreSQL 16

No limit

QueryQuery History

1

2

3

4

5

6

7

8

9

10

11

12

13

14

/*Q7 :: 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

songs_num

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

songs_num

desc

limit

10;

Data OutputMessagesNotifications

artist_id

[PK] character varying (50)

name

character varying (120)

songs_num

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

Successfully run. Total query runtime: 94 msec. 10 rows affected.

Total rows: 10 of 10

Query complete 00:00:00.094

Ln 7, Col 11

```
1
2 ✓ /*Q8 :: Return all the track names that have a song length longer than the average song length.
3 Return the Name and Milliseconds for each track. Order by the song length with the
4 longest songs listed first */
5
6 ✓ select name, milliseconds from track where milliseconds > (
7     select avg(milliseconds) as avg_track_length from track
8 ) order by milliseconds desc;
```

	name character varying (150)	milliseconds integer
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 Space	

✓ Successfully run

✓ Successfully run. Total query runtime: 59 msec. 494 rows affected. ✗

Total rows: 494 of 494 Query complete 00:00:00.059

Ln 2, Col 5

```
1  ✓ /*Q9 :: Find how much amount spent by each customer on artists? Write a query to return
2  customer name, artist name and total spent */
3
4  ✓ with best_selling_artist as (
5      select artist.artist_id as artist_id, artist.name as artist_name, sum(invoice_line.unit_price*invoice_line.quantity) as total_sales
6      from invoice_line
7      join track on track.track_id = invoice_line.track_id
8      join album on album.album_id = track.album_id
9      join artist on artist.artist_id = album.artist_id
10     group by 1
11     order by 3 desc limit 1 )
12 select c.customer_id, c.first_name, c.last_name, bsa.artist_name, sum(il.unit_price*il.quantity) as amount_spent
13 from invoice i
14 join customer c on c.customer_id = i.customer_id
15 join invoice_line il on il.invoice_id = i.invoice_id
16 join track t on t.track_id = il.track_id
17 join album alb on alb.album_id = t.album_id
18 join best_selling_artist bsa on bsa.artist_id = alb.artist_id
19 group by 1, 2 ,3 ,4 order by 5 desc ;
```

	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	
7	47	Lucas	Mancini	Queen	

✓ Successfully run. Total query runtime: 59 msec. 43 rows affected. ✕

Ln 19, Col 21


```

1  ✓ /*Q10 :: We want to find out the most popular music Genre for each country. We determine the
2  most popular genre as the genre with the highest amount of purchases. Write a query
3  that returns each country along with the top Genre. For countries where the maximum
4  number of purchases is shared return all Genres */
5
6  ✓ with popular_genre as
7  (
8      select count(invoice_line.quantity) as purchases, customer.country, genre.name, genre.genre_id,
9      row_number() over(partition by customer.country order by count(invoice_line.quantity) desc) as R_Quantity
10     from invoice_line
11     join invoice on invoice.invoice_id = invoice_line.invoice_id
12     join customer on customer.customer_id = invoice.customer_id
13     join track on track.track_id = invoice_line.track_id
14     join genre on genre.genre_id = track.genre_id
15     group by 2,3,4 order by 2 asc , 1 desc
16 )
17 select * from popular_genre where R_Quantity <= 1

```

	purchases bigint	country character varying (50)	name character varying (120)	genre_id character varying (50)	r_quantity 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

✓ Successfully run. Total query runtime: 41 msec. 24 rows affected. ✗

Ln 1, Col 6


```
1  ✓ /*Q11 :: Write a query that determines the customer that has spent the most on music for each
2  country. Write a query that returns the country along with the top customer and how
3  much they spent. For countries where the top amount spent is shared, provide all
4  customers who spent this amount */
5
6  ✓ with Customer_with_country as (
7      select customer.customer_id,first_name,last_name,billing_country,sum(total) as total_spending,
8      row_number() over(partition by billing_country order by sum(total) desc) as RowNo
9      from invoice
10     join customer on customer.customer_id = invoice.customer_id
11     group by 1,2,3,4 order by 4 asc,5 desc)
12 select * from Customer_with_country where RowNo <= 1
13
```

	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	Canada		
7	57	Luis	Roias	Chile	97.020	

✓ Successfully run. Total query runtime: 74 msec. 24 rows affected. ✕



PostgreSQL

Thank You

BY : *Satyabrata Brahmachary*