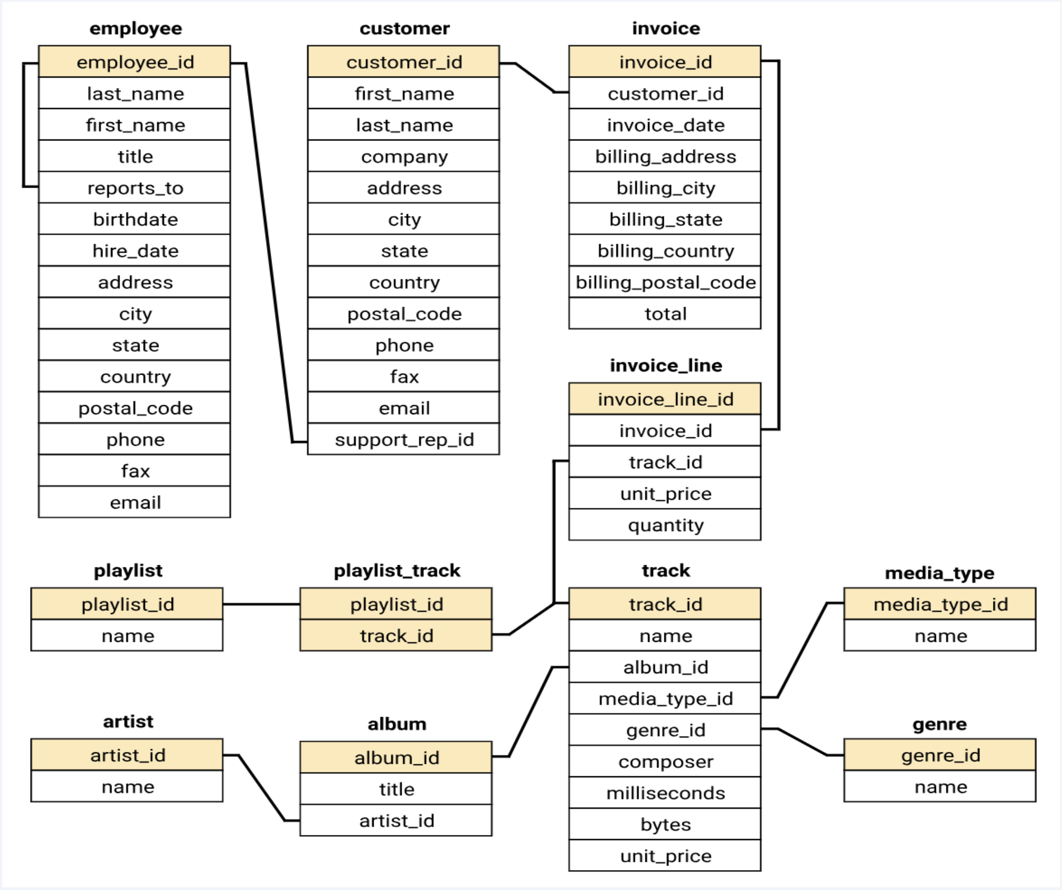
MUSIC STORE ANALYSIS

SQL PROJECT BY ADITYA D GAYEN

OVERVIEW

1. Utilize the power of SQL to embark on a comprehensive journey through the digital music store database.
2. Extend a gentle hand to stakeholders, offering them insights finely tuned to their needs for thoughtful decision-making.
3. Explore geographical growth patterns, purchasing power variations, and total revenue trends.
4. Identify the most profitable city in terms of revenue generation and the highest-spending customers.
5. Provide actionable recommendations tailored to drive business growth and optimize performance metrics.

SCHEMA DIAGRAM



employee
- employee_id
- last_name
- first_name
- title
- reports_to
- birthdate
- hire_date
- address
- city
- state
- country
- postal_code
- phone
- fax
- email

customer
- customer_id
- first_name
- last_name
- company
- address
- city
- state
- country
- postal_code
- phone
- fax
- email
- support_rep_id

invoice
- invoice_id
- customer_id
- invoice_date
- billing_address
- billing_city
- billing_state
- billing_country
- billing_postal_code
- total

invoice_line
- invoice_line_id
- invoice_id
- track_id
- unit_price
- quantity

playlist
- playlist_id
- name

playlist_track
- playlist_id
- track_id

track
- track_id
- name
- album_id
- media_type_id
- genre_id
- composer
- milliseconds
- bytes
- unit_price

media_type
- media_type_id
- name

artist
- artist_id
- name

album
- album_id
- title
- artist_id

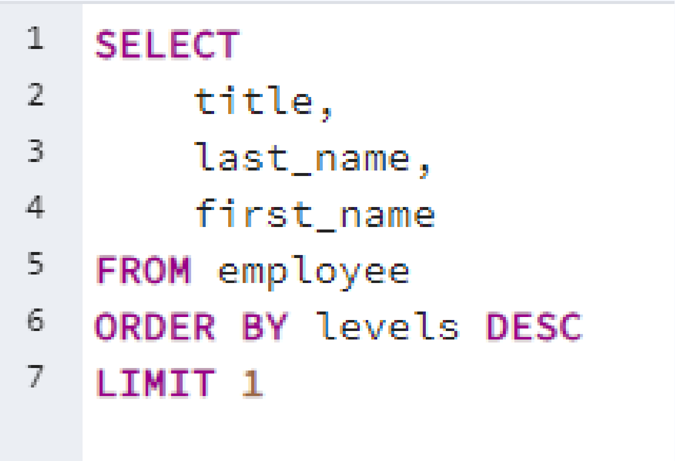
genre
- genre_id
- name

A schema diagram visually displays a database's structure, including tables, columns, and their relationships. It acts as a blueprint for understanding data organization, aiding in database design, development, and maintenance. This clear overview supports efficient data management and informed decision-making for administrators, developers, and stakeholders.

EASY

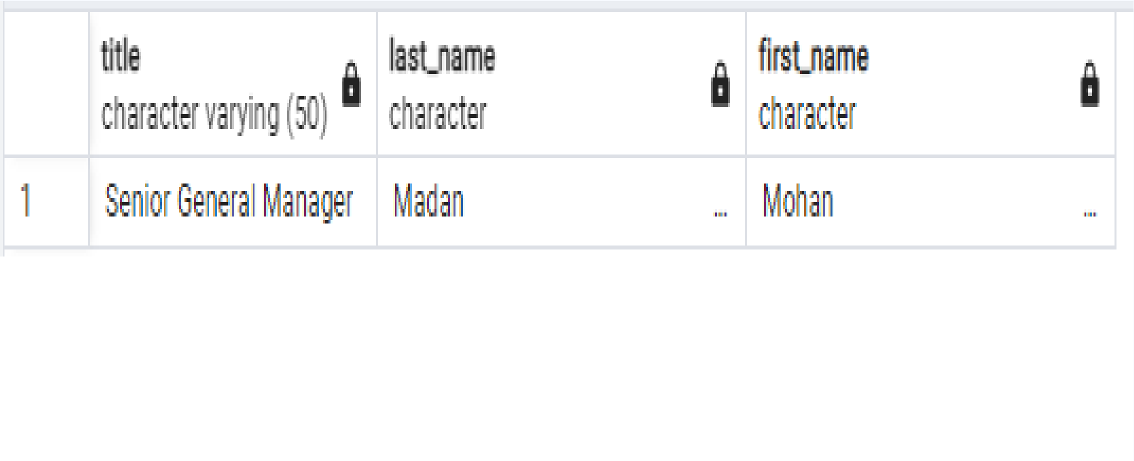
WHO IS THE SENIOR MOST EMPLOYEE BASED ON JOB TITLE?

QUERY



```
SELECT
    title,
    last_name,
    first_name
FROM employee
ORDER BY levels DESC
LIMIT 1
```

TABLE

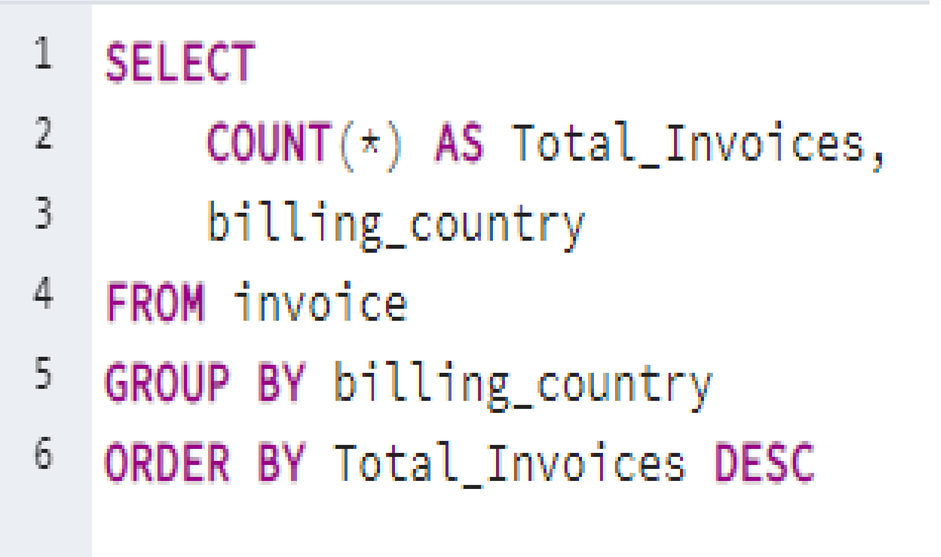


	title character varying (50)	last_name character	first_name character
1	Senior General Manager	Madan	Mohan

EASY

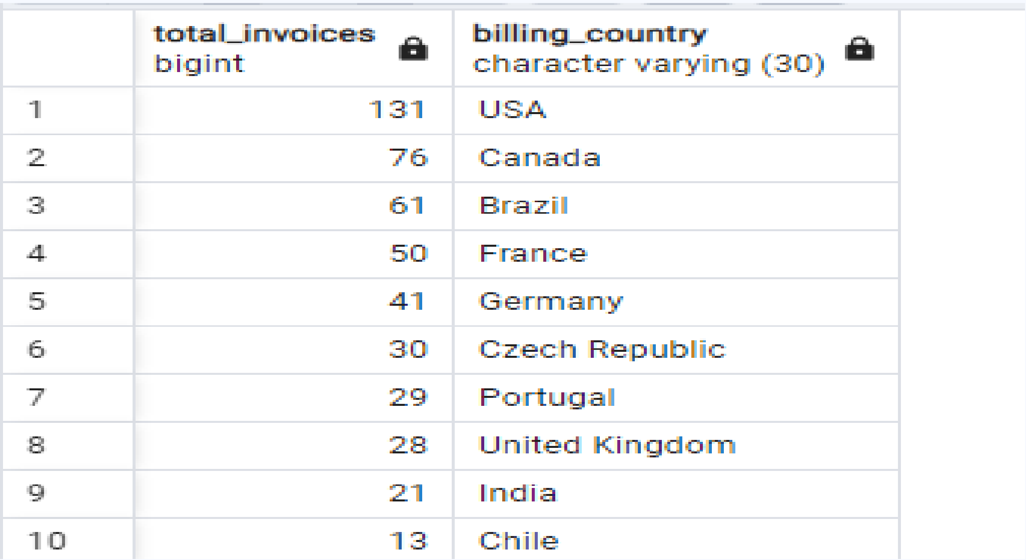
WHICH COUNTRIES HAVE THE MOST INVOICES?

QUERY



```
SELECT
    COUNT(*) AS Total_Invoices,
    billing_country
FROM invoice
GROUP BY billing_country
ORDER BY Total_Invoices DESC
```

TABLE

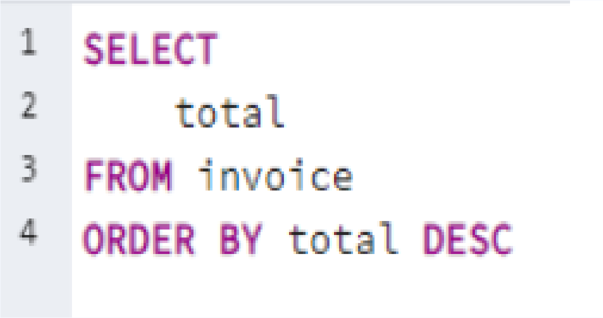


	total_invoices bigint	billing_country character varying (30)
1	131	USA
2	76	Canada
3	61	Brazil
4	50	France
5	41	Germany
6	30	Czech Republic
7	29	Portugal
8	28	United Kingdom
9	21	India
10	13	Chile

EASY

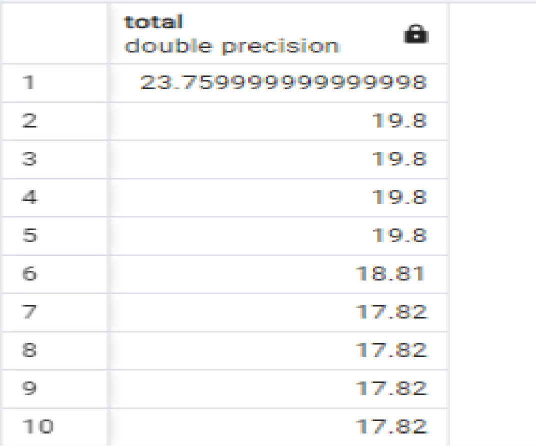
WHAT ARE TOP 3 VALUES OF TOTAL INVOICE?

QUERY



```
SELECT
    total
FROM invoice
ORDER BY total DESC
```

TABLE

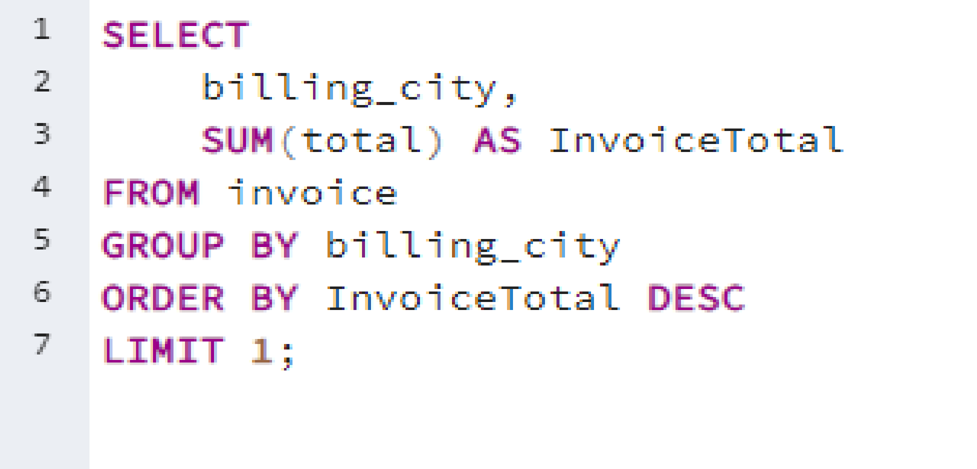


	total double precision
1	23.759999999999998
2	19.8
3	19.8
4	19.8
5	19.8
6	18.81
7	17.82
8	17.82
9	17.82
10	17.82

EASY

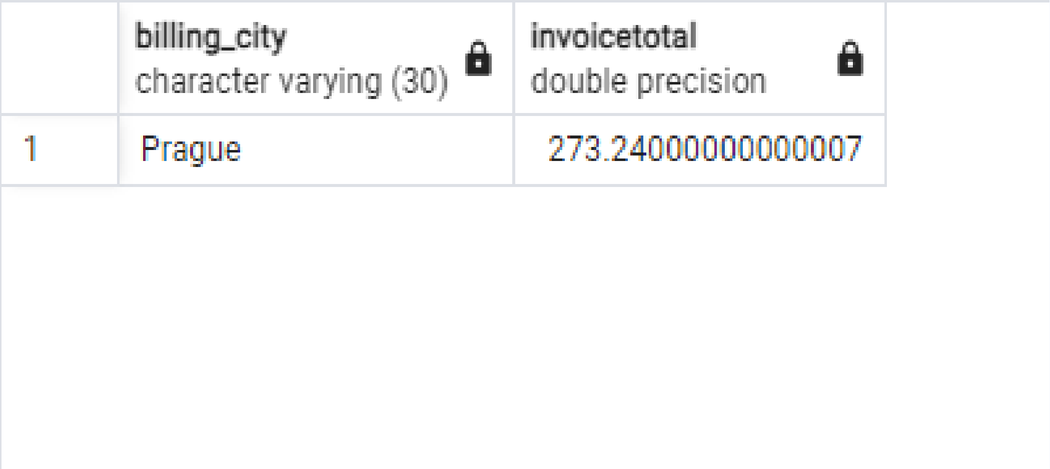
WHICH CITY HAS THE BEST CUSTOMERS? WE WOULD LIKE TO THROW A PROMOTIONAL MUSIC FESTIVAL IN THE CITY WE MADE THE MOST MONEY ?. WRITE A QUERY THAT RETURNS ONE CITY THAT HAS THE HIGHEST SUM OF INVOICE TOTALS. RETURN BOTH THE CITY NAME & SUM OF ALL INVOICE TOTALS

QUERY



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

TABLE

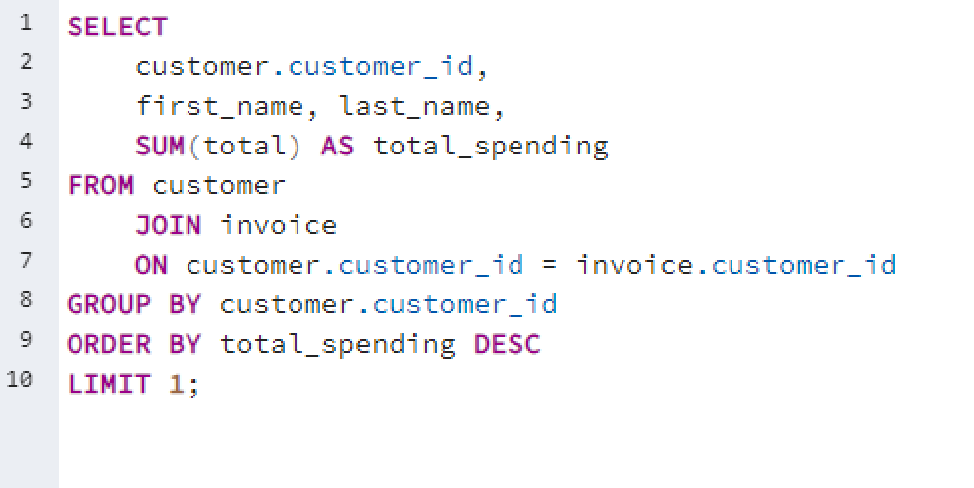


	billing_city character varying (30) 🔒	invoicetotal double precision 🔒
1	Prague	273.24000000000007

EASY

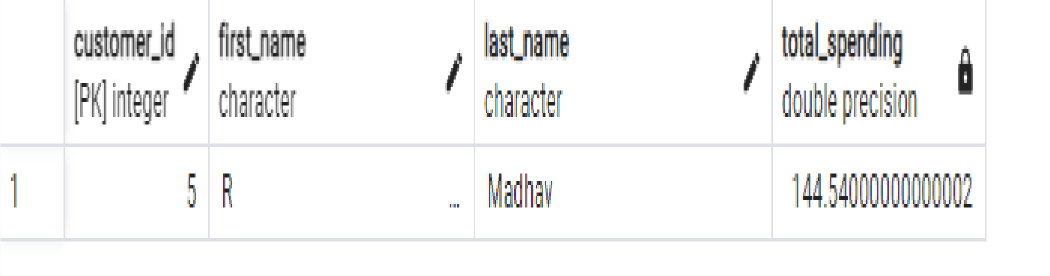
WHO IS THE BEST CUSTOMER? THE CUSTOMER WHO HAS SPENT THE MOST MONEY WILL BE DECLARED THE BEST CUSTOMER? WRITE A QUERY THAT RETURNS THE PERSON WHO HAS SPENT THE MOST MONEY

QUERY



```sql
SELECT
    customer.customer_id,
    first_name, last_name,
    SUM(total) AS total_spending
FROM customer
    JOIN invoice
    ON customer.customer_id = invoice.customer_id
GROUP BY customer.customer_id
ORDER BY total_spending DESC
LIMIT 1;
```

TABLE

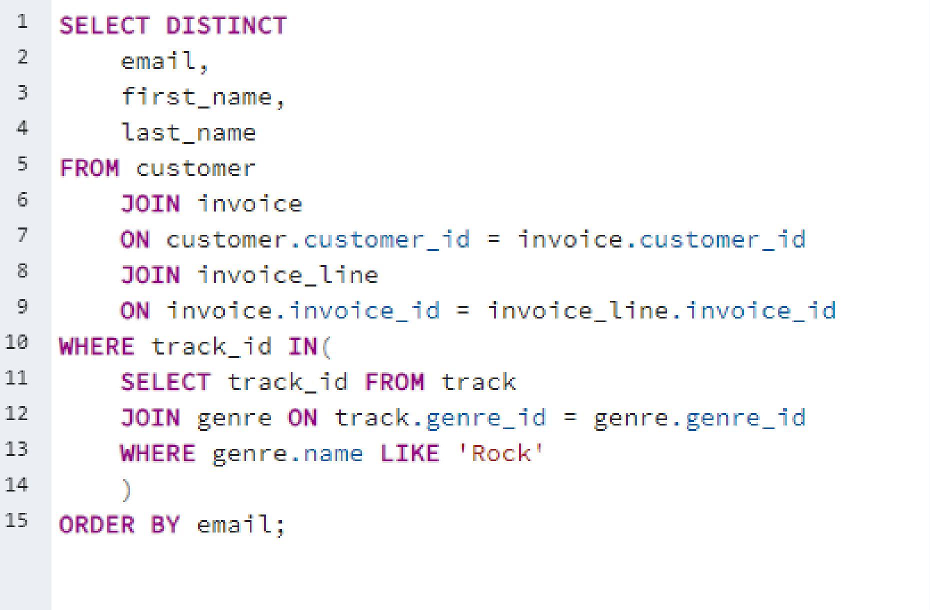


	customer_id [PK] integer	first_name character	last_name character	total_spending double precision
1	5	R	Madhav	144.54000000000002

MODERATE

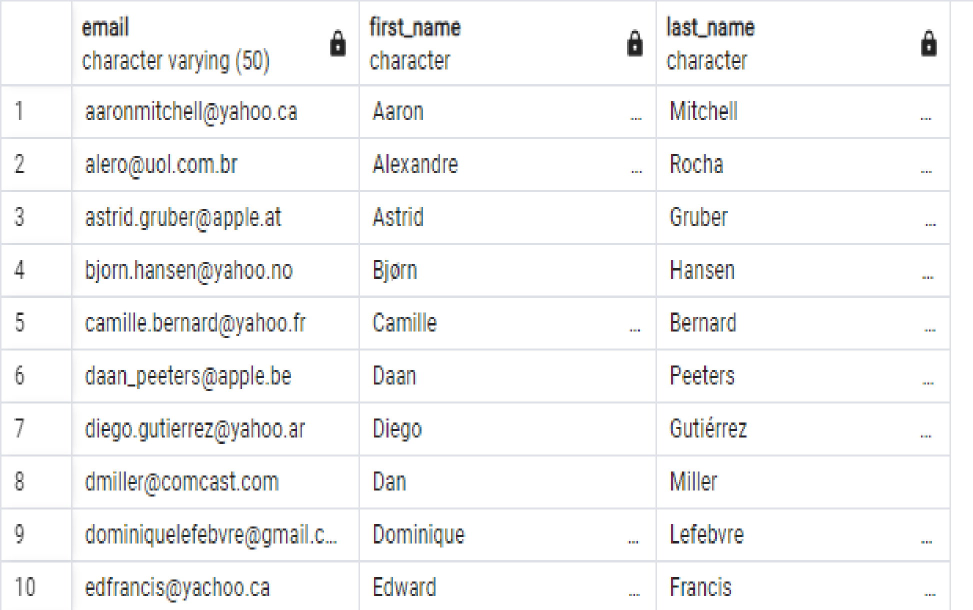
WRITE QUERY TO RETURN THE EMAIL, FIRST NAME, LAST NAME, & GENRE OF ALL ROCK MUSIC LISTENERS. RETURN YOUR LIST ORDERED ALPHABETICALLY BY EMAIL STARTING WITH A.

QUERY



```sql
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 LIKE 'Rock'
    )
ORDER BY email;
```

TABLE

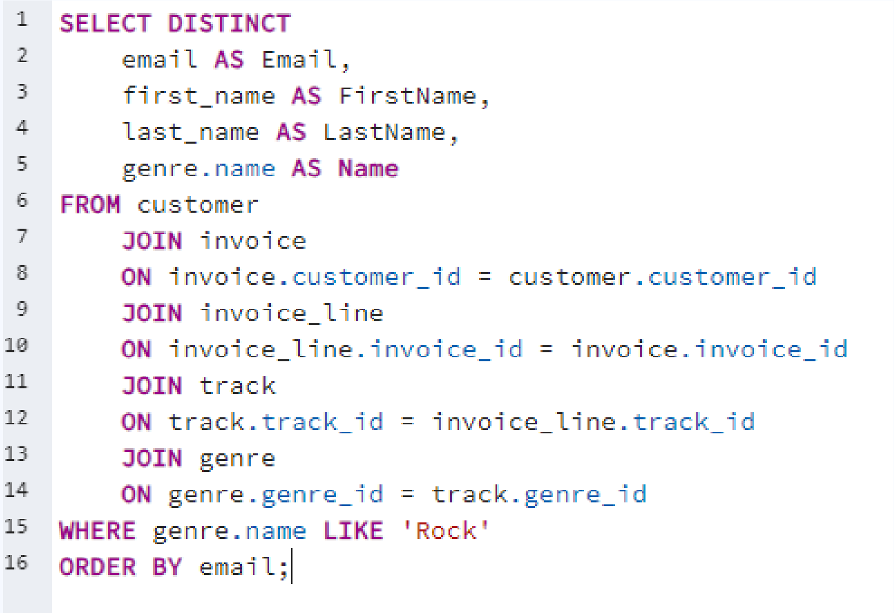


	email character varying (50)	first_name character	last_name character
1	aaronmitchell@yahoo.ca	Aaron	Mitchell
2	alero@uol.com.br	Alexandre	Rocha
3	astrid.gruber@apple.at	Astrid	Gruber
4	bjorn.hansen@yahoo.no	Bjørn	Hansen
5	camille.bernard@yahoo.fr	Camille	Bernard
6	daan_peeters@apple.be	Daan	Peeters
7	diego.gutierrez@yahoo.ar	Diego	Gutiérrez
8	dmiller@comcast.com	Dan	Miller
9	dominiquelefebvre@gmail.c...	Dominique	Lefebvre
10	edfrancis@yachoo.ca	Edward	Francis

MODERATE

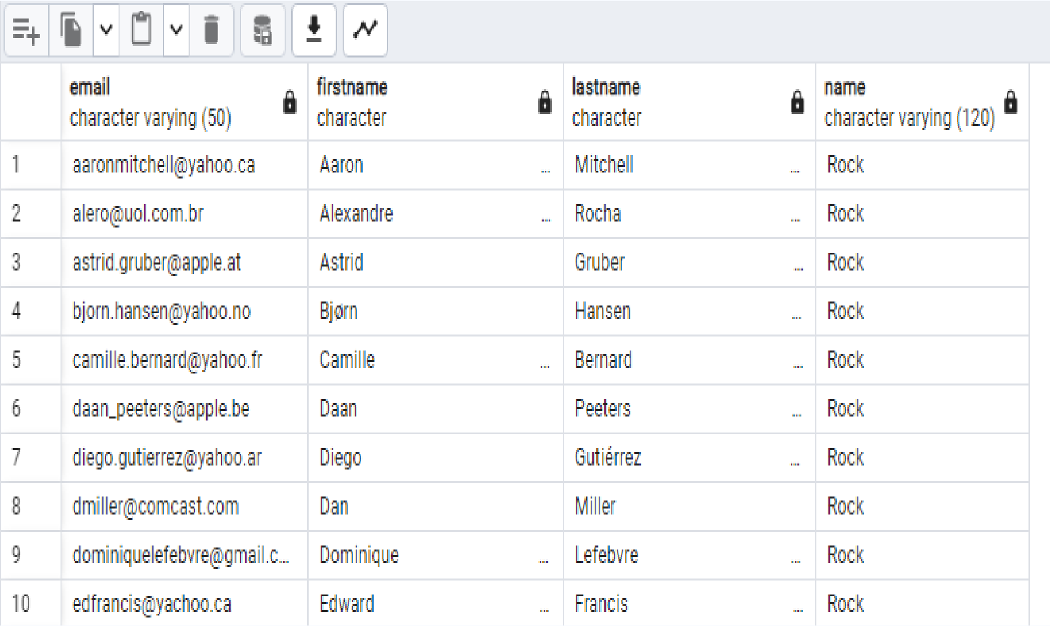
WRITE QUERY TO RETURN THE EMAIL, FIRST NAME, LAST NAME, & GENRE OF ALL ROCK MUSIC LISTENERS. RETURN YOUR LIST ORDERED ALPHABETICALLY BY EMAIL STARTING WITH A. (THIS IS SOLVED IN A DIFFERENT METHOD)

QUERY



```
SELECT DISTINCT
    email AS Email,
    first_name AS FirstName,
    last_name AS LastName,
    genre.name AS Name
FROM customer
    JOIN invoice
    ON invoice.customer_id = customer.customer_id
    JOIN invoice_line
    ON invoice_line.invoice_id = invoice.invoice_id
    JOIN track
    ON track.track_id = invoice_line.track_id
    JOIN genre
    ON genre.genre_id = track.genre_id
WHERE genre.name LIKE 'Rock'
ORDER BY email;
```

TABLE

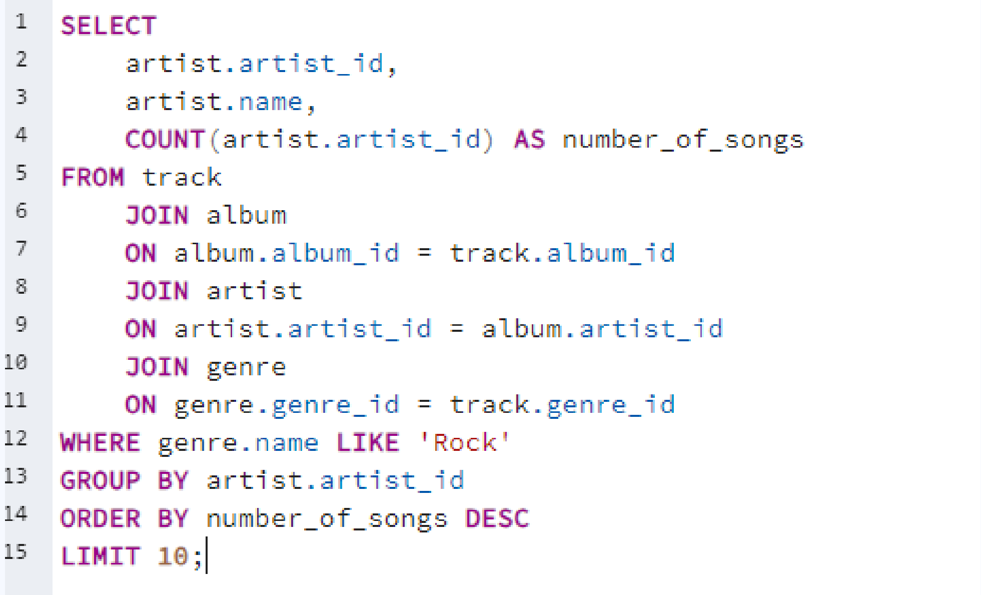


	email character varying (50)	firstname character	lastname character	name character varying (120)
1	aaronmitchell@yahoo.ca	Aaron	Mitchell	Rock
2	alero@uol.com.br	Alexandre	Rocha	Rock
3	astrid.gruber@apple.at	Astrid	Gruber	Rock
4	bjorn.hansen@yahoo.no	Bjørn	Hansen	Rock
5	camille.bernard@yahoo.fr	Camille	Bernard	Rock
6	daan_peeters@apple.be	Daan	Peeters	Rock
7	diego.gutierrez@yahoo.ar	Diego	Gutiérrez	Rock
8	dmiller@comcast.com	Dan	Miller	Rock
9	dominiquelefebvre@gmail.c...	Dominique	Lefebvre	Rock
10	edfrancis@yachoo.ca	Edward	Francis	Rock

MODERATE

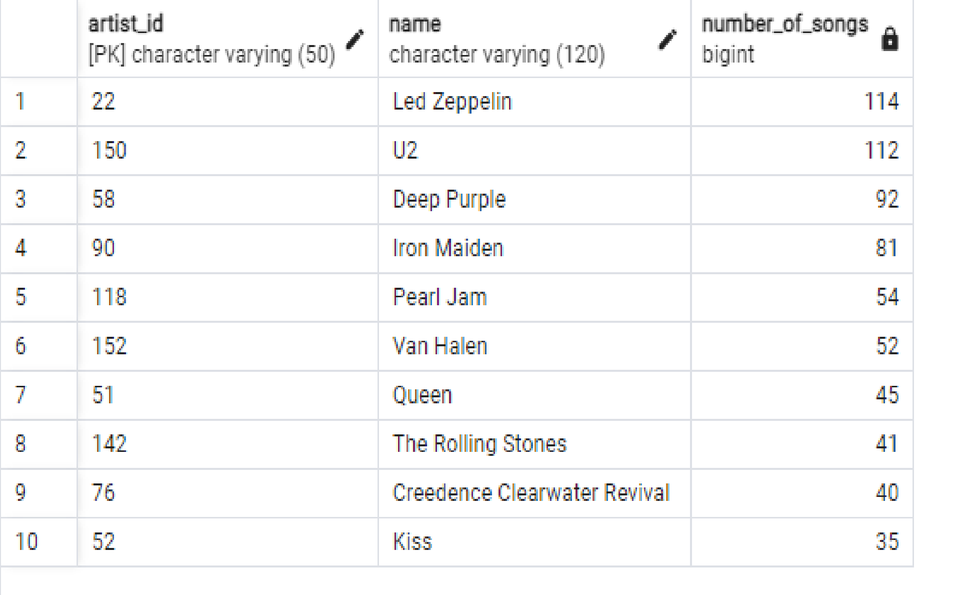
LET'S INVITE THE ARTISTS WHO HAVE WRITTEN THE MOST ROCK MUSIC IN OUR DATASET. WRITE A QUERY THAT RETURNS THE ARTIST NAME AND TOTAL TRACK COUNT OF THE TOP 10 ROCK BANDS.

QUERY



```
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;
```

TABLE

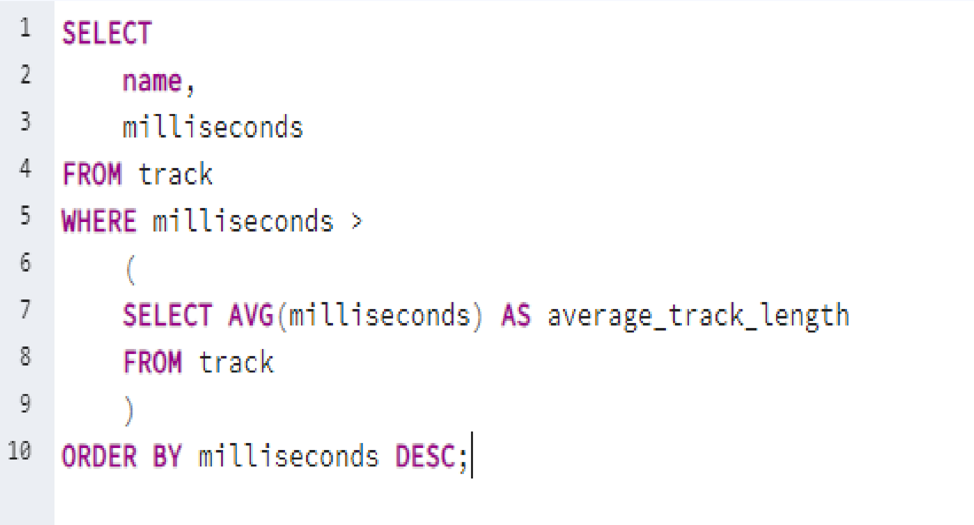


	artist_id [PK] character varying (50)	name character varying (120)	number_of_songs bigint
1	22	Led Zeppelin	114
2	150	U2	112
3	58	Deep Purple	92
4	90	Iron Maiden	81
5	118	Pearl Jam	54
6	152	Van Halen	52
7	51	Queen	45
8	142	The Rolling Stones	41
9	76	Creedence Clearwater Revival	40
10	52	Kiss	35

MODERATE

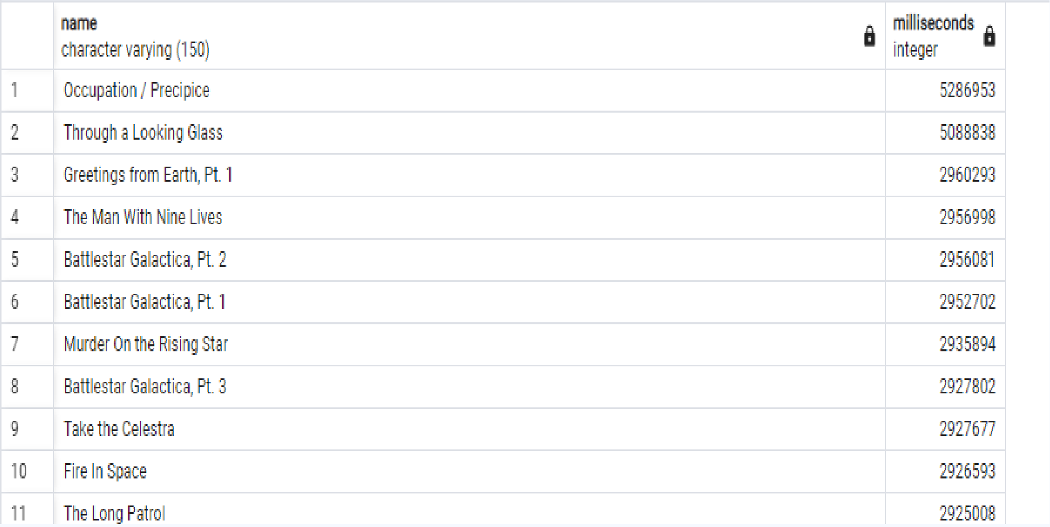
RETURN ALL THE TRACK NAMES THAT HAVE A SONG LENGTH LONGER THAN THE AVERAGE SONG LENGTH. RETURN THE NAME AND MILLISECONDS FOR EACH TRACK. ORDER BY THE SONG LENGTH WITH THE LONGEST SONGS LISTED FIRST.

QUERY



```
SELECT
    name,
    milliseconds
FROM track
WHERE milliseconds >
    (
    SELECT AVG(milliseconds) AS average_track_length
    FROM track
    )
ORDER BY milliseconds DESC;
```

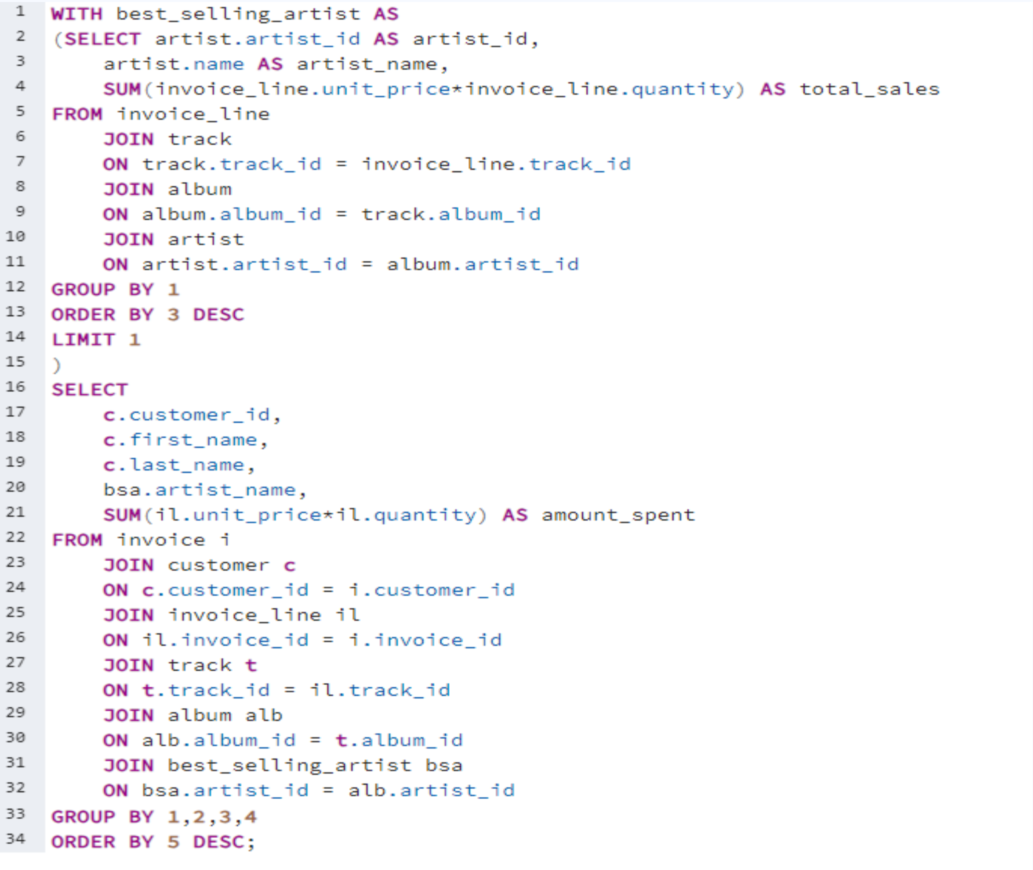
TABLE



	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	2926593
11	The Long Patrol	2925008

ADVANCED

QUERY

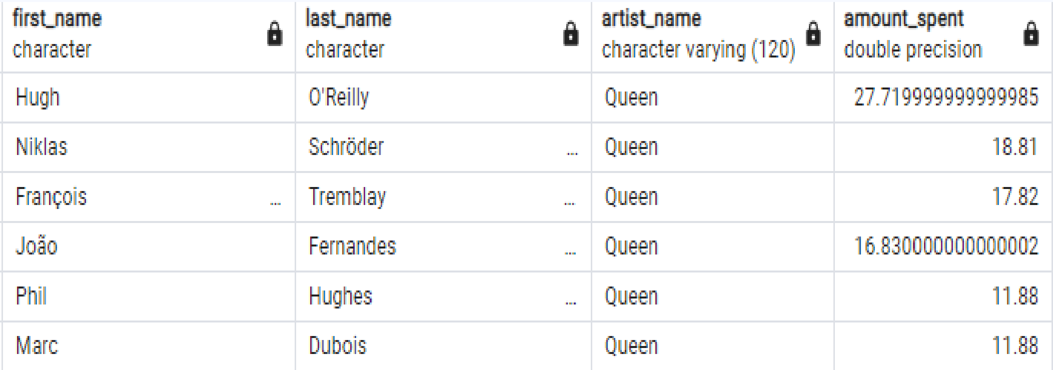


```sql
WITH best_selling_artist AS
(SELECT artist.artist_id AS artist_id,
    artist.name AS artist_name,
    SUM(invoice_line.unit_price*invoice_line.quantity) AS total_sales
FROM invoice_line
    JOIN track
    ON track.track_id = invoice_line.track_id
    JOIN album
    ON album.album_id = track.album_id
    JOIN artist
    ON artist.artist_id = album.artist_id
GROUP BY 1
ORDER BY 3 DESC
LIMIT 1
)
SELECT
    c.customer_id,
    c.first_name,
    c.last_name,
    bsa.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;
```

FIND HOW MUCH AMOUNT SPENT BY EACH CUSTOMER ON ARTISTS? WRITE A QUERY TO RETURN CUSTOMER NAME, ARTIST NAME AND TOTAL SPENT FIRST, FIND WHICH ARTIST HAS EARNED THE MOST ACCORDING TO THE INVOICE LINES.

NOW USE THIS ARTIST TO FIND WHICH CUSTOMER SPENT THE MOST ON THIS ARTIST. FOR THIS QUERY, YOU WILL NEED TO USE THE INVOICE, INVOICE LINE, TRACK, CUSTOMER, ALBUM, AND ARTIST TABLES. NOTE, THIS ONE IS TRICKY BECAUSE THE TOTAL SPENT IN THE INVOICE TABLE MIGHT NOT BE ON A SINGLE PRODUCT, SO YOU 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.

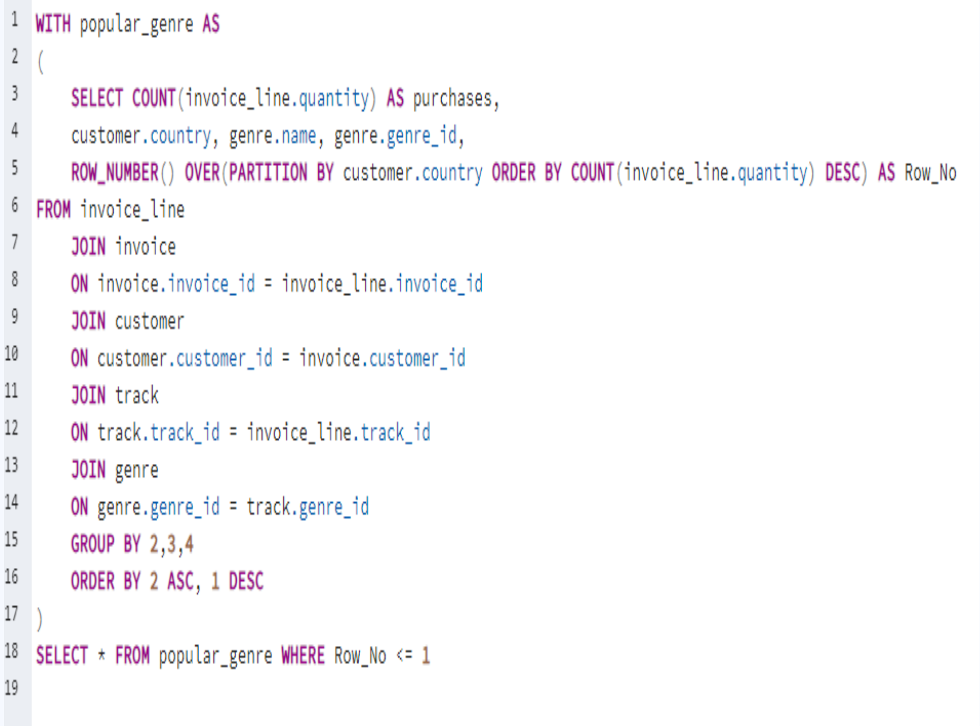
TABLE



first_name character	last_name character	artist_name character varying (120)	amount_spent double precision
Hugh	O'Reilly	Queen	27.719999999999985
Niklas	Schröder	Queen	18.81
François	Tremblay	Queen	17.82
João	Fernandes	Queen	16.830000000000002
Phil	Hughes	Queen	11.88
Marc	Dubois	Queen	11.88

ADVANCED

QUERY

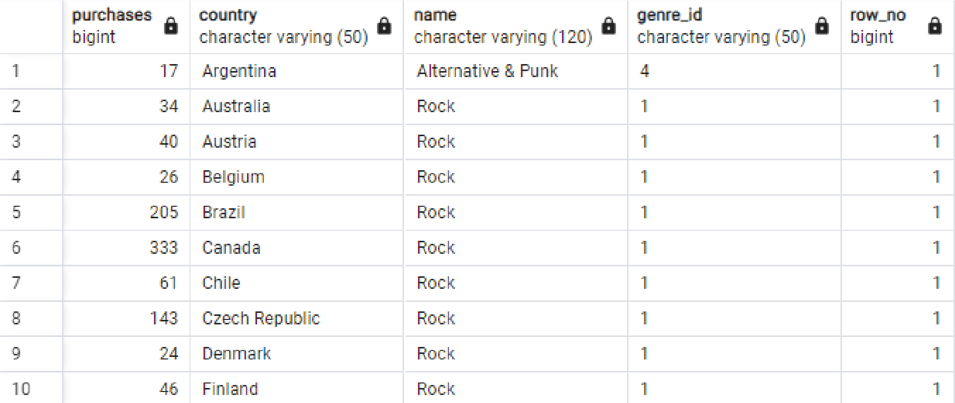


```
WITH popular_genre AS
(
    SELECT COUNT(invoice_line.quantity) AS purchases,
    customer.country, genre.name, genre.genre_id,
    ROW_NUMBER() OVER(PARTITION BY customer.country ORDER BY COUNT(invoice_line.quantity) DESC) AS Row_No
FROM invoice_line
    JOIN invoice
    ON invoice.invoice_id = 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 Row_No <= 1
```

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.

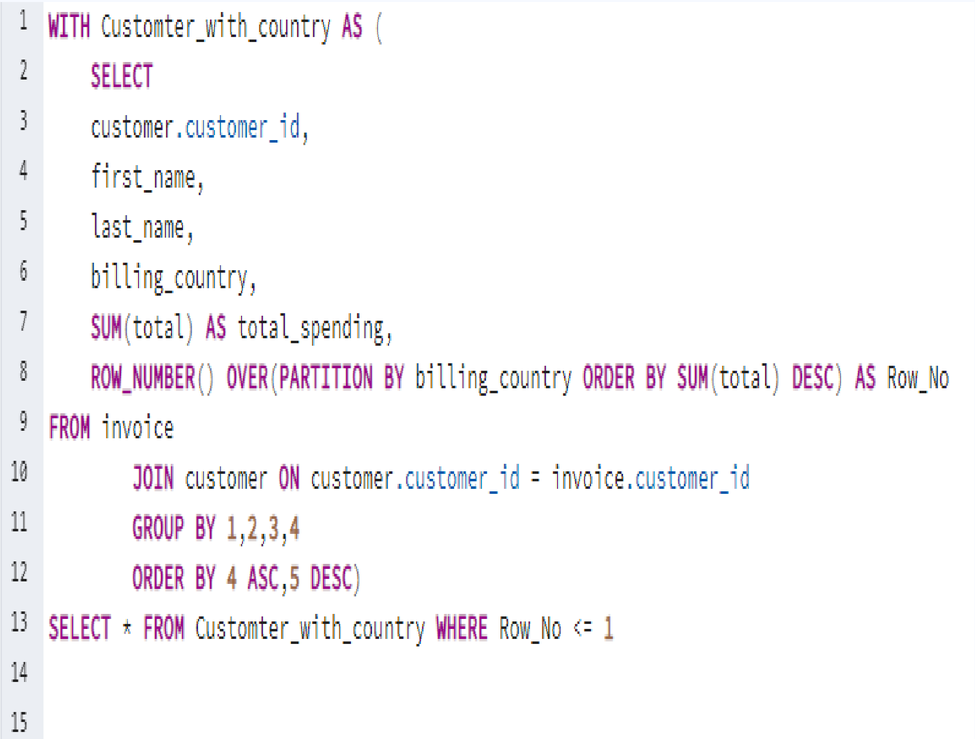
TABLE



	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
8	143	Czech Republic	Rock	1	1
9	24	Denmark	Rock	1	1
10	46	Finland	Rock	1	1

ADVANCED

QUERY

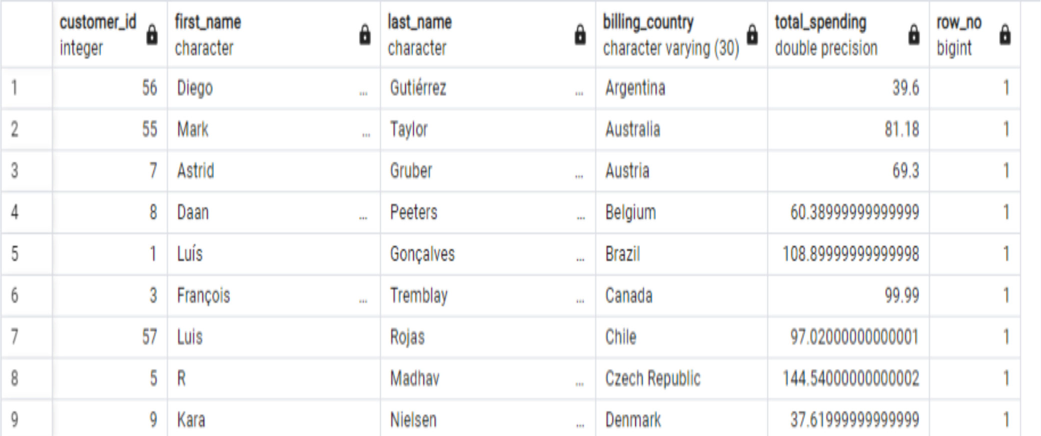


```
WITH Customter_with_country AS (
    SELECT
    customer.customer_id,
    first_name,
    last_name,
    billing_country,
    SUM(total) AS total_spending,
    ROW_NUMBER() OVER(PARTITION BY billing_country ORDER BY SUM(total) DESC) AS Row_No
FROM invoice
        JOIN customer ON customer.customer_id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

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.

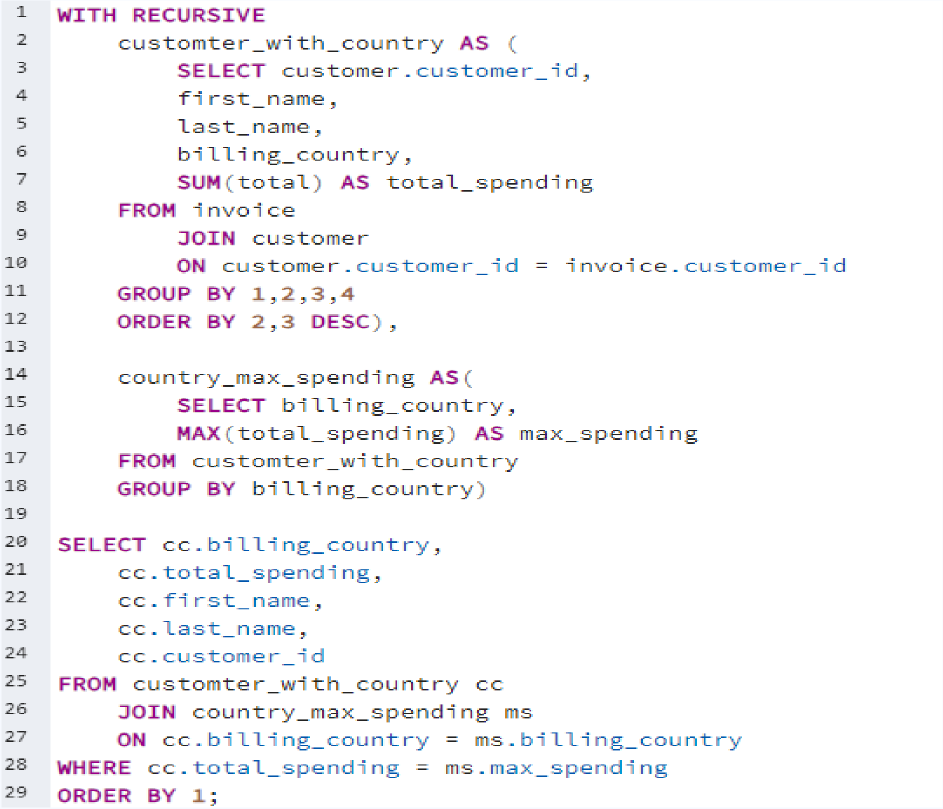
TABLE



	customer_id integer	first_name character	last_name character	billing_country character varying (30)	total_spending double precision	row_no 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	99.99	1
7	57	Luis	Rojas	Chile	97.02000000000001	1
8	5	R	Madhav	Czech Republic	144.54000000000002	1
9	9	Kara	Nielsen	Denmark	37.61999999999999	1

ADVANCED

QUERY

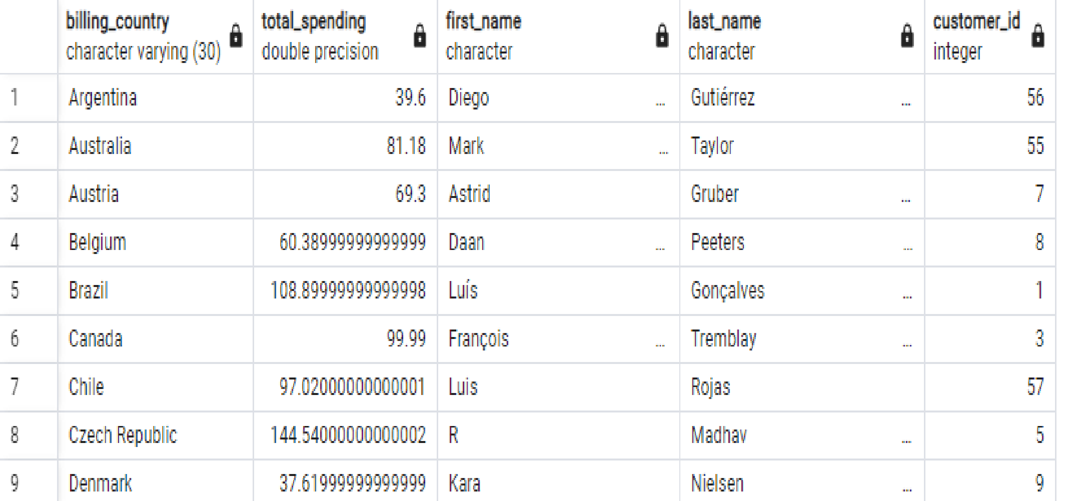


```sql
WITH RECURSIVE
    customter_with_country AS (
        SELECT customer.customer_id,
        first_name,
        last_name,
        billing_country,
        SUM(total) AS total_spending
    FROM invoice
        JOIN customer
        ON customer.customer_id = invoice.customer_id
    GROUP BY 1,2,3,4
    ORDER BY 2,3 DESC),

    country_max_spending AS(
        SELECT billing_country,
        MAX(total_spending) AS max_spending
    FROM customter_with_country
    GROUP BY billing_country)

SELECT cc.billing_country,
    cc.total_spending,
    cc.first_name,
    cc.last_name,
    cc.customer_id
FROM customter_with_country cc
    JOIN country_max_spending ms
    ON cc.billing_country = ms.billing_country
WHERE cc.total_spending = ms.max_spending
ORDER BY 1;
```

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.

THERE ARE TWO PARTS IN QUESTION FIRST FIND THE MOST SPENT ON MUSIC FOR EACH COUNTRY AND SECOND FILTER THE DATA FOR RESPECTIVE CUSTOMERS.

TABLE



	billing_country character varying (30)	total_spending double precision	first_name character	last_name character	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
5	Brazil	108.89999999999998	Luís	Gonçalves	1
6	Canada	99.99	François	Tremblay	3
7	Chile	97.02000000000001	Luis	Rojas	57
8	Czech Republic	144.54000000000002	R	Madhav	5
9	Denmark	37.61999999999999	Kara	Nielsen	9

THANK YOU