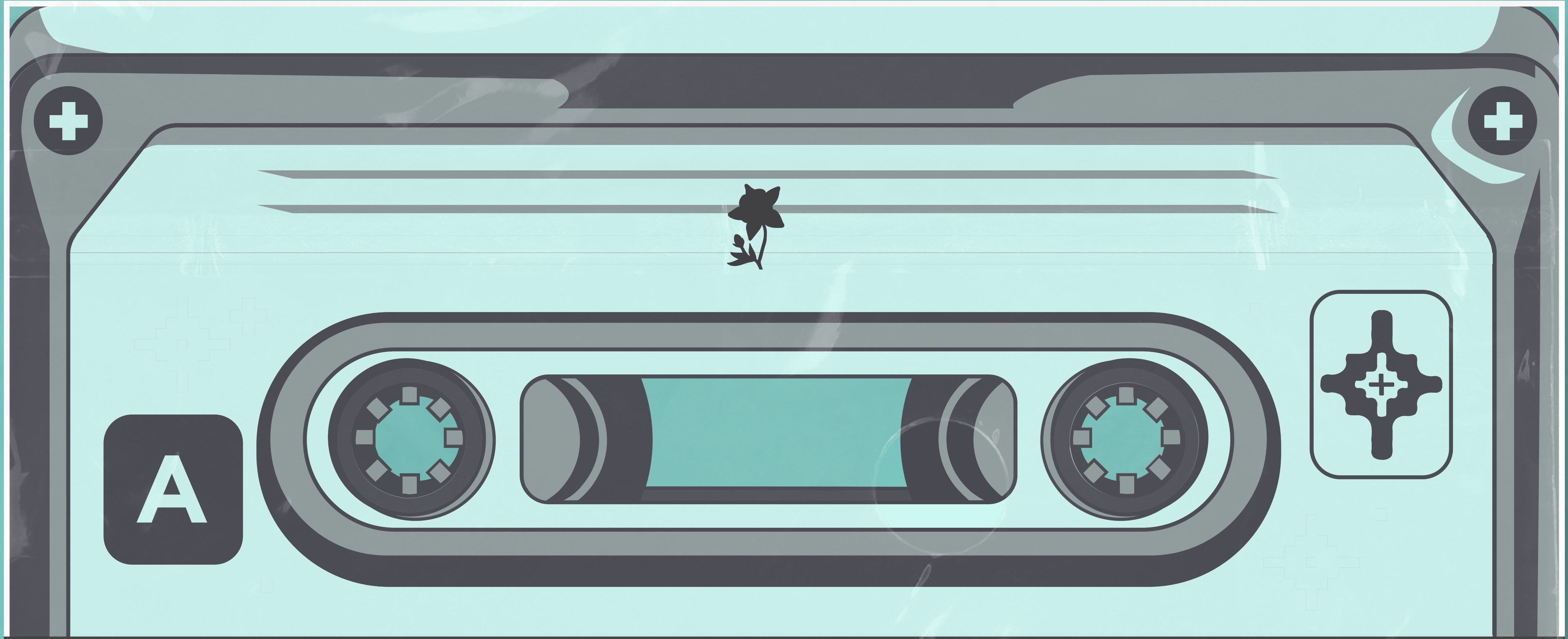


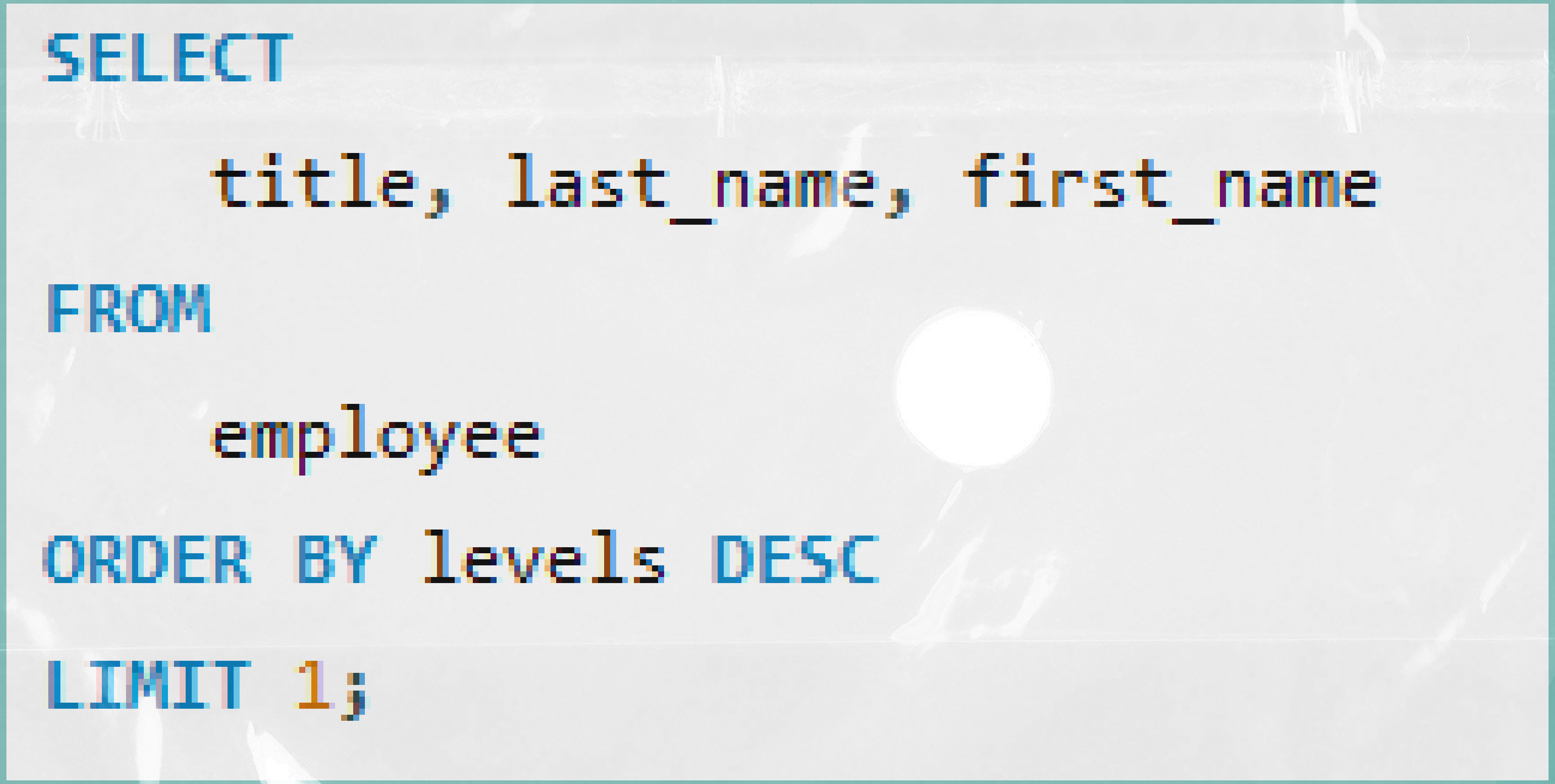
MUSIC PLAYER

JYOTI BAWA

INTRODUCTION

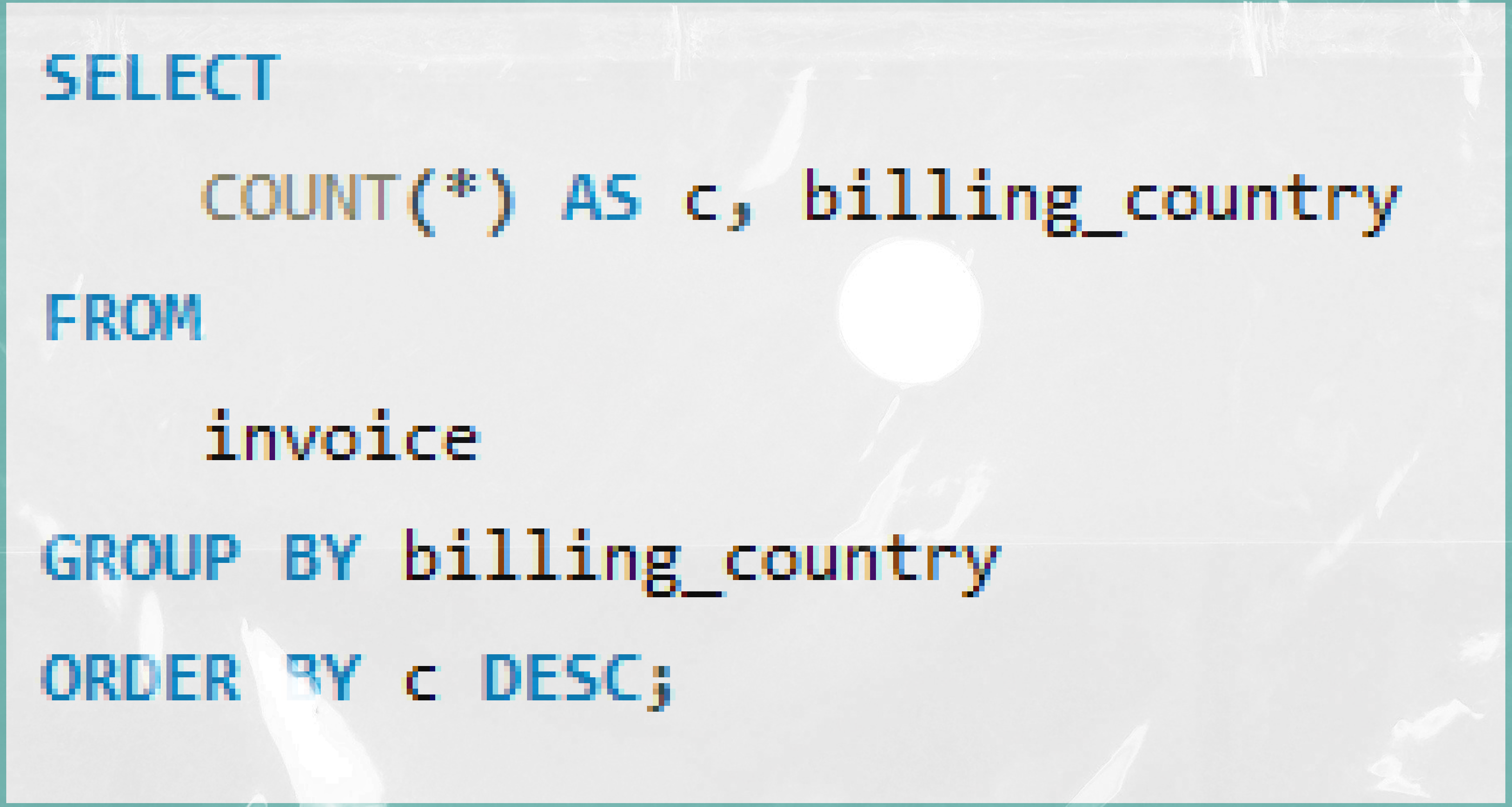
This SQL project is built on a Music Store database to explore real-world data insights using structured queries. It covers key concepts like JOINS, CTEs, aggregation, window functions, and subqueries across three levels: basic, moderate, and advanced.
The project analyzes customer behavior, top-selling artists, most popular genres per country, and revenue patterns — helping sharpen my skills in data analysis, query optimization, and relational database design.

WHO IS THE SENIOR MOST EMPLOYEE BASED ON JOB TITLE?



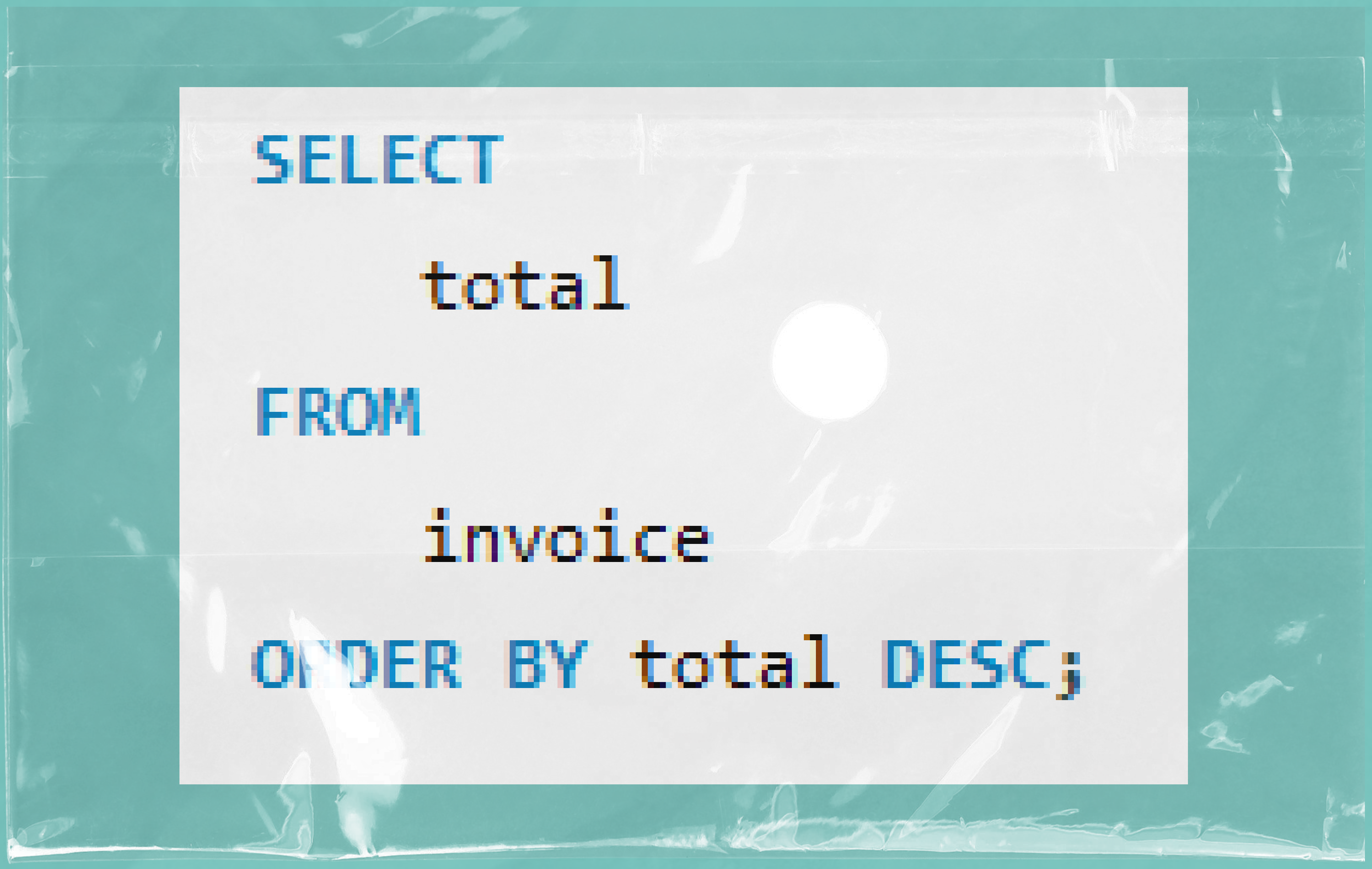
```
SELECT
    title, last_name, first_name
FROM
    employee
ORDER BY levels DESC
LIMIT 1;
```

WHICH COUNTRIES HAVE THE MOST INVOICES?



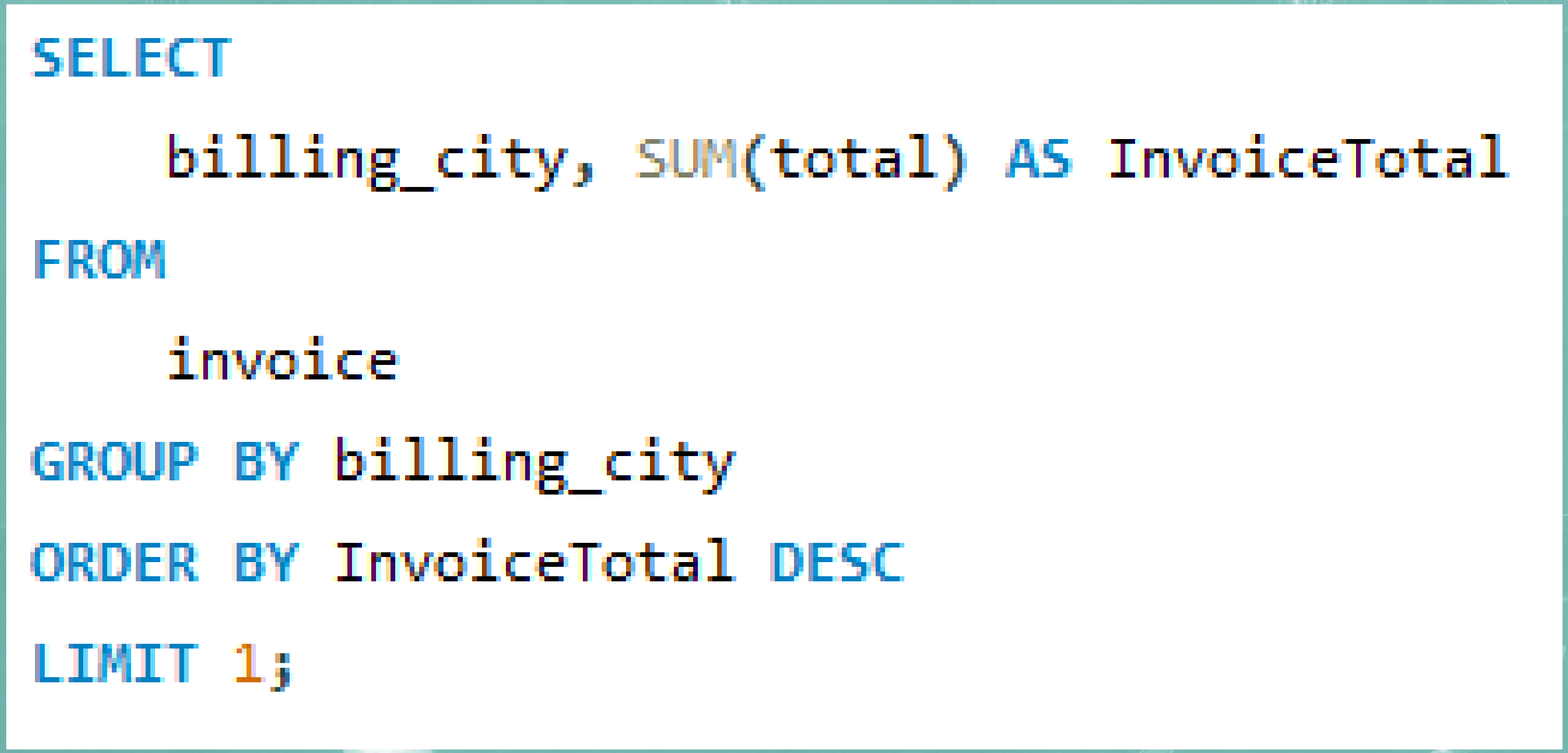
```
SELECT
    COUNT(*) AS c, billing_country
FROM
    invoice
GROUP BY billing_country
ORDER BY c DESC;
```

WHAT ARE TOP 3 VALUES OF TOTAL INVOICE?



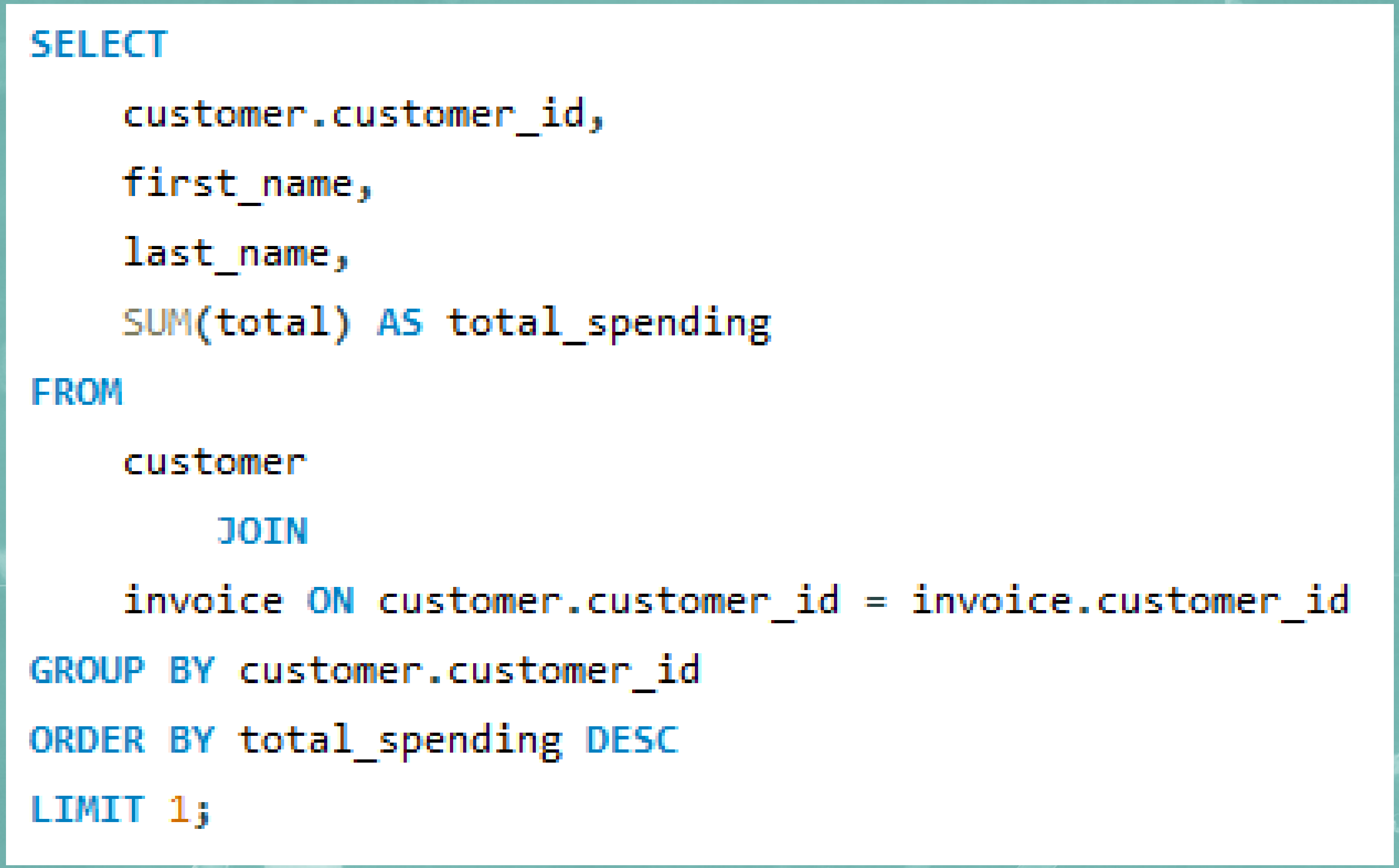
```
SELECT
    total
FROM
    invoice
ORDER BY total DESC;
```

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.



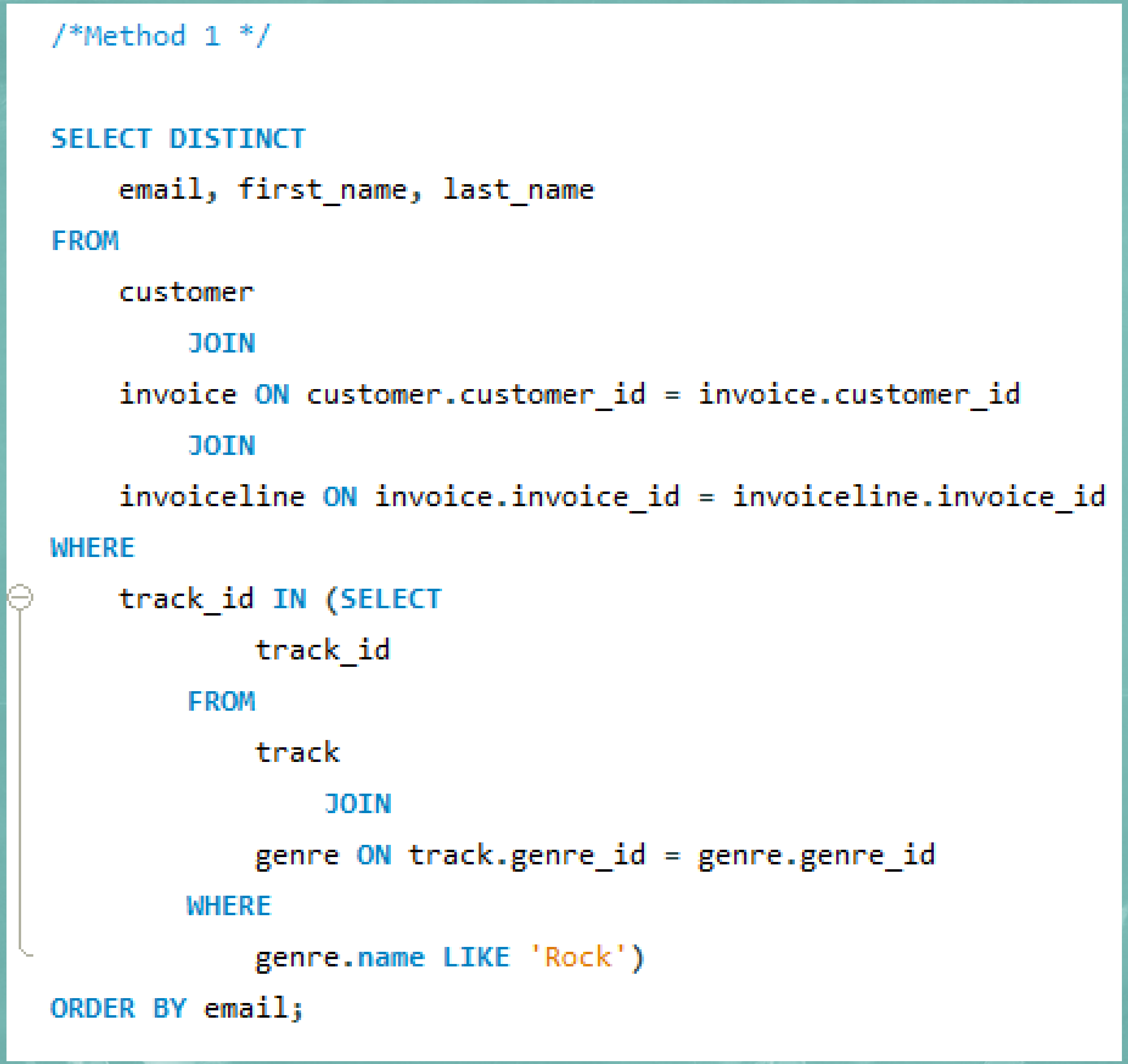
```
SELECT
    billing_city, SUM(total) AS InvoiceTotal
FROM
    invoice
GROUP BY billing_city
ORDER BY InvoiceTotal DESC
LIMIT 1;
```

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.

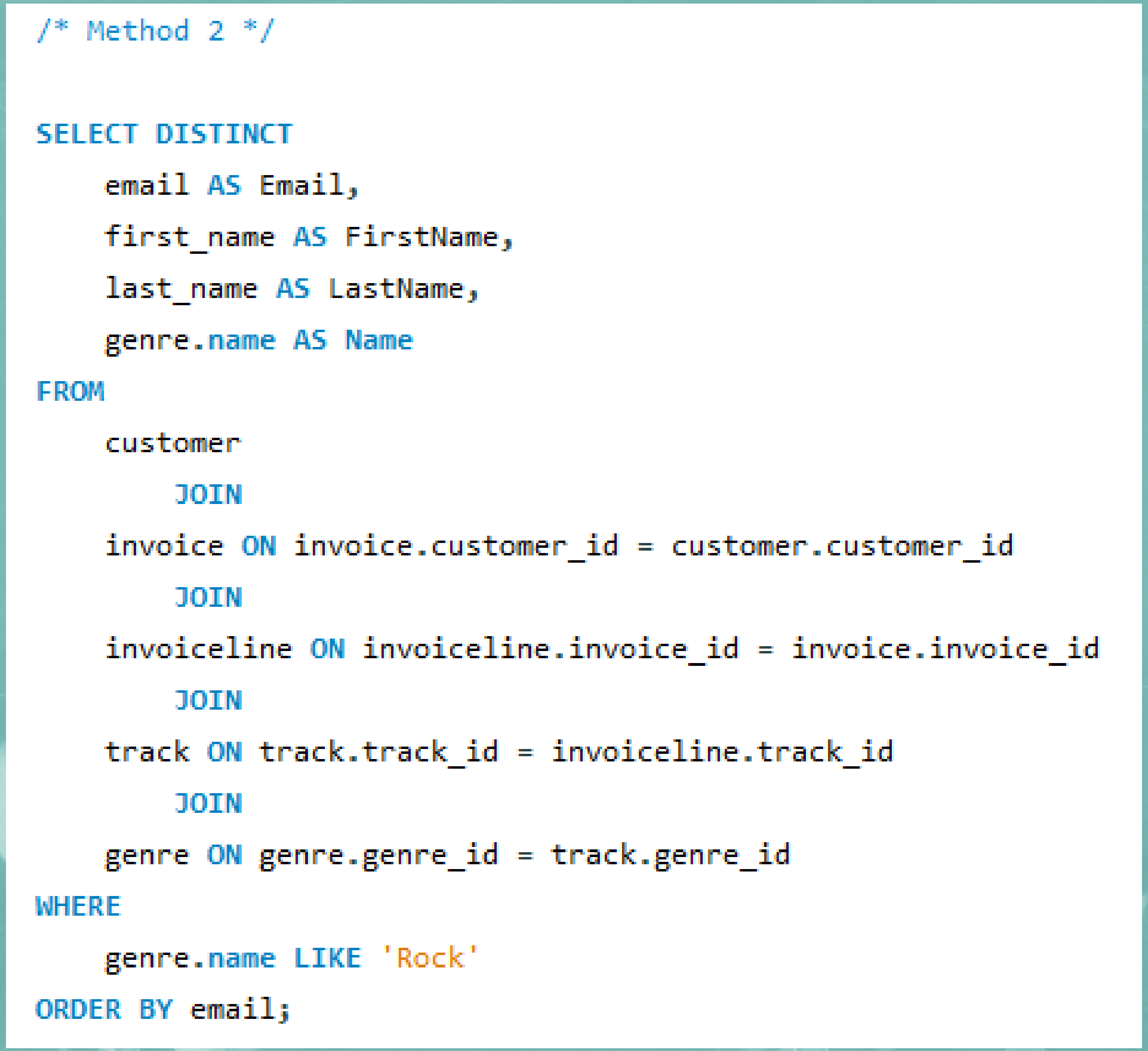


```
SELECT
    customer.customer_id,
    first_name,
    last_name,
    SUM(total) AS total_spending
FROM
    customer
        JOIN
    invoice ON customer.customer_id = invoice.customer_id
GROUP BY customer.customer_id
ORDER BY total_spending DESC
LIMIT 1;
```

Write query to return the email, first name, last name, & Genre of all Rock Music listeners.

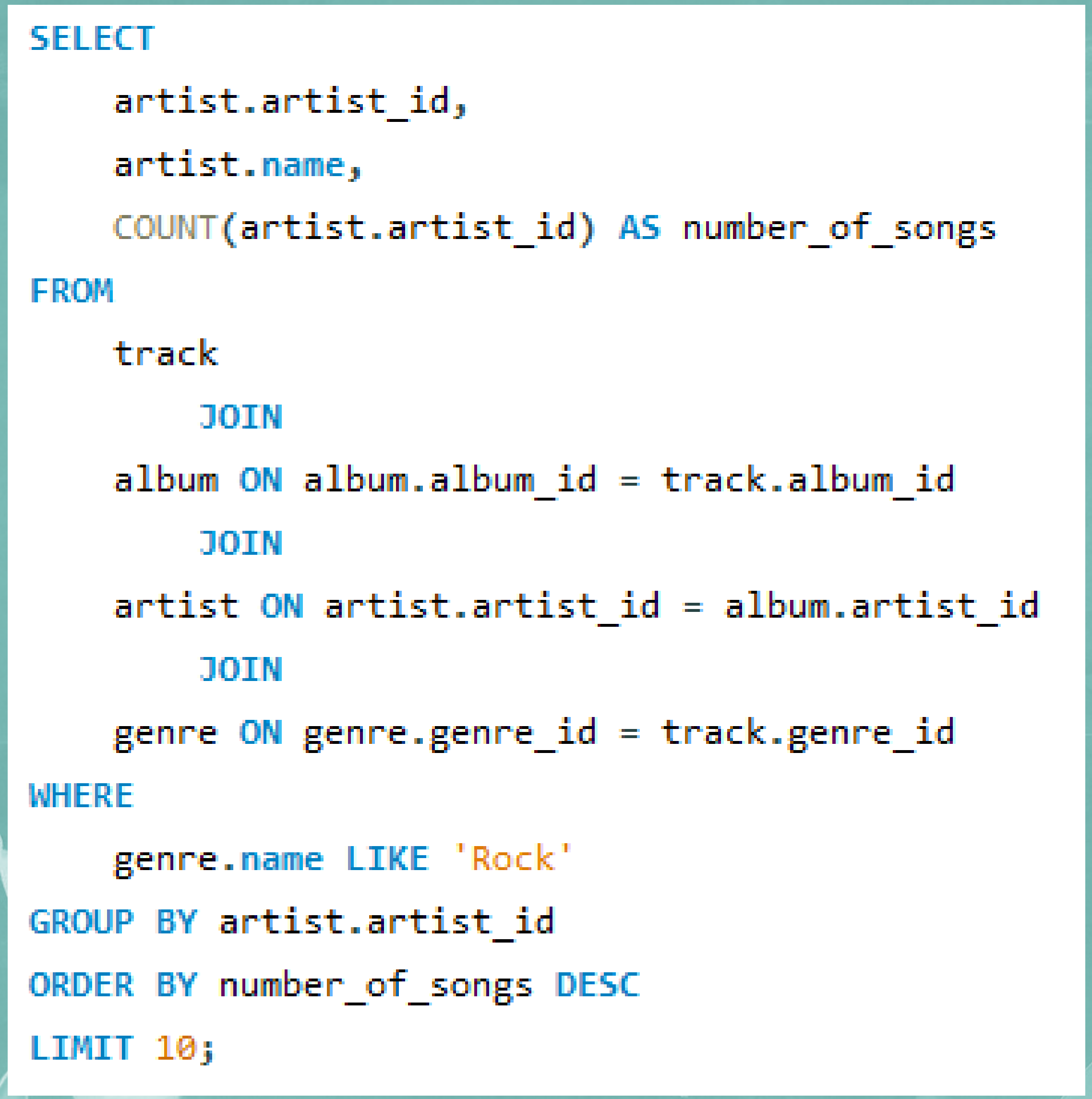


```
/*Method 1 */

SELECT DISTINCT
    email, first_name, last_name
FROM
    customer
        JOIN
    invoice ON customer.customer_id = invoice.customer_id
        JOIN
    invoiceline ON invoice.invoice_id = invoiceline.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;
```



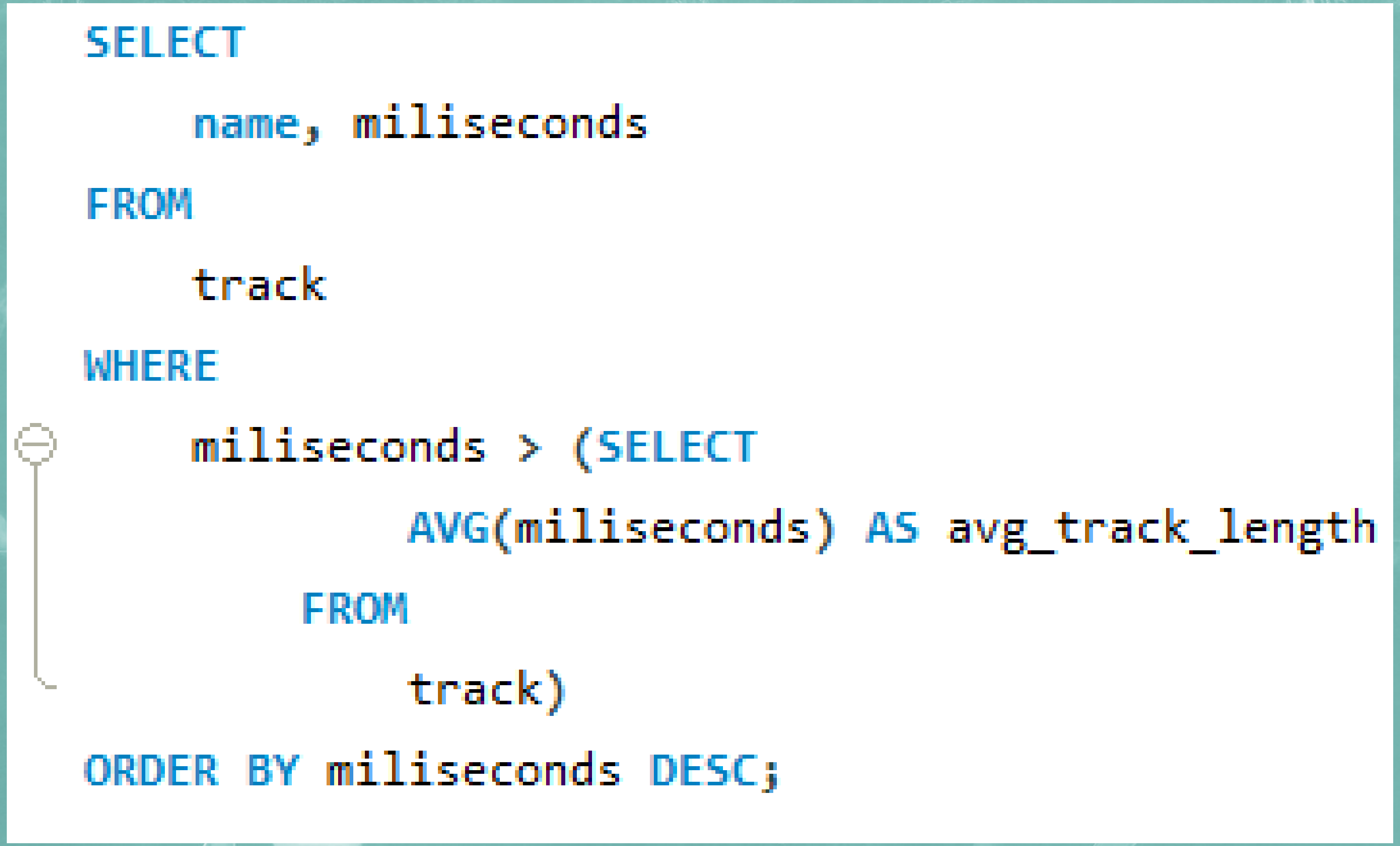
```
/* Method 2 */

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
    invoiceline ON invoiceline.invoice_id = invoice.invoice_id
        JOIN
    track ON track.track_id = invoiceline.track_id
        JOIN
    genre ON genre.genre_id = track.genre_id
WHERE
    genre.name LIKE 'Rock'
ORDER BY email;
```

Write a query that returns the Artist name and total track count of the top 10 rock bands .



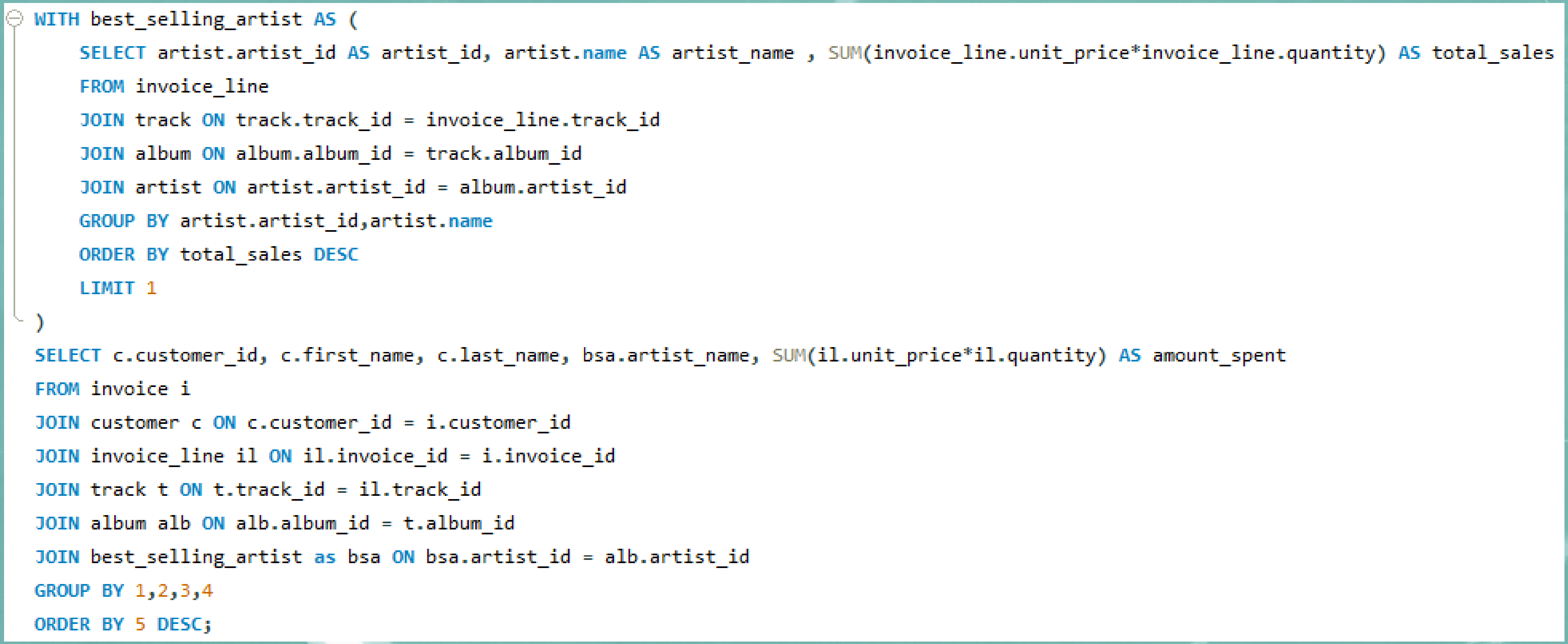
```
SELECT
    artist.artist_id,
    artist.name,
    COUNT(artist.artist_id) AS number_of_songs
FROM
    track
        JOIN
    album ON album.album_id = track.album_id
        JOIN
    artist ON artist.artist_id = album.artist_id
        JOIN
    genre ON genre.genre_id = track.genre_id
WHERE
    genre.name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

Return all the track names that have a song length longer than the average song length. Return the Name and Milliseconds for each track.



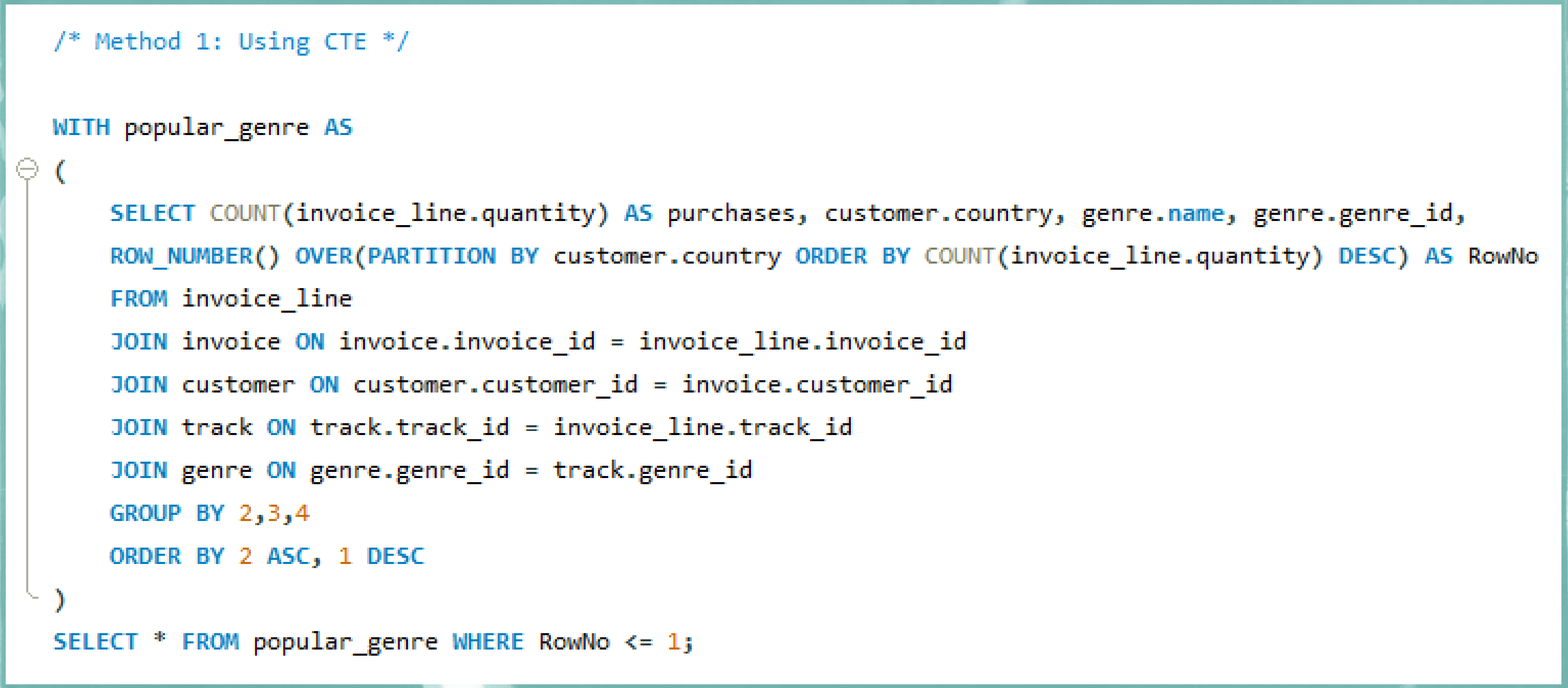
```
SELECT
    name, miliseconds
FROM
    track
WHERE
    miliseconds > (SELECT
            AVG(miliseconds) AS avg_track_length
        FROM
            track)
ORDER BY miliseconds DESC;
```

Find how much amount spent by each customer on artists? Write a query to return customer name, artist name and total spent .

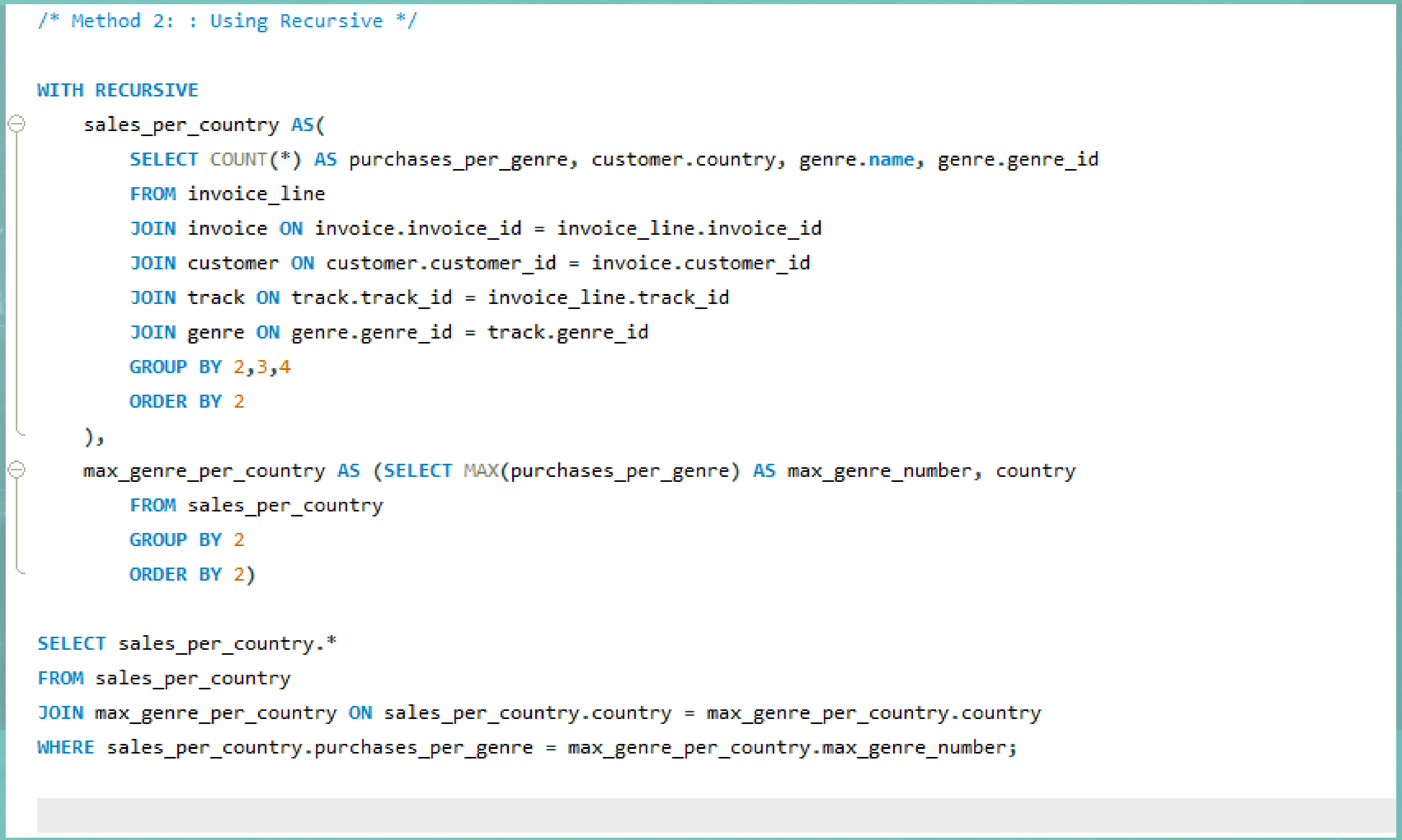


```sql
WITH best_selling_artist AS (
    SELECT artist.artist_id AS artist_id, artist.name AS artist_name , SUM(invoice_line.unit_price*invoice_line.quantity) AS total_sales
    FROM invoice_line
    JOIN track ON track.track_id = invoice_line.track_id
    JOIN album ON album.album_id = track.album_id
    JOIN artist ON artist.artist_id = album.artist_id
    GROUP BY artist.artist_id,artist.name
    ORDER BY total_sales 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 as bsa ON bsa.artist_id = alb.artist_id
GROUP BY 1,2,3,4
ORDER BY 5 DESC;
```

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;
```



```
/* 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;
```

CONCLUSION

- Gained hands-on experience in writing SQL queries on a real-world music store database.
- Used JOINS, GROUP BY, HAVING, ORDER BY, and subqueries for data retrieval and filtering.
- Implemented CTEs and recursive queries to solve complex analytical problems.
- Analyzed revenue patterns across cities and countries for business insights.
- Strengthened my understanding of relational database design and query optimization.

THANK YOU