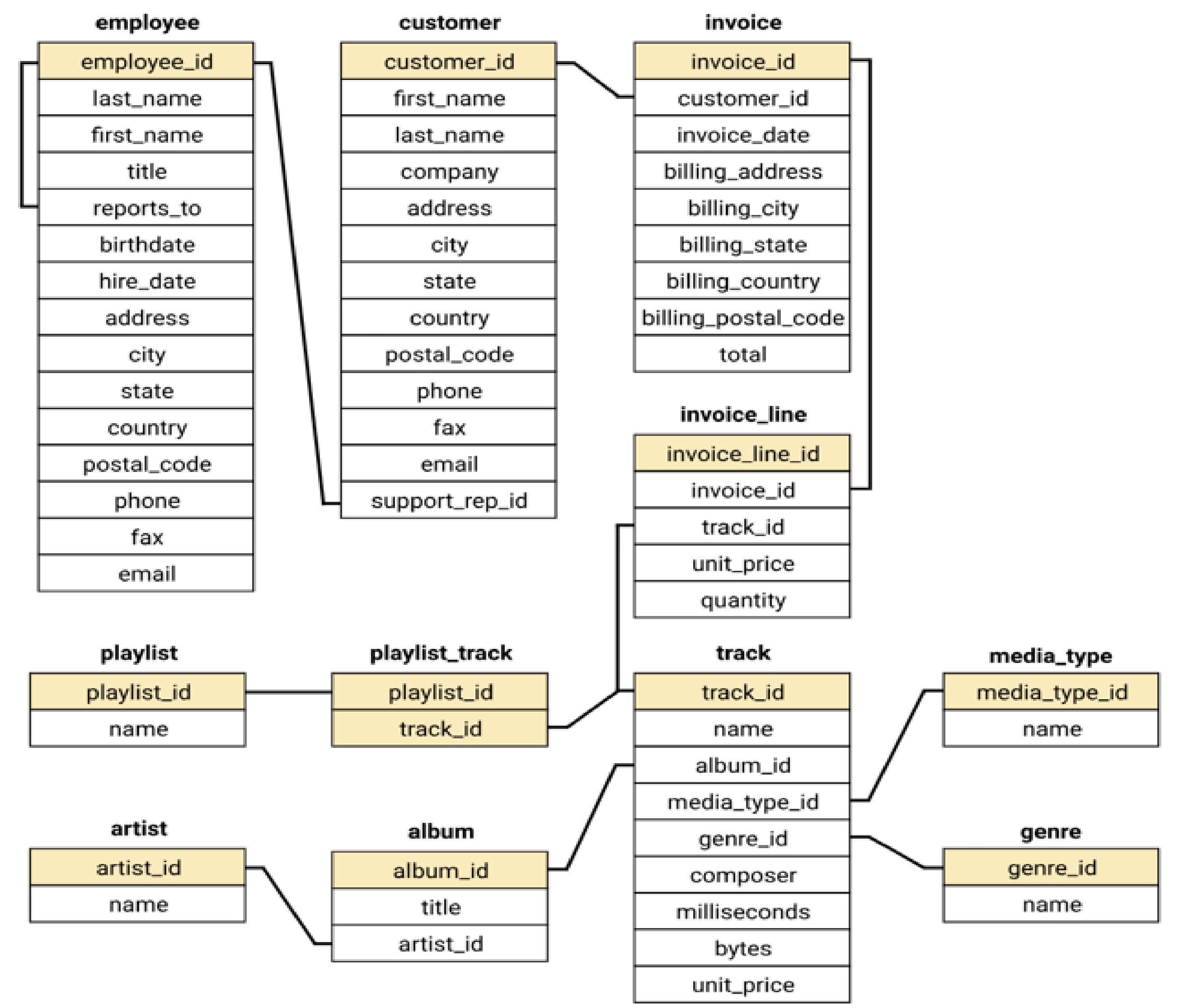
# MUSIC STORE ANALYSIS

~A MySQL Project Report

# AGENDA

Examine the datasets with MySQL and help the store to understand its business growth by solving the business requirements

## 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

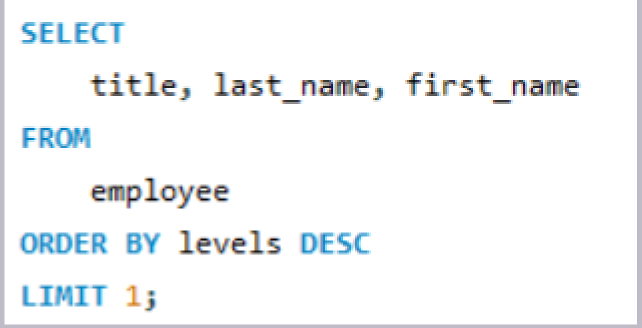
# IMPORTANT TERMS AND CONCEPTS USED

WINDOWS,PARTITION,ROW_NUMBER,

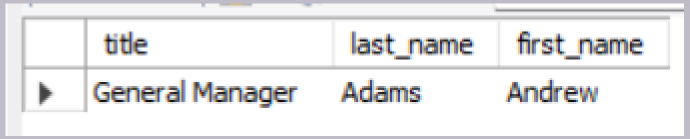
CTE, RECURSIVE CTE,

JOIN,ORDER BY, GROUP BY

# Q1: WHO IS THE SENIOR MOST EMPLOYEE BASED ON JOB TITLE?

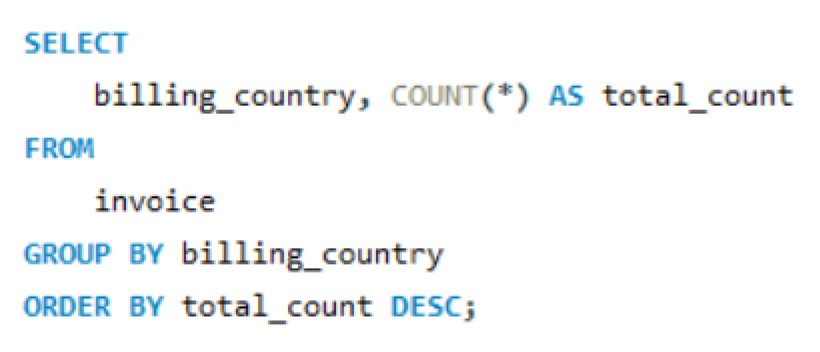


```
SELECT
    title, last_name, first_name
FROM
    employee
ORDER BY levels DESC
LIMIT 1;
```

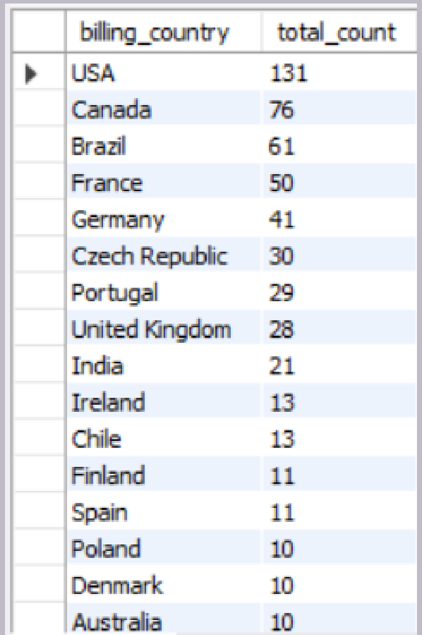


| title | last_name | first_name |
|---|---|---|
| General Manager | Adams | Andrew |

# Q2: WHICH COUNTRIES HAVE THE MOST INVOICES?

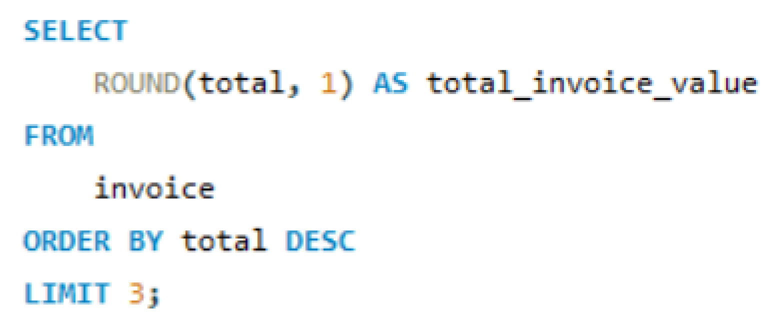


```
SELECT
    billing_country, COUNT(*) AS total_count
FROM
    invoice
GROUP BY billing_country
ORDER BY total_count DESC;
```

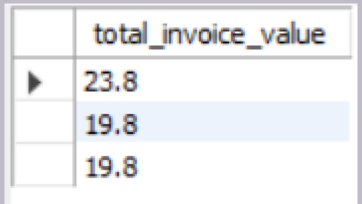


| billing_country | total_count |
|---|---|
| USA | 131 |
| Canada | 76 |
| Brazil | 61 |
| France | 50 |
| Germany | 41 |
| Czech Republic | 30 |
| Portugal | 29 |
| United Kingdom | 28 |
| India | 21 |
| Ireland | 13 |
| Chile | 13 |
| Finland | 11 |
| Spain | 11 |
| Poland | 10 |
| Denmark | 10 |
| Australia | 10 |

# Q3: WHAT ARE TOP 3 VALUES OF TOTAL INVOICE?

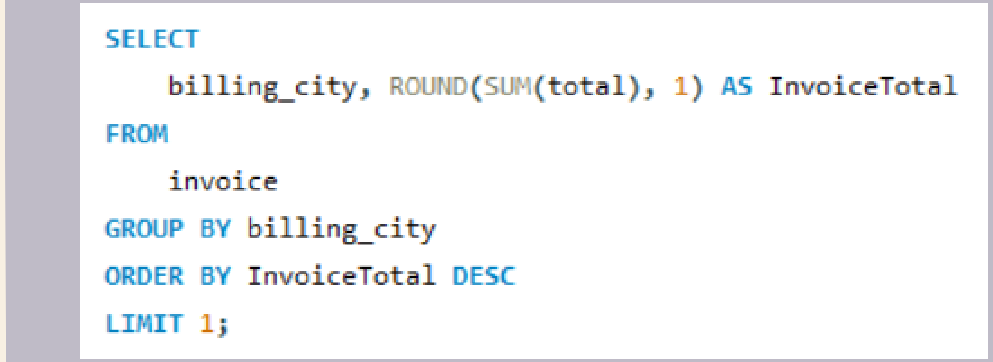


```
SELECT
    ROUND(total, 1) AS total_invoice_value
FROM
    invoice
ORDER BY total DESC
LIMIT 3;
```

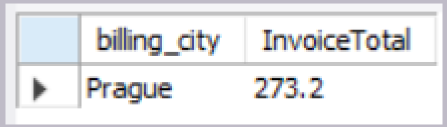


| total_invoice_value |
|---|
| 23.8 |
| 19.8 |
| 19.8 |

# Q4: WHICH CITY HAS THE BEST CUSTOMERS? 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

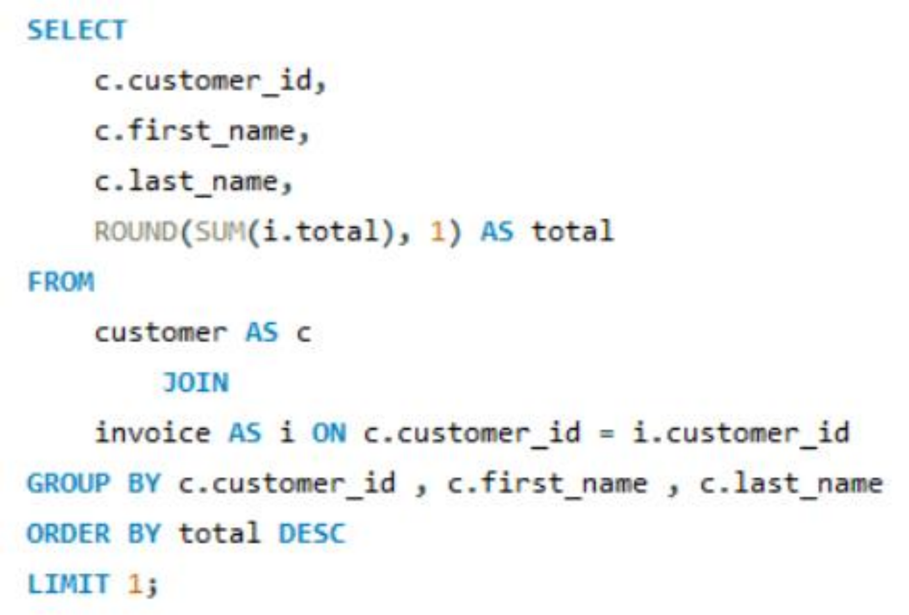


```sql
SELECT
    billing_city, ROUND(SUM(total), 1) AS InvoiceTotal
FROM
    invoice
GROUP BY billing_city
ORDER BY InvoiceTotal DESC
LIMIT 1;
```

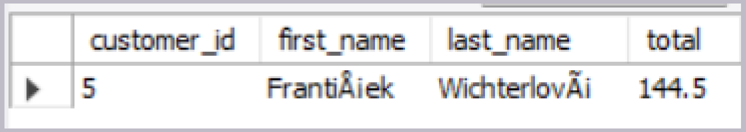


| | billing_city | InvoiceTotal |
|---|---|---|
| ▶ | Prague | 273.2 |

# Q5: WHO IS THE BEST CUSTOMER? WRITE A QUERY THAT RETURNS THE PERSON WHO HAS SPENT THE MOST MONEY.

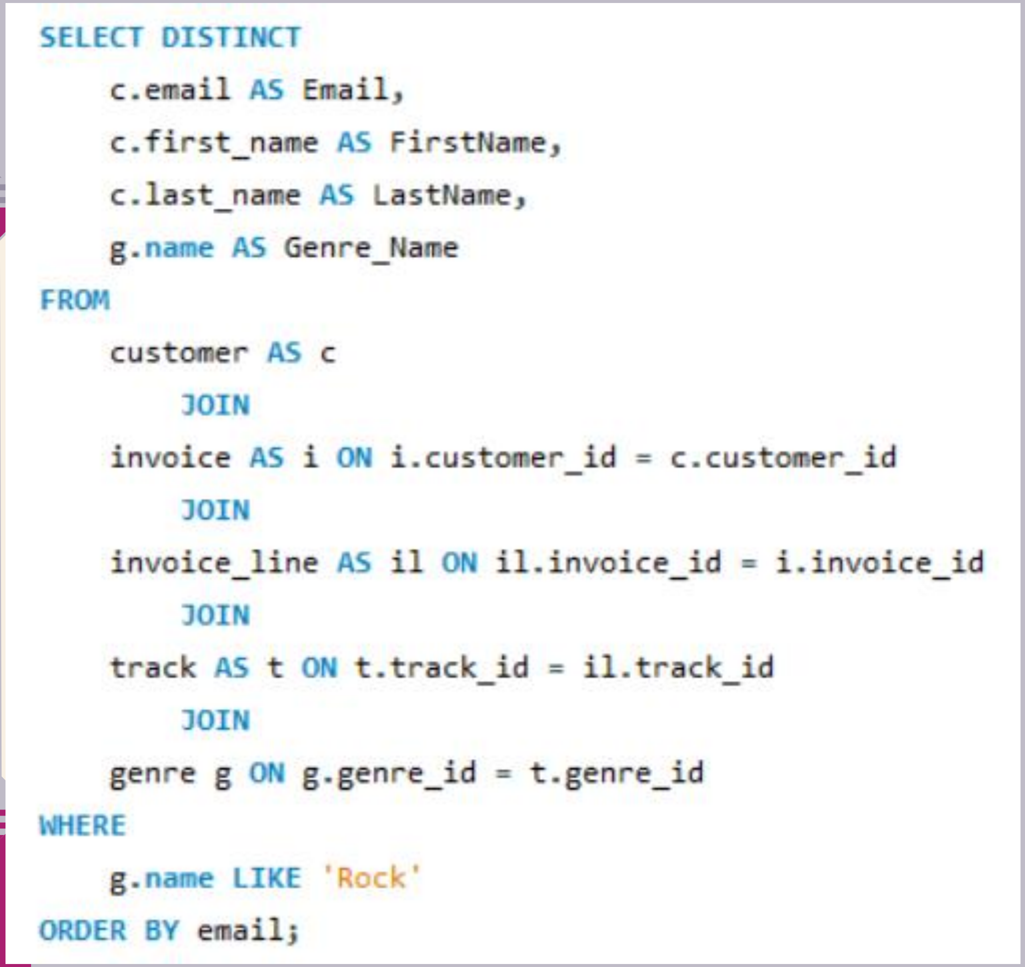


```
SELECT
    c.customer_id,
    c.first_name,
    c.last_name,
    ROUND(SUM(i.total), 1) AS total
FROM
    customer AS c
        JOIN
    invoice AS i ON c.customer_id = i.customer_id
GROUP BY c.customer_id , c.first_name , c.last_name
ORDER BY total DESC
LIMIT 1;
```

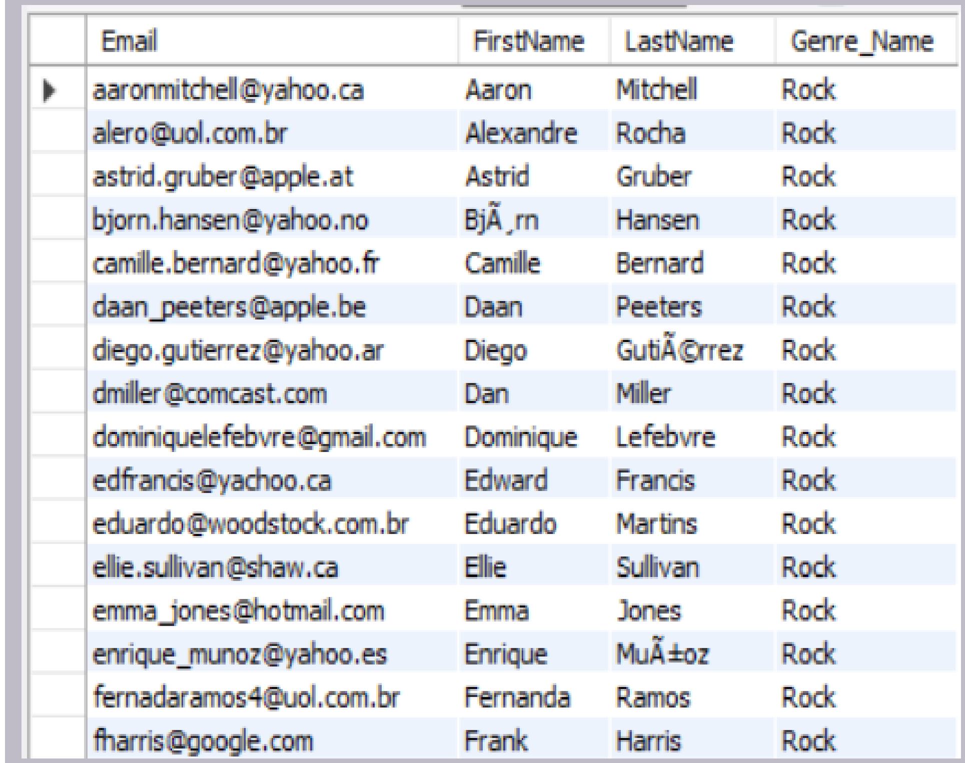


| | customer_id | first_name | last_name | total |
|---|---|---|---|---|
| ▶ | 5 | FrantiÅiek | WichterlovÃi | 144.5 |

# Q6: WRITE QUERY TO RETURN THE EMAIL, FIRST NAME, LAST NAME AND GENRE OF ALL ROCK MUSIC LISTENERS. RETURN YOUR LIST ORDERED ALPHABETICALLY BY EMAIL STARTING WITH A.

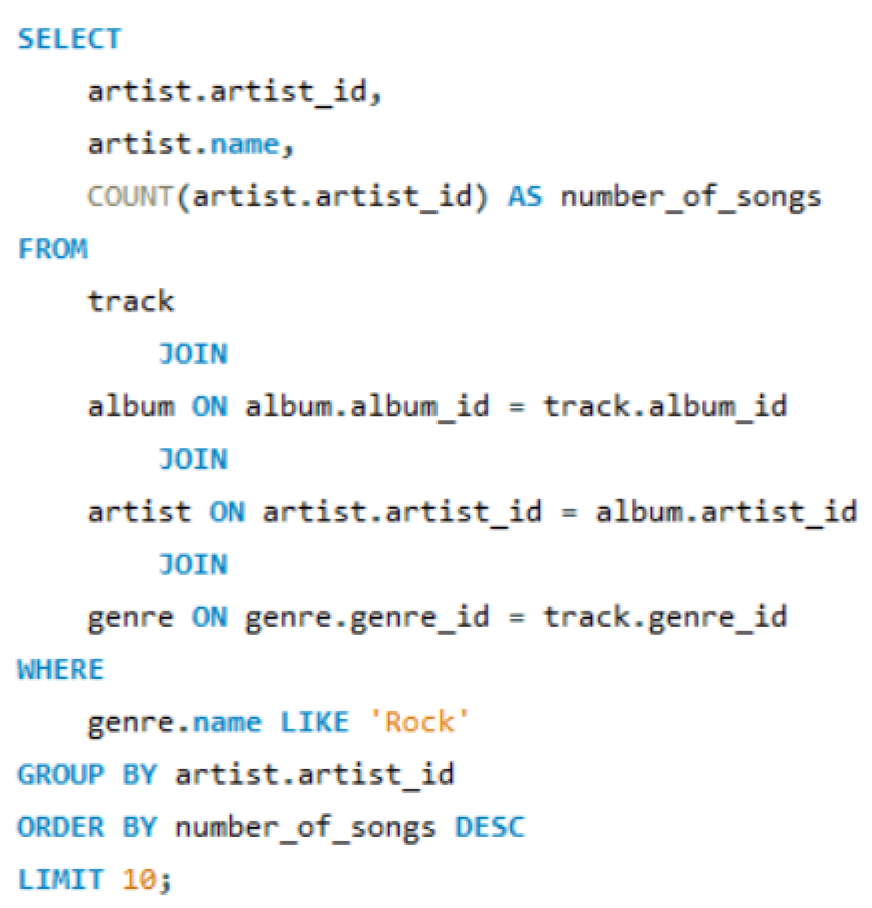


```sql
SELECT DISTINCT
    c.email AS Email,
    c.first_name AS FirstName,
    c.last_name AS LastName,
    g.name AS Genre_Name
FROM
    customer AS c
        JOIN
    invoice AS i ON i.customer_id = c.customer_id
        JOIN
    invoice_line AS il ON il.invoice_id = i.invoice_id
        JOIN
    track AS 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;
```

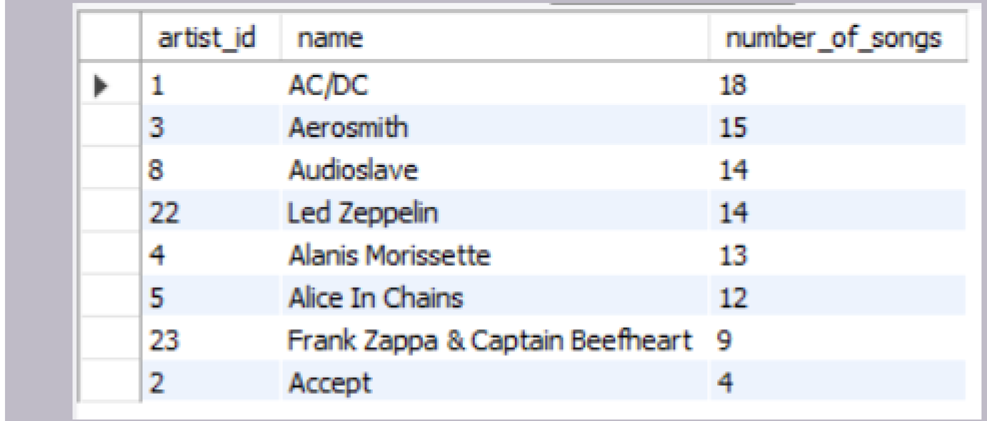


| | Email | FirstName | LastName | Genre_Name |
|---|---|---|---|---|
| ▶ | aaronmitchell@yahoo.ca | Aaron | Mitchell | Rock |
| | alero@uol.com.br | Alexandre | Rocha | Rock |
| | astrid.gruber@apple.at | Astrid | Gruber | Rock |
| | bjorn.hansen@yahoo.no | BjÃ¸rn | Hansen | Rock |
| | camille.bernard@yahoo.fr | Camille | Bernard | Rock |
| | daan_peeters@apple.be | Daan | Peeters | Rock |
| | diego.gutierrez@yahoo.ar | Diego | GutiÃ©rrez | Rock |
| | dmiller@comcast.com | Dan | Miller | Rock |
| | dominiquelefebvre@gmail.com | Dominique | Lefebvre | Rock |
| | edfrancis@yachoo.ca | Edward | Francis | Rock |
| | eduardo@woodstock.com.br | Eduardo | Martins | Rock |
| | ellie.sullivan@shaw.ca | Ellie | Sullivan | Rock |
| | emma_jones@hotmail.com | Emma | Jones | Rock |
| | enrique_munoz@yahoo.es | Enrique | MuÃ±oz | Rock |
| | fernadaramos4@uol.com.br | Fernanda | Ramos | Rock |
| | fharris@google.com | Frank | Harris | Rock |

# Q7: WRITE A QUERY THAT RETURNS THE ARTIST NAME AND TOTAL TRACK COUNT OF THE TOP 10 ROCK BANDS.

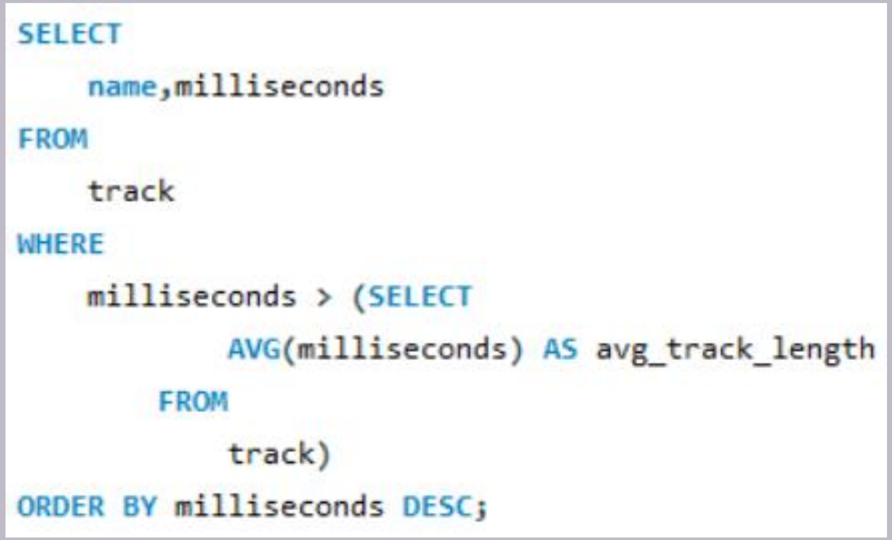


```
SELECT
    artist.artist_id,
    artist.name,
    COUNT(artist.artist_id) AS number_of_songs
FROM
    track
        JOIN
    album ON album.album_id = track.album_id
        JOIN
    artist ON artist.artist_id = album.artist_id
        JOIN
    genre ON genre.genre_id = track.genre_id
WHERE
    genre.name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

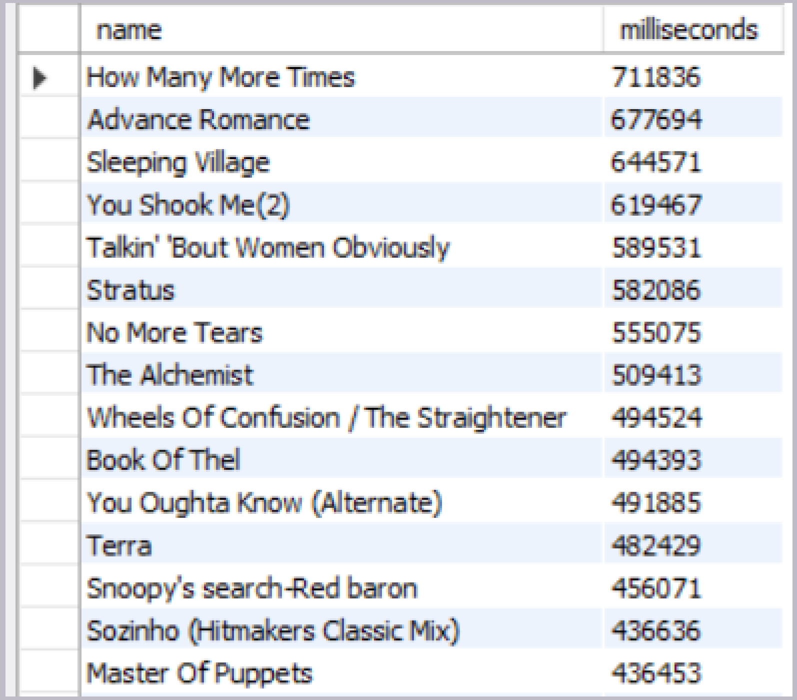


| | artist_id | name | number_of_songs |
|---|---|---|---|
| ▶ | 1 | AC/DC | 18 |
| | 3 | Aerosmith | 15 |
| | 8 | Audioslave | 14 |
| | 22 | Led Zeppelin | 14 |
| | 4 | Alanis Morissette | 13 |
| | 5 | Alice In Chains | 12 |
| | 23 | Frank Zappa & Captain Beefheart | 9 |
| | 2 | Accept | 4 |

# 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.

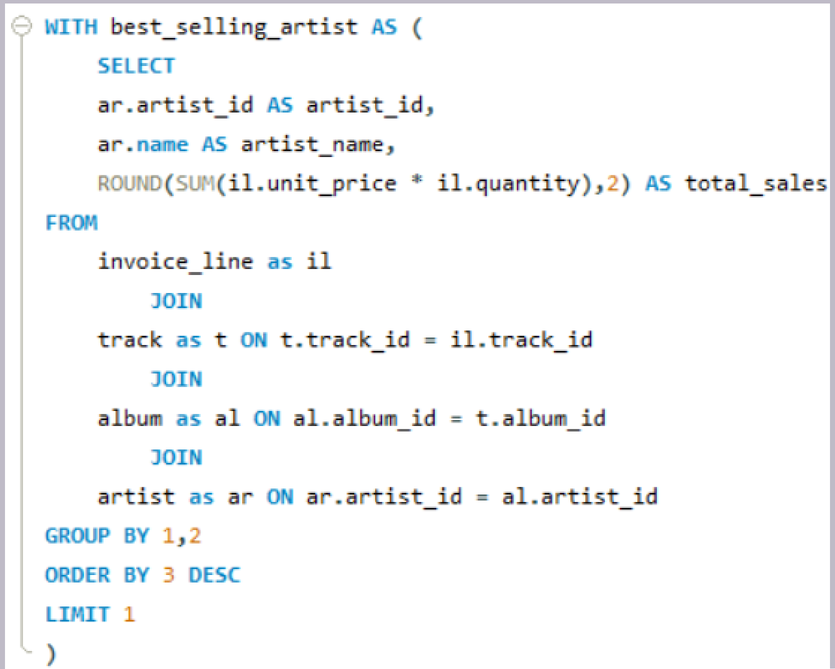


```
SELECT
    name,milliseconds
FROM
    track
WHERE
    milliseconds > (SELECT
            AVG(milliseconds) AS avg_track_length
        FROM
            track)
ORDER BY milliseconds DESC;
```

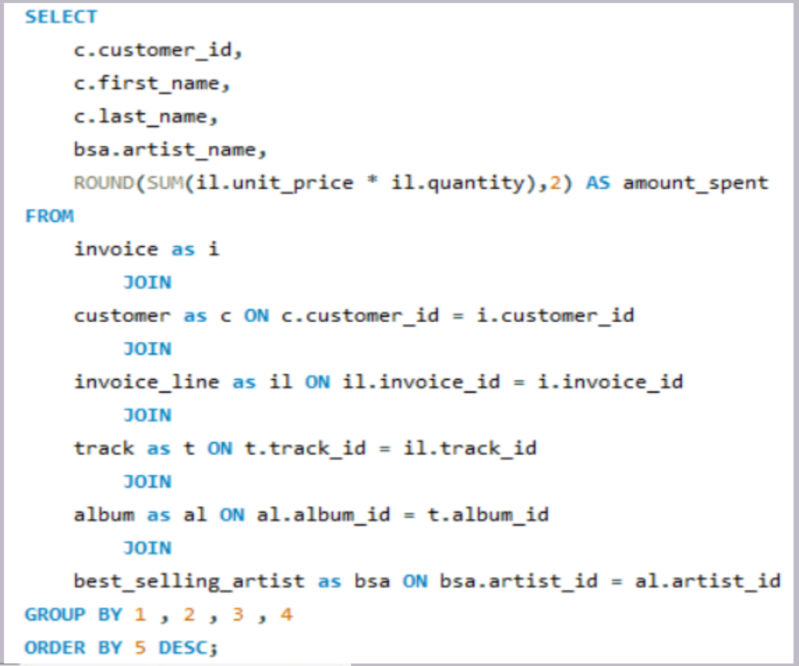


| name | milliseconds |
|---|---|
| How Many More Times | 711836 |
| Advance Romance | 677694 |
| Sleeping Village | 644571 |
| You Shook Me(2) | 619467 |
| Talkin' 'Bout Women Obviously | 589531 |
| Stratus | 582086 |
| No More Tears | 555075 |
| The Alchemist | 509413 |
| Wheels Of Confusion / The Straightener | 494524 |
| Book Of Thel | 494393 |
| You Oughta Know (Alternate) | 491885 |
| Terra | 482429 |
| Snoopy's search-Red baron | 456071 |
| Sozinho (Hitmakers Classic Mix) | 436636 |
| Master Of Puppets | 436453 |

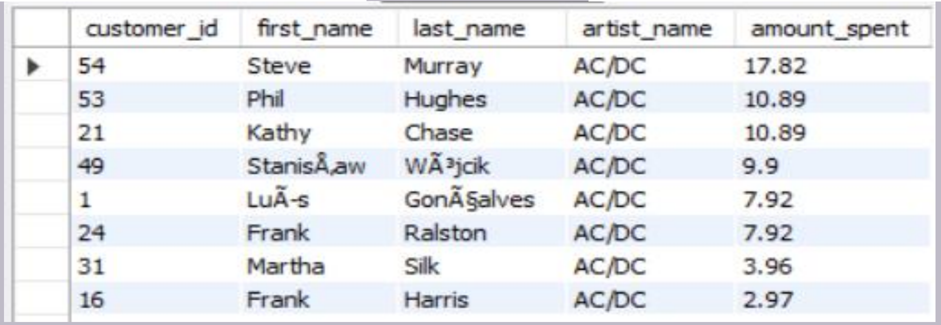
# Q9: FIND HOW MUCH AMOUNT SPENT BY EACH CUSTOMER ON ARTISTS? WRITE A QUERY TO RETURN CUSTOMER NAME, ARTIST NAME AND TOTAL SPENT



```sql
WITH best_selling_artist AS (
    SELECT
    ar.artist_id AS artist_id,
    ar.name AS artist_name,
    ROUND(SUM(il.unit_price * il.quantity),2) AS total_sales
FROM
    invoice_line as il
        JOIN
    track as t ON t.track_id = il.track_id
        JOIN
    album as al ON al.album_id = t.album_id
        JOIN
    artist as ar ON ar.artist_id = al.artist_id
GROUP BY 1,2
ORDER BY 3 DESC
LIMIT 1
)
```



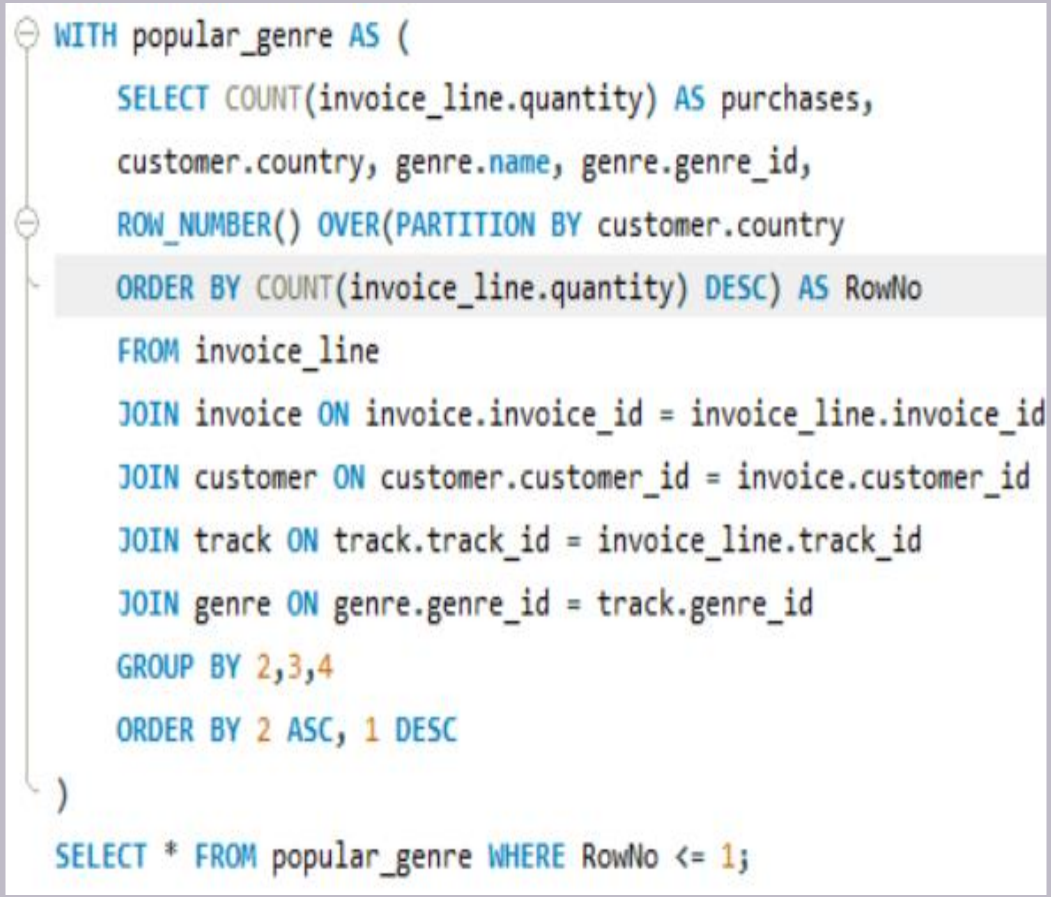
```sql
SELECT
    c.customer_id,
    c.first_name,
    c.last_name,
    bsa.artist_name,
    ROUND(SUM(il.unit_price * il.quantity),2) AS amount_spent
FROM
    invoice as i
        JOIN
    customer as c ON c.customer_id = i.customer_id
        JOIN
    invoice_line as il ON il.invoice_id = i.invoice_id
        JOIN
    track as t ON t.track_id = il.track_id
        JOIN
    album as al ON al.album_id = t.album_id
        JOIN
    best_selling_artist as bsa ON bsa.artist_id = al.artist_id
GROUP BY 1 , 2 , 3 , 4
ORDER BY 5 DESC;
```



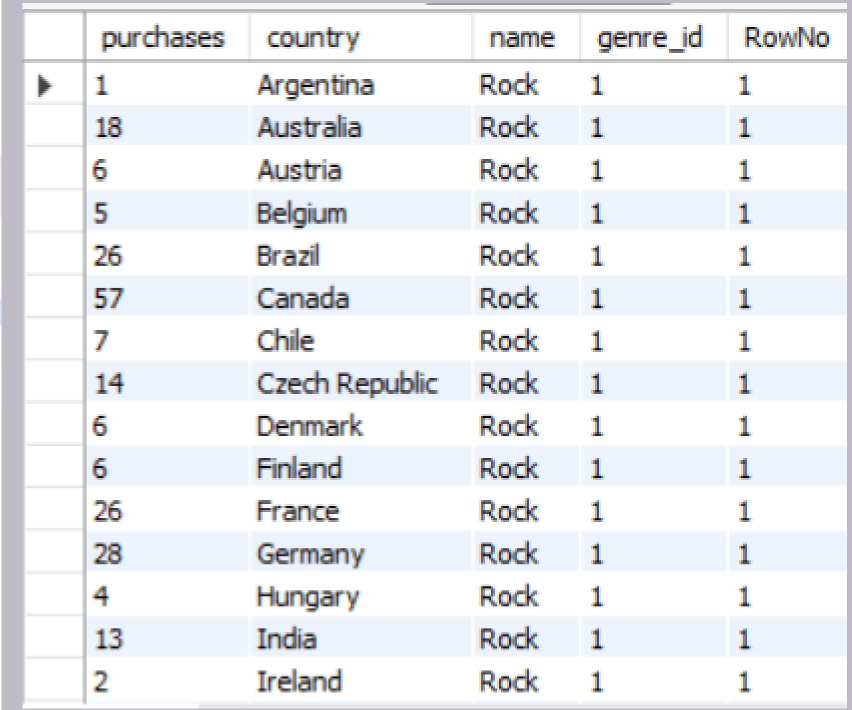
| | customer_id | first_name | last_name | artist_name | amount_spent |
|---|---|---|---|---|---|
| ▶ | 54 | Steve | Murray | AC/DC | 17.82 |
| | 53 | Phil | Hughes | AC/DC | 10.89 |
| | 21 | Kathy | Chase | AC/DC | 10.89 |
| | 49 | StanisÅ‚aw | WÃ³jcik | AC/DC | 9.9 |
| | 1 | LuÃ-s | GonÃ§alves | AC/DC | 7.92 |
| | 24 | Frank | Ralston | AC/DC | 7.92 |
| | 31 | Martha | Silk | AC/DC | 3.96 |
| | 16 | Frank | Harris | AC/DC | 2.97 |

# Q10: 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.

Method 1: Using CTE



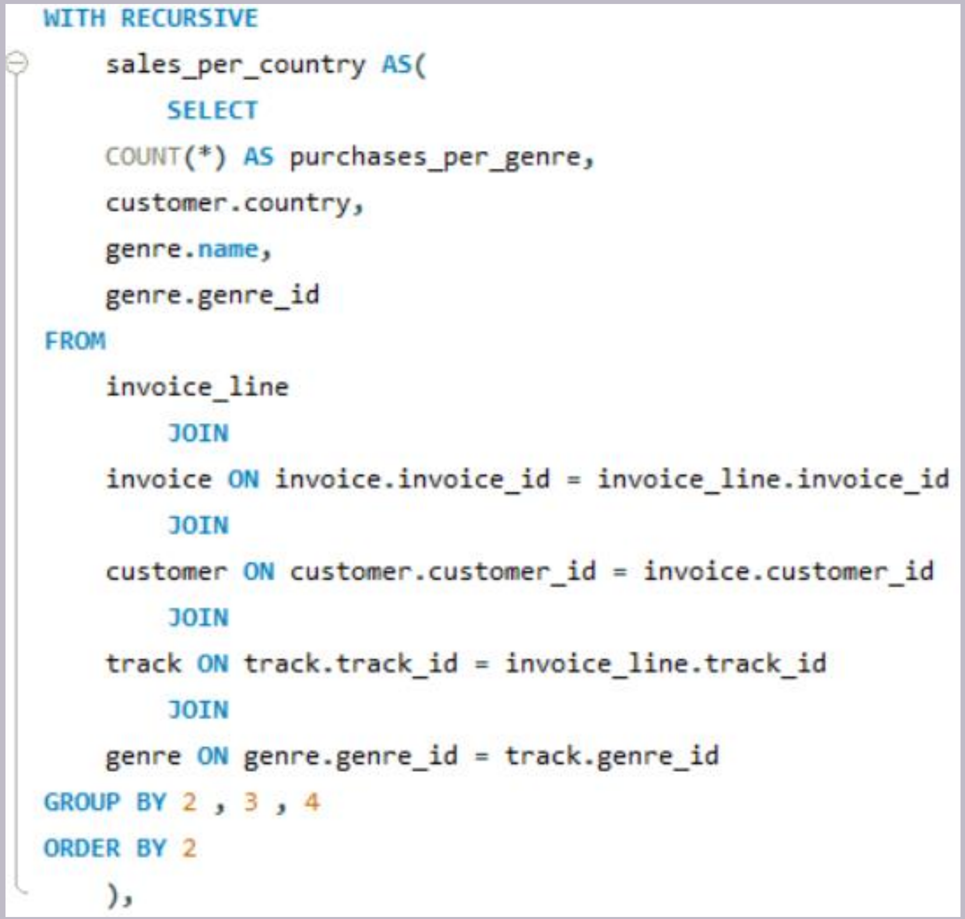
```
WITH popular_genre AS (
    SELECT COUNT(invoice_line.quantity) AS purchases,
    customer.country, genre.name, genre.genre_id,
    ROW_NUMBER() OVER(PARTITION BY customer.country
    ORDER BY COUNT(invoice_line.quantity) DESC) AS RowNo
    FROM invoice_line
    JOIN invoice ON invoice.invoice_id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



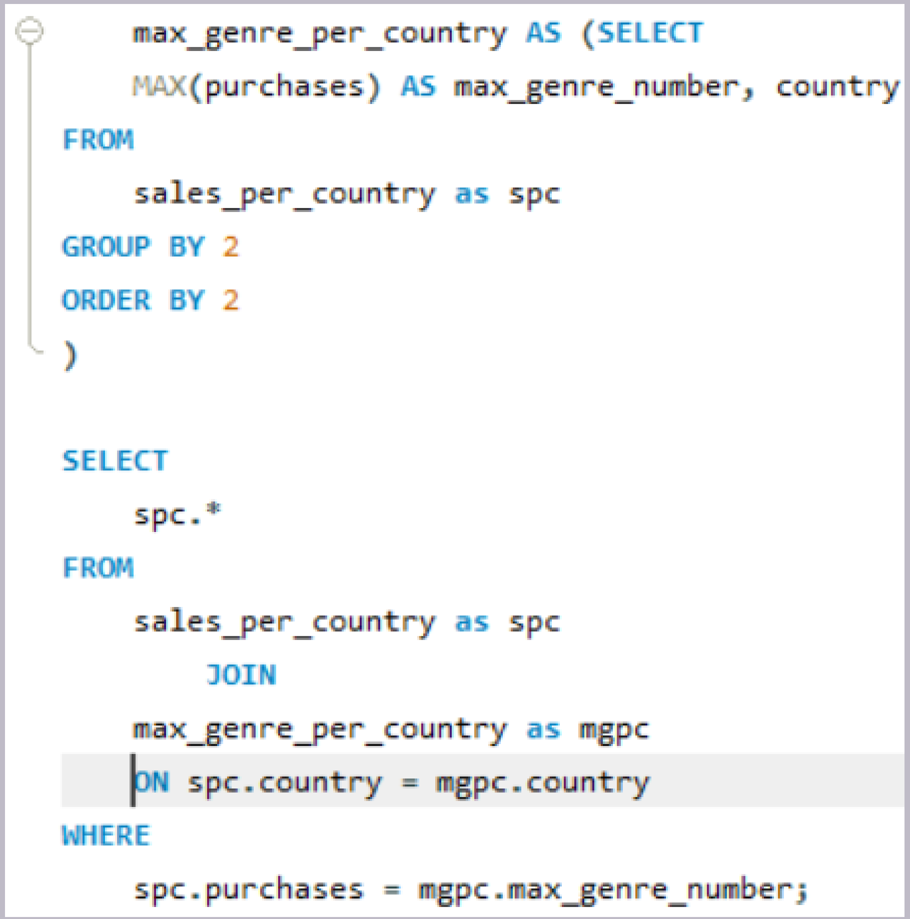
| purchases | country | name | genre_id | RowNo |
|---|---|---|---|---|
| 1 | Argentina | Rock | 1 | 1 |
| 18 | Australia | Rock | 1 | 1 |
| 6 | Austria | Rock | 1 | 1 |
| 5 | Belgium | Rock | 1 | 1 |
| 26 | Brazil | Rock | 1 | 1 |
| 57 | Canada | Rock | 1 | 1 |
| 7 | Chile | Rock | 1 | 1 |
| 14 | Czech Republic | Rock | 1 | 1 |
| 6 | Denmark | Rock | 1 | 1 |
| 6 | Finland | Rock | 1 | 1 |
| 26 | France | Rock | 1 | 1 |
| 28 | Germany | Rock | 1 | 1 |
| 4 | Hungary | Rock | 1 | 1 |
| 13 | India | Rock | 1 | 1 |
| 2 | Ireland | Rock | 1 | 1 |

# Q10: 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.

Method 2: : Using Recursive



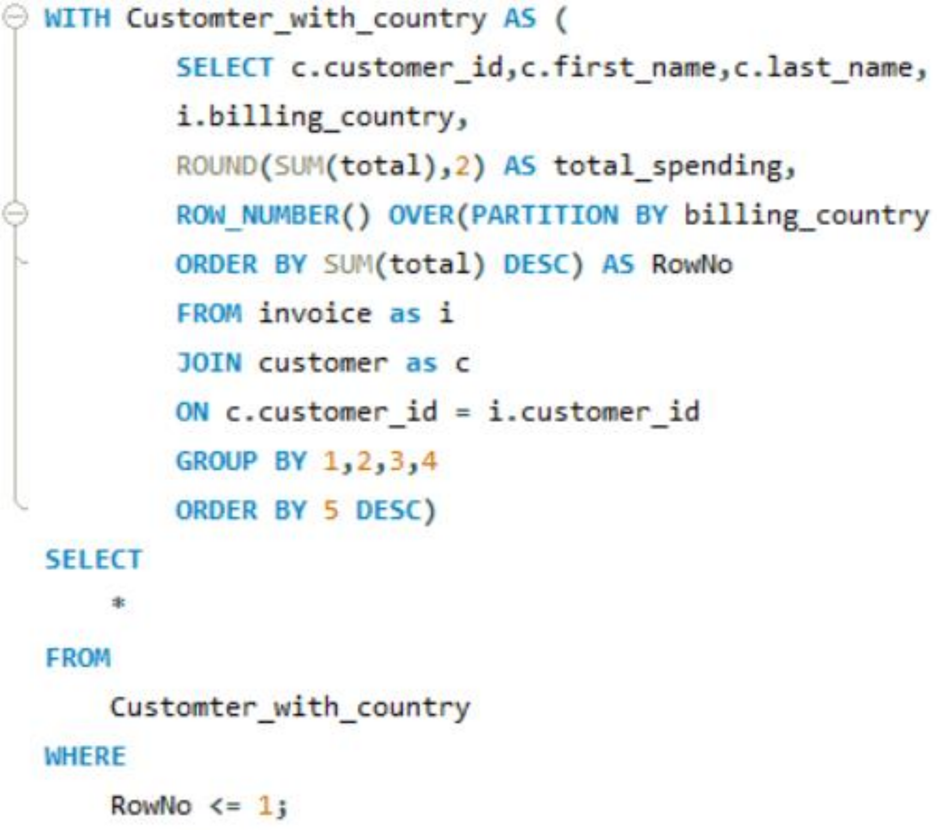
```sql
WITH RECURSIVE
    sales_per_country AS(
        SELECT
    COUNT(*) AS purchases_per_genre,
    customer.country,
    genre.name,
    genre.genre_id
FROM
    invoice_line
        JOIN
    invoice ON invoice.invoice_id = invoice_line.invoice_id
        JOIN
    customer ON customer.customer_id = 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
    ),
```



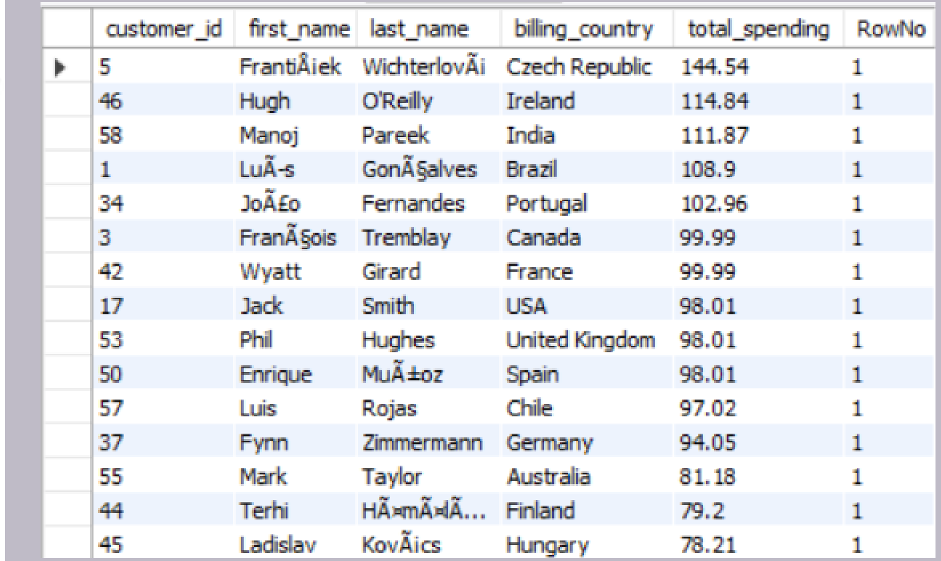
```sql
    max_genre_per_country AS (SELECT
    MAX(purchases) AS max_genre_number, country
FROM
    sales_per_country as spc
GROUP BY 2
ORDER BY 2
)

SELECT
    spc.*
FROM
    sales_per_country as spc
        JOIN
    max_genre_per_country as mgpc
    ON spc.country = mgpc.country
WHERE
    spc.purchases = mgpc.max_genre_number;
```

# 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.



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



| | customer_id | first_name | last_name | billing_country | total_spending | RowNo |
|---|---|---|---|---|---|---|
| ▶ | 5 | FrantiÅiek | WichterlovÃi | Czech Republic | 144.54 | 1 |
| | 46 | Hugh | O'Reilly | Ireland | 114.84 | 1 |
| | 58 | Manoj | Pareek | India | 111.87 | 1 |
| | 1 | LuÃ-s | GonÃ§alves | Brazil | 108.9 | 1 |
| | 34 | JoÃ£o | Fernandes | Portugal | 102.96 | 1 |
| | 3 | FranÃ§ois | Tremblay | Canada | 99.99 | 1 |
| | 42 | Wyatt | Girard | France | 99.99 | 1 |
| | 17 | Jack | Smith | USA | 98.01 | 1 |
| | 53 | Phil | Hughes | United Kingdom | 98.01 | 1 |
| | 50 | Enrique | MuÃ±oz | Spain | 98.01 | 1 |
| | 57 | Luis | Rojas | Chile | 97.02 | 1 |
| | 37 | Fynn | Zimmermann | Germany | 94.05 | 1 |
| | 55 | Mark | Taylor | Australia | 81.18 | 1 |
| | 44 | Terhi | HÃ¤mÃ¤lÃ... | Finland | 79.2 | 1 |
| | 45 | Ladislav | KovÃics | Hungary | 78.21 | 1 |

# THANK YOU

Alok Ranjan Swain

cool.alokranjan@gmail.com

GitHub