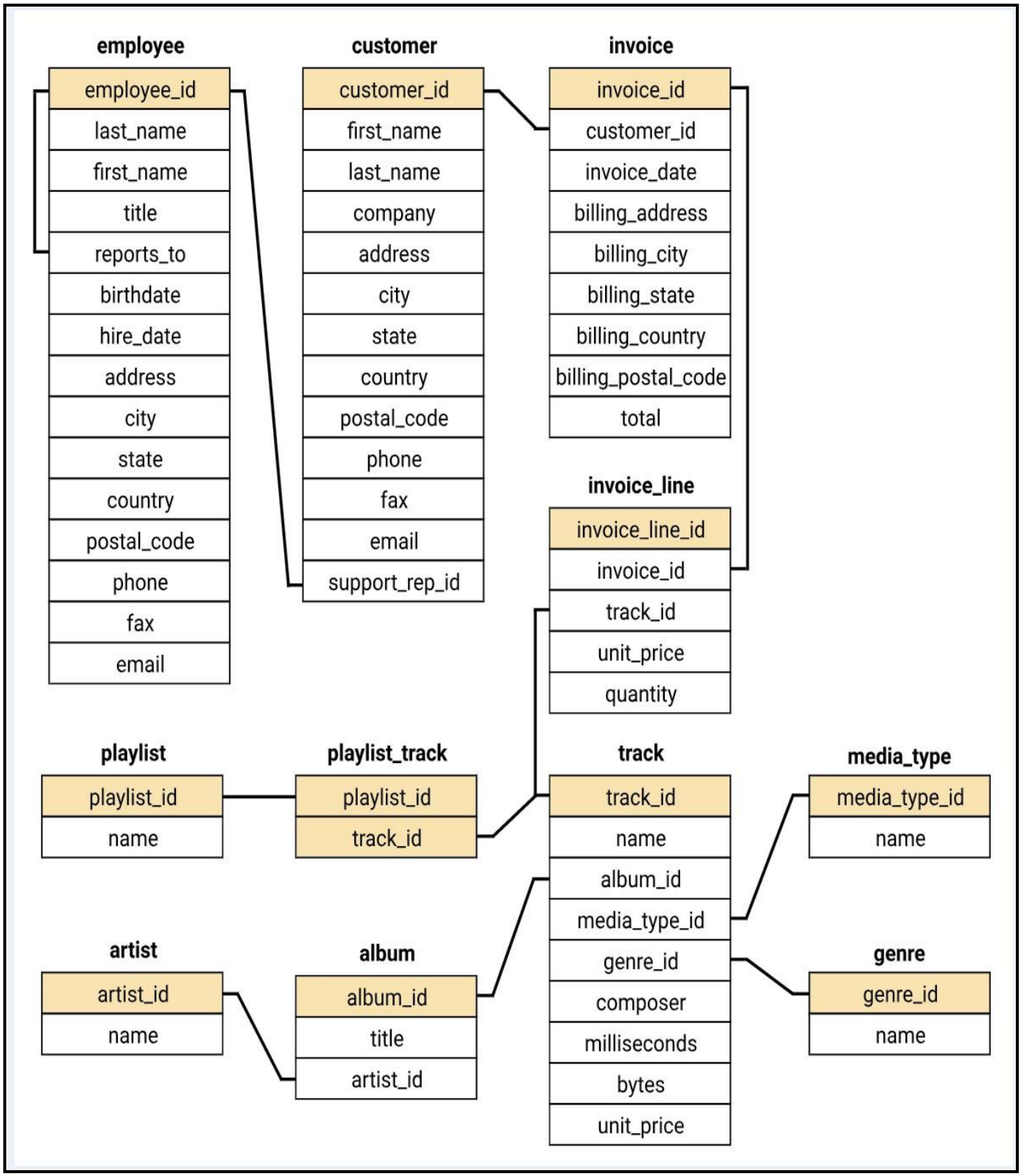
SQL PROJECT- MUSIC STORE DATA ANALYSIS

Schema Diagram



SQL PROJECT- MUSIC STORE DATA ANALYSIS

Query | Query History

```sql
--- MUSIC STORE DATA ANALYSIS

/* Q1: Who is the senior most employee based on job title? */
SELECT * FROM employee

SELECT * FROM employee
ORDER BY levels DESC
LIMIT 1
```

Data Output | Messages | Notifications

	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)
1	9	Madan	Mohan	Senior General Manager	[null]	L7

	birthdate timestamp without time zone	hire_date timestamp without time zone	address character varying (120)	city character varying (50)	state character varying (50)	country character varying (30)
1	1961-01-26 00:00:00	2016-01-14 00:00:00	1008 Vrinda Ave MT	Edmonton	AB	Canada

postal_code character varying (30)	phone character varying (30)	fax character varying (30)	email character varying (30)
T5K 2N1	+1 (780) 428-9482	+1 (780) 428-3457	madan.mohan@chinookcorp.com

```sql
/* Q2: Which countries have the most Invoices? */

SELECT * FROM invoice

SELECT COUNT(*) AS c, billing_country
FROM invoice
GROUP BY billing_country
ORDER BY c dESC
```

Data Output | Messages | Notifications

	c 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
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

```sql
/* Q3: What are top 3 values of total invoice? */

select  total from invoice
order by total desc
limit 3
```

Data Output | Messages | Notifications

	total double precision
1	23.759999999999998
2	19.8
3	19.8

```sql
/* 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 * FROM invoice;

SELECT SUM(total) AS invoice_total, billing_city FROM invoice
GROUP BY billing_city
ORDER BY invoice_total DESC
```

Data Output Messages Notifications

	invoice_total 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
13	99.99	Montréal
14	98.01	Madrid
15	98.01	Redmond
16	97.02000000000001	Santiago
17	94.05000000000001	Frankfurt
18	92.07000000000001	Orlando
19	91.08000000000001	Reno
20	91.08	Ottawa
21	86.13000000000002	Fort Worth
22	84.14999999999999	Tucson
23	82.17	Stuttgart
24	82.17	Rio de Janeiro
25	82.17	Porto
26	81.18	Sidney
27	79.2	New York
28	79.2	Edinburgh
29	79.2	Helsinki
30	78.21	Budapest
31	76.23	Madison
32	76.22999999999999	Warsaw
33	75.24000000000001	Yellowknife
34	75.24	Stockholm
35	73.25999999999999	Dijon
36	72.27000000000001	Oslo
37	72.27	Salt Lake City
38	71.28	Chicago
39	71.28	Bangalore
40	70.28999999999999	Winnipeg
41	69.3	Vienne
42	66.33	Vancouver
43	66.33	Boston
44	65.34	Amsterdam
45	64.35	Lyon
46	62.37000000000005	Halifax
47	60.38999999999999	Brussels
48	54.44999999999996	Cupertino
49	50.49	Rome
50	40.59	Toronto
51	39.6	Buenos Aires
52	37.61999999999999	Copenhagen
53	29.69999999999996	Edmonton

```sql
/* Q5: 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.*/

SELECT * FROM customer

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

Data Output Messages Notifications

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

```sql
/* 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. */

-- Method 1

SELECT DISTINCT email, first_name, last_name
FROM customer c
JOIN invoice i ON c.customer_id=i.customer_id
JOIN invoice_line il ON i.invoice_id = il.invoice_id
WHERE track_id IN(
SELECT track_id FROM track t
JOIN genre g ON t.genre_id = g.genre_id
WHERE g.name LIKE 'Rock'
)
ORDER BY email;

-- Method 2

SELECT DISTINCT email AS Email,first_name AS FirstName, last_name AS LastName, g.name AS Name
FROM customer c
JOIN invoice i ON i.customer_id = c.customer_id
JOIN invoice_line il ON il.invoice_id = i.invoice_id
JOIN track t ON t.track_id = il.track_id
JOIN genre g ON g.genre_id = t.genre_id
WHERE g.name LIKE 'Rock'
ORDER BY email;
```

Data Output Messages Notifications

	email character varying (50)	first_name character (50)	last_name character (50)
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
11	eduardo@woodstock.com.br	Eduardo	Martins
12	ellie.sullivan@shaw.ca	Ellie	Sullivan
13	emma_jones@hotmail.com	Emma	Jones
14	enrique_munoz@yahoo.es	Enrique	Muñoz
15	fernadaramos4@uol.com.br	Fernanda	Ramos
16	fharris@google.com	Frank	Harris
17	fralston@gmail.com	Frank	Ralston
18	ftremblay@gmail.com	François	Tremblay
19	fzimmermann@yahoo.de	Fynn	Zimmermann
20	hannah.schneider@yahoo.de	Hannah	Schneider
21	hholy@gmail.com	Helena	Holý
22	hleacock@gmail.com	Heather	Leacock
23	hughoreilly@apple.ie	Hugh	O'Reilly
24	isabelle_mercier@apple.fr	Isabelle	Mercier
25	jacksmith@microsoft.com	Jack	Smith
26	jenniferp@rogers.ca	Jennifer	Peterson
27	jfernandes@yahoo.pt	João	Fernandes
28	joakim.johansson@yahoo.se	Joakim	Johansson
29	johavanderberg@yahoo.nl	Johannes	Van der Berg
30	johngordon22@yahoo.com	John	Gordon
31	jubarnett@gmail.com	Julia	Barnett
32	kachase@hotmail.com	Kathy	Chase
33	kara.nielsen@jubii.dk	Kara	Nielsen
34	ladislav_kovacs@apple.hu	Ladislav	Kovács
35	leonekohler@surfeu.de	Leonie	Köhler
36	lucas.mancini@yahoo.it	Lucas	Mancini
37	luisg@embraer.com.br	Luís	Gonçalves
38	luisrojas@yahoo.cl	Luis	Rojas
39	manoj.pareek@rediff.com	Manoj	Pareek
40	marc.dubois@hotmail.com	Marc	Dubois
41	mark.taylor@yahoo.au	Mark	Taylor
42	marthasilk@gmail.com	Martha	Silk
43	masampaio@sapo.pt	Madalena	Sampaio
44	michelleb@aol.com	Michelle	Brooks
45	mphilips12@shaw.ca	Mark	Philips
46	nschroder@surfeu.de	Niklas	Schröder
47	patrick.gray@aol.com	Patrick	Gray
48	phil.hughes@gmail.com	Phil	Hughes
49	puja_srivastava@yahoo.in	Puja	Srivastava
50	r.madhav@jetbrains.com	R	Madhav
51	ricunningham@hotmail.com	Richard	Cunningham
52	robbrown@shaw.ca	Robert	Brown
53	roberto.almeida@riotur.gov.br	Roberto	Almeida
54	stanislaw.wojcik@wp.pl	Stanisław	Wójcik
55	steve.murray@yahoo.uk	Steve	Murray
56	terhi.hamalainen@apple.fi	Terhi	Hämäläinen
57	tgoyer@apple.com	Tim	Goyer
58	vstevens@yahoo.com	Victor	Stevens
59	wyatt.girard@yahoo.fr	Wyatt	Girard

```sql
/* 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 a.artist_id, a.name, COUNT(a.artist_id) AS number_of_songs
FROM track t
JOIN album ab ON ab.album_id = t.album_id
JOIN artist a ON a.artist_id = ab.artist_id
JOIN genre g ON g.genre_id = t.genre_id
WHERE g.name LIKE 'Rock'
GROUP BY a.artist_id
ORDER BY number_of_songs DESC
LIMIT 10;
```

Data Output Messages Notifications

	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

```sql
/* Q8: 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. */

SELECT NAME,milliseconds
FROM track
WHERE milliseconds > (
SELECT AVG(milliseconds) AS avg_track_length
FROM track)
ORDER BY milliseconds DESC;

SELECT AVG(milliseconds) FROM track;

SELECT NAME,milliseconds
FROM track
WHERE milliseconds > 393599
ORDER BY milliseconds DESC;
```

Data Output Messages Notifications

	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
12	The Magnificent Warriors	2924716
13	The Living Legend, Pt. 1	2924507
14	The Gun On Ice Planet Zero, Pt. 2	2924341
15	The Hand of God	2924007
16	Experiment In Terra	2923548
17	War of the Gods, Pt. 2	2923381
18	The Living Legend, Pt. 2	2923298
19	War of the Gods, Pt. 1	2922630
20	Lost Planet of the Gods, Pt. 1	2922547
21	Baltar's Escape	2922088
22	The Lost Warrior	2920045
23	Lost Planet of the Gods, Pt. 2	2914664
24	The Gun On Ice Planet Zero, Pt. 1	2907615
25	Greetings from Earth, Pt. 2	2903778
26	Crossroads, Pt. 2	2869953
27	The Young Lords	2863571
28	Dave	2825166
29	?	2782333
30	Maternity Leave	2780416
31	Three Minutes	2763666
32	Hero	2713755
33	One of Them	2698791
34	How to Stop an Exploding Man	2687103
35	The Long Con	2679583
36	Live Together, Die Alone, Pt. 2	2656531
37	S.O.S.	2639541
38	One of Us	2638096
39	The Man from Tallahassee	2637637
40	The Cost of Living	2637500
41	The Glass Ballerina	2637458
42	Every Man for Himself	2637387
43	Not In Portland	2637345
44	Not In Portland	2637303
45	A Tale of Two Cities	2636970
46	Flashes Before Your Eyes	2636636
47	Stranger In a Strange Land	2636428
48	Left Behind	2635343
49	Tricia Tanaka Is Dead	2635010
50	Lost Survival Guide	2632590

```sql
/* Q9: 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, you
will need to use the Invoice, InvoiceLine, 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 InvoiceLine table to find out how many of each product was purchased,
and then multiply this by the price for each artist. */

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;
```

Data Output Messages Notifications

	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	27.719999999999985
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
17	30	Edward	Francis	Queen	1.98
18	35	Madalena	Sampaio	Queen	1.98
19	36	Hannah	Schneider	Queen	1.98
20	11	Alexandre	Rocha	Queen	1.98
21	8	Daan	Peeters	Queen	1.98
22	42	Wyatt	Girard	Queen	1.98
23	44	Terhi	Hämäläinen	Queen	1.98
24	1	Luís	Gonçalves	Queen	1.98
25	48	Johannes	Van der Berg	Queen	1.98
26	49	Stanisław	Wójcik	Queen	1.98
27	52	Emma	Jones	Queen	1.98
28	57	Luis	Rojas	Queen	1.98
29	15	Jennifer	Peterson	Queen	1.98
30	28	Julia	Barnett	Queen	1.98
31	27	Patrick	Gray	Queen	0.99
32	58	Manoj	Pareek	Queen	0.99
33	45	Ladislav	Kovács	Queen	0.99
34	26	Richard	Cunningham	Queen	0.99
35	59	Puja	Srivastava	Queen	0.99
36	13	Fernanda	Ramos	Queen	0.99
37	6	Helena	Holý	Queen	0.99
38	22	Heather	Leacock	Queen	0.99
39	19	Tim	Goyer	Queen	0.99
40	39	Camille	Bernard	Queen	0.99
41	55	Mark	Taylor	Queen	0.99
42	50	Enrique	Muñoz	Queen	0.99
43	43	Isabelle	Mercier	Queen	0.99

```sql
/* Q10: 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 and second need
data at country level. */

-- Method 1: Using CTE

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 RowNo
    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 RowNo <= 1

```

```sql
-- Method 2: : Using Recursive

WITH RECURSIVE
    sales_per_country AS(
        SELECT COUNT(*) AS purchases_per_genre, customer.country, genre.name, genre.genre_id
        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
    ),
    max_genre_per_country AS (SELECT MAX(purchases_per_genre) AS max_genre_number, country
        FROM sales_per_country
        GROUP BY 2
        ORDER BY 2)

SELECT sales_per_country.*
FROM sales_per_country
JOIN max_genre_per_country ON sales_per_country.country = max_genre_per_country.country
WHERE sales_per_country.purchases_per_genre = max_genre_per_country.max_genre_number;
```

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	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
11	211	France	Rock	1	1
12	194	Germany	Rock	1	1
13	44	Hungary	Rock	1	1
14	102	India	Rock	1	1
15	72	Ireland	Rock	1	1
16	35	Italy	Rock	1	1
17	33	Netherlands	Rock	1	1
18	40	Norway	Rock	1	1
19	40	Poland	Rock	1	1
20	108	Portugal	Rock	1	1
21	46	Spain	Rock	1	1
22	60	Sweden	Rock	1	1
23	166	United Kingdom	Rock	1	1
24	561	USA	Rock	1	1

```sql
/* Q11: 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:  Similar to the above question. There are two parts in question-
first find the most spent on music for each country and second filter the data for respective customers. */

-- Method 1: using CTE

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 RowNo
        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 RowNo <= 1
```

```sql
-- Method 2: Using Recursive

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;

-- Thank You :)
```

Data Output Messages Notifications

	customer_id integer	first_name character (50)	last_name character (50)	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	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
10	44	Terhi	Hämäläinen	Finland	79.2	1
11	42	Wyatt	Girard	France	99.99	1
12	37	Fynn	Zimmermann	Germany	94.05000000000001	1
13	45	Ladislav	Kovács	Hungary	78.21	1
14	58	Manoj	Pareek	India	111.86999999999999	1
15	46	Hugh	O'Reilly	Ireland	114.83999999999997	1
16	47	Lucas	Mancini	Italy	50.49	1
17	48	Johannes	Van der Berg	Netherlands	65.34	1
18	4	Bjørn	Hansen	Norway	72.27000000000001	1
19	49	Stanisław	Wójcik	Poland	76.22999999999999	1
20	34	João	Fernandes	Portugal	102.96000000000001	1
21	50	Enrique	Muñoz	Spain	98.01	1
22	51	Joakim	Johansson	Sweden	75.24	1
23	53	Phil	Hughes	United Kingdom	98.01	1
24	17	Jack	Smith	USA	98.01	1

THANK YOU

Presented by:
Fayyas KP