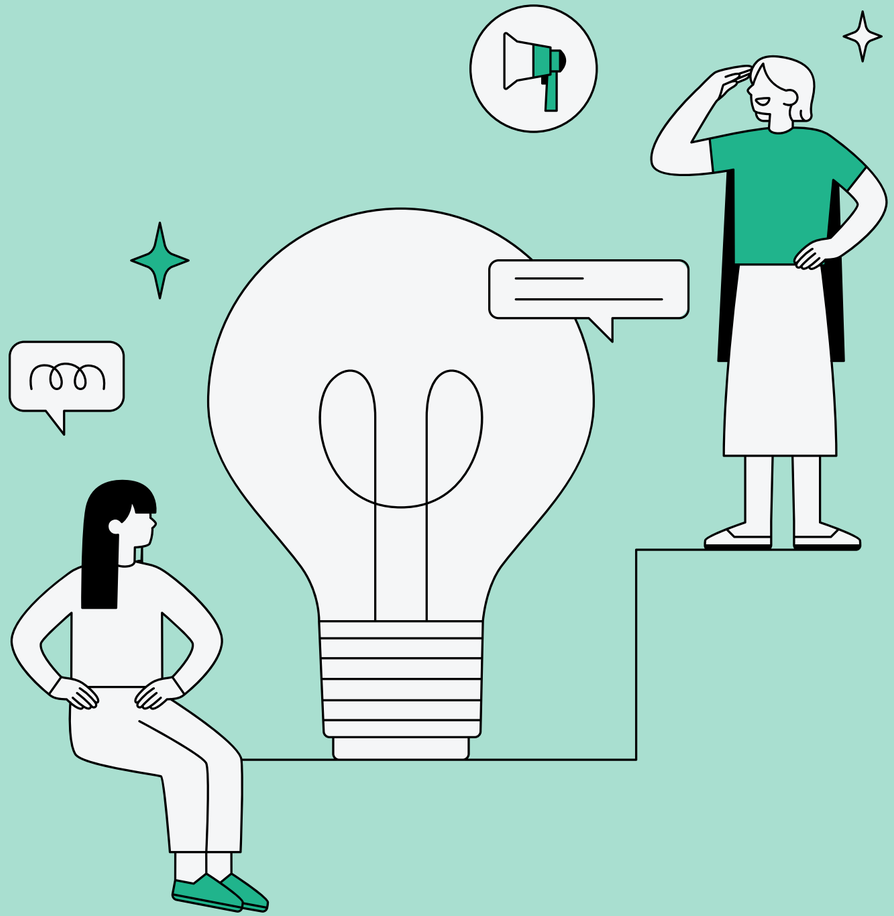
Music store Analysis

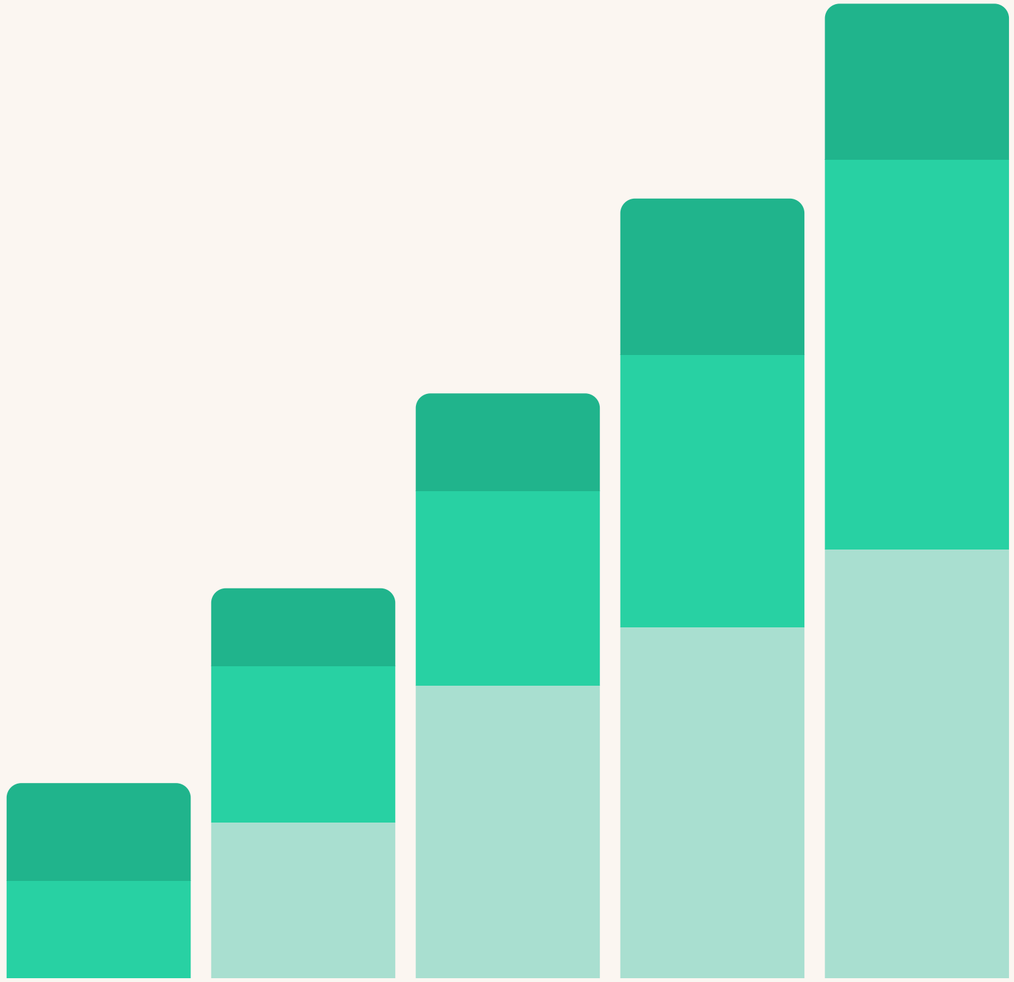
Objective

The objective of this project is to leverage SQL to analyze a comprehensive music dataset from a streaming service. By querying and managing the data, the goal is to extract valuable insights into music consumption patterns, identify popular genres, artists, and tracks, and understand user preferences. The project aims to visualize key trends and provide data-driven recommendations to enhance music services and user experiences.



Methodology used in the analysis

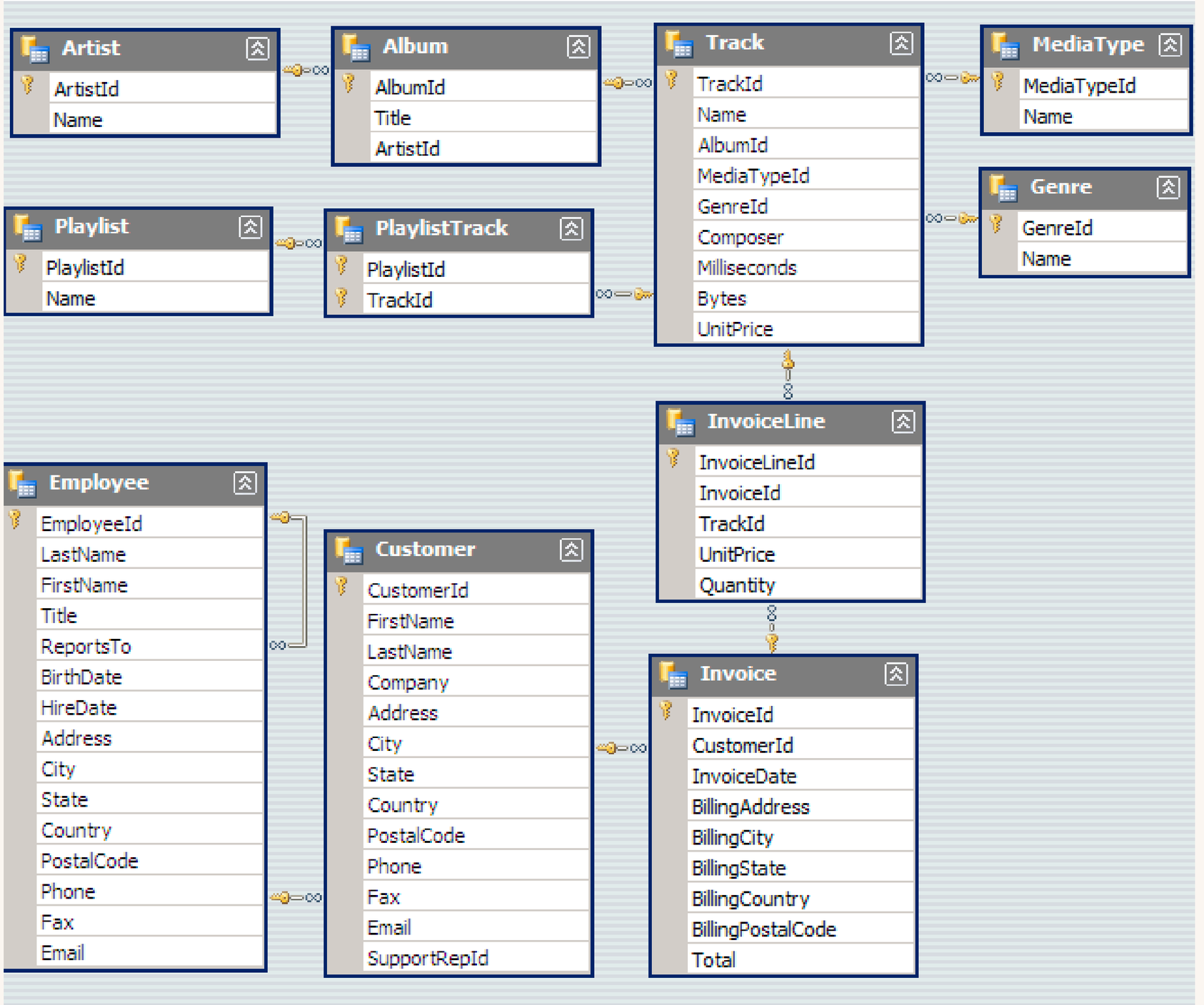
The methodology for this music data analysis project involves several key steps. Initially, data was imported into a PostgreSQL database and cleaned to ensure accuracy and consistency. SQL queries were then utilized to explore the dataset, extracting relevant metrics such as play counts, user demographics, and track metadata. Aggregate functions, joins, and subqueries were employed to identify patterns and trends. Visualization tools like Tableau or Excel were used to present the findings, highlighting key insights and supporting the development of data-driven recommendations.



Overview of current trends

The current trends in the music industry, as revealed by our analysis, indicate a growing preference for streaming services, with significant increases in daily active users and play counts. Pop and hip-hop genres dominate the charts, while emerging genres like indie and electronic are gaining popularity. Users show a strong inclination towards curated playlists and algorithm-generated recommendations. Additionally, there is a notable rise in the consumption of international music, reflecting a more globalized music taste among listeners. These trends highlight the evolving landscape of music consumption and the importance of personalization and diversity in music streaming services.

Schema of the project



Who is the senior most employee based on job title?

```sql
SELECT * FROM
    EMPLOYEE
ORDER BY
    LEVELS DESC
LIMIT
    1;
```

	employee_id [PK] character varying (50)	last_name character	first_name character	title character varying (50)	reports_to character varying (30)	levels character
1	9	Madan	Mohan ...	Senior General Manager	[null]	L7

Which countries have the most Invoices?

```sql
SELECT
    COUNT(INVOICE_ID) AS INVOICE_COUNT,
    BILLING_COUNTRY
FROM
    INVOICE
GROUP BY
    BILLING_COUNTRY
ORDER BY
    INVOICE_COUNT DESC;
```

	invoice_count bigint	billing_country character varying (3
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

What are top 3 values of total invoice?

```
SELECT
    TOTAL
FROM
    INVOICE
ORDER BY
    TOTAL DESC
LIMIT
    3;
```

	total double precision
1	23.759999999999998
2	19.8
3	19.8

Which city has the best customers? We would like to throw a promotional Music Festival in the city we made the most money.

```sql
SELECT
    BILLING_CITY,
    SUM(TOTAL) AS INVOICE_TOTALS
FROM
    INVOICE
GROUP BY
    BILLING_CITY
ORDER BY
    INVOICE_TOTALS DESC
LIMIT
    1;
```

	billing_city character varying (30)	invoice_totals double precision
1	Prague	273.2400000000007

Who is the best customer? The customer who has spent the most money will be declared the best customer.

```sql
SELECT
    C.CUSTOMER_ID,
    C.FIRST_NAME,
    C.LAST_NAME,
    SUM(I.TOTAL) AS TOTAL_SPENT_MONEY
FROM
    INVOICE I
    JOIN CUSTOMER C ON I.CUSTOMER_ID = C.CUSTOMER_ID
GROUP BY
    C.CUSTOMER_ID
ORDER BY
    TOTAL_SPENT_MONEY DESC
LIMIT
    1;
```

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

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.

```sql
SELECT DISTINCT
    C.EMAIL,
    C.FIRST_NAME,
    C.LAST_NAME
FROM
    CUSTOMER C
    JOIN INVOICE I 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
WHERE
    T.TRACK_ID IN (
        SELECT
            T.TRACK_ID
        FROM
            GENRE G
            JOIN TRACK T ON T.GENRE_ID = G.GENRE_ID
        WHERE
            G.NAME = 'Rock'
    ) ORDER BY
    C.EMAIL;
```

	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
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

Let's invite the artists who have written the most rock music in our dataset.

```sql
SELECT DISTINCT
    ARTIST.ARTIST_ID,
    ARTIST.NAME,
    COUNT(TRACK.TRACK_ID) AS TOTAL_TRACK
FROM
    ARTIST
    JOIN ALBUM ON ARTIST.ARTIST_ID = ALBUM.ARTIST_ID
    JOIN TRACK ON TRACK.ALBUM_ID = ALBUM.ALBUM_ID
WHERE
    TRACK.TRACK_ID IN (
        SELECT
            TRACK.TRACK_ID
        FROM
            TRACK
            JOIN GENRE ON TRACK.GENRE_ID = GENRE.GENRE_ID
        WHERE
            GENRE.NAME = 'Rock'
    )
GROUP BY
    ARTIST.ARTIST_ID
ORDER BY
    TOTAL_TRACK DESC
LIMIT
    10;
```

	artist_id [PK] character varying (50)	name character varying (120)	total_track 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

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.

```sql
SELECT
    NAME,
    MILLISECONDS
FROM
    TRACK
WHERE
    MILLISECONDS > (
        SELECT
            AVG(MILLISECONDS)
        FROM
            TRACK
    )
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	2926593
11	The Long Patrol	2925008
12	The Magnificent Warriors	2924716
13	The Living Legend, Pt. 1	2924507

Find how much amount spent by each customer on artists?

```sql
SELECT
    CUSTOMER.CUSTOMER_ID,
    CUSTOMER.FIRST_NAME,
    CUSTOMER.LAST_NAME,
    ARTIST.NAME,
    SUM(INVOICE_LINE.UNIT_PRICE * INVOICE_LINE.QUANTITY) AS TOTAL_SPENT
FROM
    CUSTOMER
    JOIN INVOICE ON CUSTOMER.CUSTOMER_ID = INVOICE.CUSTOMER_ID
    JOIN INVOICE_LINE ON INVOICE.INVOICE_ID = INVOICE_LINE.INVOICE_ID
    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
    1,2,3,4
ORDER BY
    1;
```

```sql
--using CTE
WITH customer_spending AS (
    SELECT
        customer.customer_id,
        customer.first_name,
        customer.last_name,
        artist.name AS artist_name,
        SUM(invoice_line.unit_price * invoice_line.quantity) AS total_spent
    FROM customer
    JOIN invoice ON customer.customer_id = invoice.customer_id
    JOIN invoice_line ON invoice.invoice_id = invoice_line.invoice_id
    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
        customer.customer_id,
        customer.first_name,
        customer.last_name,
        artist.name
)

SELECT
    customer_id,first_name,last_name,artist_name,
    total_spent AS total_spent_per_customer
FROM customer_spending
ORDER BY
    customer_id ;
```

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.

```sql
WITH
    POPULARGENRE_FOREACH_COUNTRY AS (
        SELECT
            CUSTOMER.COUNTRY,
            GENRE.NAME,
            GENRE.GENRE_ID,
            COUNT(INVOICE_LINE.QUANTITY) AS PURCHESES,
            ROW_NUMBER() OVER (
                PARTITION BY
                    CUSTOMER.COUNTRY
                ORDER BY
                    COUNT(INVOICE_LINE.QUANTITY) DESC
            ) AS ROWNO
        FROM
            CUSTOMER
            JOIN INVOICE ON CUSTOMER.CUSTOMER_ID = INVOICE.CUSTOMER_ID
            JOIN INVOICE_LINE ON INVOICE.INVOICE_ID = INVOICE_LINE.INVOICE_ID
            JOIN TRACK ON TRACK.TRACK_ID = INVOICE_LINE.TRACK_ID
            JOIN GENRE ON GENRE.GENRE_ID = TRACK.GENRE_ID
        GROUP BY
            1,
            2,
            3
        ORDER BY
            4)
SELECT
    *
FROM
    POPULARGENRE_FOREACH_COUNTRY
WHERE
    ROWNO <= 1;
```

Write a query that determines the customer that has spent the most on music for each country.

```sql
WITH
    CUSTOMER_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
    CUSTOMER_WITH_COUNTRY
WHERE
    ROWNO <= 1
```

Thank you very much!