

Analyze Data for An Internet Music Outlet With SQL

In this project I did data analysis for an online music store. Some of the key ways I could make a **significant impact** on the business was by answering these questions:

1. **Targeted Marketing Strategies:** Utilize the insights from the analysis to tailor marketing efforts.
2. **Resource Allocation Optimization:** Allocate resources efficiently by understanding customer behavior.
3. **Revenue Maximization:** Leverage the analysis to boost revenue.
4. **Customer Retention and Acquisition:** Enhance customer relationships.
5. **Strategic Decision-Making:** Provide actionable insights to management.

➡ All the queries, their solutions and explanations are given below.

Query 1:

Finding the senior most employee form the employee table.

1. Who is the senior-most employee based on title?
2. SELECT
3. employee_id,
4. first_name,
5. last_name,
6. levels
7. FROM "music_store_data.employee"
8. ORDER BY levels DESC
9. LIMIT 1;

The screenshot shows a SQL query editor interface. The top bar includes tabs for 'Untitled 2' and 'employee', and various buttons like 'RUN', 'SAVE', 'SHARE', and 'MORE'. A status message 'Query completed.' with a checkmark is visible. The code area contains a query to find the senior-most employee based on title, ordered by levels in descending order, and limited to one row. The results section shows a single row with columns: Row, employee_id, first_name, last_name, and levels. The result is a single row: Row 1, employee_id 9, first_name Mohan, last_name Madan, and levels L7.

```
1 --Who is the senior most employee based on title?
2 select
3 employee_id,
4 first_name,
5 last_name,
6 levels
7 from `music_store_data.employee`
8 order by levels desc
9 limit 1
```

JOB INFORMATION	RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	employee_id	first_name	last_name	levels		
1	9	Mohan	Madan	L7		

Query 2:

Finding top countries by the number of invoices by grouping with “billing country”.

1. --Which countries have the most invoices?
2. select
3. billing_country,
4. count(invoice_id) as invoices_count
5. from 'music_store_data_invoice'
6. group by billing_country
7. order by desc;

Made By Pranisha Sharma

The screenshot shows a data analysis interface with a query editor and a results table. The query editor contains the following SQL code:

```
1 --Which countries have the most invoices?
2 select
3 billing_country,
4 count(invoice_id) as invoices_count
5 from 'music_store_data_invoice'
6 group by billing_country
7 order by 2 desc;
```

The results table has columns: Row, billing_country, and invoices_count. The data is as follows:

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

At the bottom, there are navigation links for results per page (50), page 1-24 of 24, and arrows for navigating through the pages.

Query 3:

Finding the city with top sales. Here we take the sum of “total” column and group the data by the “billing city”.

1. --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. Write both the city name and sum of all invoice totals.
2. select
3. billing_city,
4. sum(total) as invoice_total
5. from 'music_store_data_invoice'
6. group by billing_city
7. order by invoice_total desc
8. limit 1;

The screenshot shows a data analysis interface with a query editor and a results table.

Query Editor:

```
1 --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. Write both the city name and sum of all invoice totals.
2 select
3 billing_city,
4 sum(total) as invoice_total
5 from 'music_store_data.invoice'
6 group by billing_city
7 order by invoice_total desc
```

Results Table:

Row	billing_city	invoice_total
1	Prague	273.2399999999...
2	Mountain View	169.29
3	London	166.3200000000...
4	Berlin	158.4
5	Paris	151.47
6	São Paulo	129.6900000000...

Page Footer:

Processing location: US Press Alt+F1 for Accessibility Options.

Results per page: 50 ▾ 1 - 50 of 53 < < > >|

Query 4:

Finding out the best customer for our company. Since customer data is present in the “Customers” table and their purchasing data is in the “Invoice” table, we perform a join operation to obtain the required results.

```
1. --Who is the best customer?
2. --Here we need to find the customer who has spent the most money.
3. select
4. customer.first_name,
5. count(invoice.customer_id) as purchase_count,
6. sum(invoice.total) as invoice_total
7. from 'music_store_data.customer' customer
8. join 'music_store_data.invoice' invoice
9. on customer.customer_id = invoice.customer_id
10. group by customer.first_name, invoice.customer_id
11. order by invoice_total desc;
```

The screenshot shows a database query editor interface with the following details:

- Query Editor Header:** *Untitled, employee, RUN, SAVE, DOWNLOAD, SHARE, SCHEDULE, MORE, Query completed.
- Query Text:** The SQL code for Query 4 is displayed in the editor.
- Processing Location:** US
- Query Results:** The results are presented in a table with the following columns: Row, first_name, purchases_count, and invoice_total.
- Data:** The results show the top customers based on total spending:

Row	first_name	purchases_count	invoice_total
1	František	18	144.5400000000...
2	Helena	12	128.7
3	Hugh	13	114.8399999999...
4	Manoj	13	111.8699999999...
5	Luis	13	108.8999999999...
6	Fernanda	15	106.9199999999...
7	João	13	102.9600000000...
8	Francois	0	00.00

- Page Navigation:** Results per page: 50, 1 – 50 of 59, navigation arrows.

Query 5:

See what the data for rock music listeners looks like. Since email, first name and last name are in the “Customer” table and genre is in the “Genre” table, we will join them. But these table can't be joined directly as they share a relation. Thus, we join these three tables: Customer, Invoice and Genre tables.

1. Return the email, first name, last name and genre of all rock music listeners. Order alphabetically by email.
2. Here email, first name, last name are in customer table and genre is in genre table.
3. Thus we need to join customer, invoice, invoice_line, track and genre tables to get the answer.
4. select distinct customer.email, customer.first_name, customer.last_name
5. from "music_store_data.customer" customer
6. join "music_store_data.invoice" invoice on customer.customer_id = invoice.customer_id
7. join "music_store_data.invoice_line" invoice_line on invoice.invoice_id = invoice_line.invoice_id
8. where track_id in(
 select track_id
 from "music_store_data.track" track
 join "music_store_data.genre" genre on track.genre_id = genre.genre_id
 where genre.name like 'rock')
9. order by customer.email;

The screenshot shows a data processing interface with a query editor and a preview results section. The query editor contains the SQL code for Query 5, which joins the Customer, Invoice, Invoice Line, Track, and Genre tables to find rock music listeners. The preview results show a table with columns: Row, email, first_name, and last_name. The data includes rows for Aaron Mitchell, Alexandre Rocha, Astrid Gruber, Bjørn Hansen, Camille Bernard, Daan Peeters, and Diego Gutiérrez.

```
1 --Return the email, first name, last name and genre of all rock music listeners. Order alphabetically by email.  
2 --Here email, first name and last name are in customer table and genre is in genre table.  
3 --Thus we need to join customer, invoice, invoice_line, track and genre tables to get the answer.  
4 select distinct customer.email, customer.first_name, customer.last_name  
5 from `music_store_data.customer` customer  
6 join `music_store_data.invoice` invoice on customer.customer_id = invoice.customer_id  
7 join `music_store_data.invoice_line` invoice_line on invoice.invoice_id = invoice_line.invoice_id  
8 where track_id in(  
9     select track_id  
10    from `music_store_data.track` track  
11    join `music_store_data.genre` genre on track.genre_id = genre.genre_id  
12    where genre.name like 'Rock')  
13 order by customer.email
```

Processing location: US

Query results

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	email	first_name	last_name				
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.autierrez@yahoo.ar	Diego	Gutiérrez				

Query 6:

Finding the Rock band popularity based on track count. By looking at the schema we realise that we need to join 4 tables in order to obtain the desired result. Finally group the data by artist id and name to obtain the top artists.

```
1. -- Who are the artists who have written the most rock music? Fetch the
   artist name and total track count of top 10 rock bands.
2. select
3. artist.artist_id, artist.name, count(artist.artist_id) as songs_count
4. from 'music_store_data.track' track
5. join 'music_store_data.album' album
6. on track.album_id = album.album_id
7. join 'music_store_data.artist' artist
8. on artist.artist_id = album.artist_id
9. join 'music_store_data.genre' genre
10.on genre.genre_id = track.genre_id
11.where genre.genre_title = 'Rock'
12.group by artist.artist_id, artist.name
13.order by songs_count desc
14.limit 10;
```

Screenshot of a database query editor showing the results of Query 6. The query has been run and completed successfully.

The results table displays the top 10 artists with the most tracks, ordered by song count in descending order. The columns are artist_id, name, and songs_count.

Row	artist_id	name	songs_count
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

Query 7:

Finding track names that have song length longer than average song length. To retrieve all track names with a song length longer than the average, we'll need to calculate the average song length first. Then we can compare each track's length to this average. Analysing the longest tracks provides valuable insights for content curation, user engagement, and revenue optimization.

```
1. -- Return all the track names that have a song length longer than the
   average song length. Order by 'the song' length with 'longest' song listed
   first.
2. select
3. name, milliseconds
4. from 'music_store_data.track'
5. where milliseconds > (
   select avg(milliseconds) as average_length
   from 'music_store_data.track')
6. order by milliseconds desc
7. limit 10;
```

The screenshot shows a data analysis tool interface with the following components:

- Top Bar:** Includes tabs for "Untitled", "RUN", "SAVE", "DOWNLOAD", "SHARE", "SCHEDULE", "MORE", and a status message "Query completed".
- Query Editor:** Displays the SQL query for finding the top 10 longest tracks. The code is as follows:

```
1 -- Return all the track names that have a song length longer than the average song length. Order by 'the song' length with 'longest' song listed first.
2 select
3 name, milliseconds
4 from 'music_store_data.track'
5 where milliseconds > (
6 | select avg(milliseconds) as average_length
7 | from 'music_store_data.track'
8 )
9 order by milliseconds desc
10
```

- Processing Location:** Set to "US".
- Query Results:** A table showing the top 10 longest tracks. The columns are "name" and "milliseconds".

Row	name	milliseconds
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

- Bottom Right:** Buttons for "SAVE RESULTS", "EXPLORE DATA", and navigation icons.
- Bottom Status:** "Results per page: 50" and page navigation controls.

Query 8:

Finding how much each customer has spent on artists. We combine data from multiple tables (e.g., `customer`, `invoice`, `invoice_line`, `track`, `album`, and `artist`) to calculate the total amount spent by each customer on artists. This query provides valuable insights for revenue growth and customer engagement.

```
1. --Find how much amount was spent by each customer on artists? Write a query  
    to return customer name, artist name and total spent  
  
2. WITH best_selling_artist AS (  
3.     SELECT artist.artist_id, artist.name AS artist_name,  
        SUM(invoice_line.unit_price*invoice_line.quantity) AS total_sales  
4.     FROM 'music_store_data.invoice_line' invoice_line  
        JOIN 'music_store_data.track' track  
        ON track.track_id = invoice_line.track_id  
        JOIN music_store_data.album album  
        ON album.album_id = track.album_id  
        JOIN music_store_data.artist artist  
        ON artist.artist_id = album.artist_id  
        GROUP BY 1,2  
        ORDER BY 3 DESC  
5.     LIMIT 1  
6. )  
7.     SELECT customer.customer_id,  
        customer.first_name,customer.last_name,bsa.artist_name,  
        SUM(invoice_line.unit_price*invoice_line.quantity) as amount_spent  
8.   FROM 'music_store_data.invoice' invoice  
9.   JOIN 'music_store_data.customer' customer  
10.  ON customer.customer_id = invoice.customer_id  
11.  JOIN 'music_store_data.invoice_line' invoice_line  
12.  ON invoice_line.invoice_id = invoice.invoice_id  
13.  JOIN 'music_store_data.track' track  
14.  ON track.track_id = invoice_line.track_id  
15.  JOIN 'music_store_data.album' album  
16.  ON album.album_id = track.album_id  
17.  JOIN best_selling_artist bsa  
18.  ON bsa.artist_id=album.artist_id  
19. GROUP BY 1,2,3,4  
20. ORDER BY 5 DESC;
```

Untitled

RUN **SAVE** **DOWNLOAD** **SHARE** **SCHEDULE** **MORE**

Query completed.

```

1 --Find how much amount was spent by each customer on artists? Write a query to return customer name, artist name and total spent
2
3 WITH best_selling_artist AS (
4   SELECT artist.artist_id AS artist_id, artist.name AS artist_name, SUM(invoice_line.unit_price*invoice_line.quantity) AS total_sales
5   FROM `music_store_data.invoice_line` invoice_line
6   JOIN `music_store_data.track` track
7   ON track.track_id = invoice_line.track_id
8   JOIN `music_store_data.album` album
9   ON album.album_id = track.album_id
10  JOIN `music_store_data.artist` artist
11  ON artist.artist_id = album.artist_id
12  GROUP BY 1,2
13  ORDER BY 3 DESC
14  LIMIT 1
15 )
16 SELECT customer.customer_id, customer.first_name, customer.last_name, bsa.artist_name, SUM(invoice_line.unit_price*invoice_line.quantity) AS amount_spent
17 FROM `music_store_data.invoice` invoice
18 JOIN `music_store_data.customer` customer
19 ON customer.customer_id = invoice.customer_id
20 JOIN `music_store_data.invoice_line` invoice_line
21 ON invoice_line.invoice_id = invoice.invoice_id
22 JOIN `music_store_data.track` track
23 ON track.track_id = invoice_line.track_id
24 JOIN `music_store_data.album` album
25 ON album.album_id = track.album_id
26 JOIN best_selling_artist bsa
27 ON bsa.artist_id = album.artist_id
28 GROUP BY 1,2,3,4
29 ORDER BY 5 DESC

```

Processing location: US

Press Alt+F1 for Accessibility Options.

Query results

SAVE RESULTS

EXPLORE DATA

Results image:

Untitled

RUN **SAVE** **DOWNLOAD** **SHARE** **SCHEDULE** **MORE**

Query completed.

```

1 --Find how much amount was spent by each customer on artists? Write a query to return customer name, artist name and total spent
2
3 WITH best_selling_artist AS (
4   SELECT artist.artist_id AS artist_id, artist.name AS artist_name, SUM(invoice_line.unit_price*invoice_line.quantity) AS total_sales
5   FROM `music_store_data.invoice_line` invoice_line
6   JOIN `music_store_data.track` track
7   ON track.track_id = invoice_line.track_id
8   JOIN `music_store_data.album` album
9   ON album.album_id = track.album_id
10  JOIN `music_store_data.artist` artist
11  ON artist.artist_id = album.artist_id
12  GROUP BY 1,2
13  ORDER BY 3 DESC
14  LIMIT 1
15 )
16 SELECT customer.customer_id, customer.first_name, customer.last_name, bsa.artist_name, SUM(invoice_line.unit_price*invoice_line.quantity) AS amount_spent
17 FROM `music_store_data.invoice` invoice
18 JOIN `music_store_data.customer` customer
19 ON customer.customer_id = invoice.customer_id
20 JOIN `music_store_data.invoice_line` invoice_line
21 ON invoice_line.invoice_id = invoice.invoice_id
22 JOIN `music_store_data.track` track
23 ON track.track_id = invoice_line.track_id
24 JOIN `music_store_data.album` album
25 ON album.album_id = track.album_id
26 JOIN best_selling_artist bsa
27 ON bsa.artist_id = album.artist_id
28 GROUP BY 1,2,3,4
29 ORDER BY 5 DESC

```

Processing location: US

Press Alt+F1 for Accessibility Options.

Query results

SAVE RESULTS

EXPLORE DATA

JOB INFORMATION		RESULTS		CHART	PREVIEW	JSON	EXECUTION DETAILS		EXECUTION GRAPH	
Row	customer_id	first_name	last_name				artist_name	amount_spent		
1	46	Hugh	O'Reilly				Queen	27.71999999999999...		
2	38	Niklas	Schröder				Queen	18.81		
3	3	François	Tremblay				Queen	17.82		
4	34	João	Fernandes				Queen	16.83000000000000...		
5	41	Marc	Dubois				Queen	11.88		
6	53	Phil	Hughes				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	František	Wichterlová				Queen	3.96		

Results per page: 50 ▾ 1 - 43 of 43 | < < > >|

Query 9:

Finding the most popular music genre for each country. With the help of a CTE, we calculate the total number of purchases for each genre in each country. The ROW_NUMBER() function assigns a rank to each genre within a country based on the purchase count. PARTITION BY CountryId groups the data by country. Within each country, the ranking is calculated. This analysis provides actionable insights for marketing, content curation, and revenue growth within the music industry.

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.

2. with popular_genre as (
3. select count(invoice_line.quantity) as purchases, customer.country,
genre.name, genre.genre_id,
4. row_number() over(partition by customer.country order by
count(invoice_line.quantity) desc) as rowno
5. from 'music_store_data.invoice_line' invoice_line
6. join 'music_store_data.invoice' invoice
7. on invoice.invoice_id = invoice_line.invoice_id
8. join 'music_store_data.customer' customer
9. on customer.customer_id = invoice.customer_id
10. join 'music_store_data.track' track
11. on track.track_id = invoice_line.track_id
12. join 'music_store_data.genre' genre
13. on genre.genre_id = track.genre_id
14. group by 2, 3, 4
15. order by 2 asc, 1 desc
16.)
17. select * from popular_genre where rowno <= 1;

*Untitled

RUN **SAVE** **DOWNLOAD** **SHARE** **SCHEDULE** **MORE** **Query completed.**

```

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
2 of purchases. Write a query that returns each country along with the top Genre. For countries where the maximum number of purchases is shared
3 return all Genres.
4
5 with popular_genre as
6 (
7   | select count(invoice_line.quantity) as purchases, customer.country, genre.name, genre.genre_id,
8   | row_number() over(partition by customer.country order by count(invoice_line.quantity) desc) as rowno
9   | from `music_store_data.invoice_line` invoice_line
10  | join `music_store_data.invoice` invoice
11  | on invoice.invoice_id = invoice_line.invoice_id
12  | join `music_store_data.customer` customer
13  | on customer.customer_id = invoice.customer_id
14  | join `music_store_data.track` track
15  | on track.track_id = invoice_line.track_id
16  | join `music_store_data.genre` genre
17  | on genre.genre_id = track.genre_id
18  | group by 2,3,4
19  | order by 2 asc, 1 desc
)
19 select * from popular_genre where rowno <= 1

```

Processing location: US

Press Alt+F1 for Accessibility Options.

Query results

SAVE RESULTS

EXPLORE DATA

JOB INFORMATION		RESULTS		CHART	PREVIEW	JSON	EXECUTION DETAILS		EXECUTION GRAPH	
Row	purchases	country				name	genre_id	rowno		
1	17	Argentina				Alternative & Punk		4		1

Results per page: 50 1 – 24 of 24

Results image:

JOB INFORMATION		RESULTS		CHART	PREVIEW	JSON	EXECUTION DETAILS		EXECUTION GRAPH	
Row	purchases	country				name	genre_id	rowno		
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

Results per page: 50 1 – 24 of 24

Query 10:

Finding the top spending customer for each country based on their spending on music. With the help of a CTE, we calculate the total spending for each customer in each country. Then we assign a row number to each customer within their country, ordered by total spending (descending). Finally, we select rows from the CTE where the row number is 1 (i.e., the top spending customer for each country).

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 top customer and how much they spent. For countries where the top amount spent is shared, provide all customers who spent this amount.
2. with customer_with_country as (
 3. select customer.customer_id, first_name, last_name, billing_country,
 sum(total) as total_spending,
 4. row_number() over(partition by billing_country order by sum(total) desc) as
 rowno
 5. from 'music_store_data.invoice' invoice
 6. join 'music_store_data.customer' customer on customer.customer_id =
 invoice.customer_id
 7. group by 1,2,3,4
 8. order by 4 asc,5 desc)
 9. select * from customer_with_country where rowno <= 1;

Untitled RUN SAVE DOWNLOAD SHARE SCHEDULE MORE Query completed.

```

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 top customer and how much they spent. For countries where the top amount spent is shared, provide all customers who spent this amount.
2
3 with customer_with_country as (
4   select customer.customer_id , first_name , last_name , billing_country , sum(total) as total_spending,
5   row_number() over(partition by billing_country order by sum(total) desc) as rowno
6   from `music_store_data.invoice` invoice
7   join `music_store_data.customer` customer on customer.customer_id = invoice.customer_id
8   group by 1,2,3,4
9   order by 4 asc,5 desc)
10 select * from customer_with_country where rowno <= 1

```

Processing location: US × Press Alt+F1 for Accessibility Options.

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS		EXECUTION GRAPH	
Row	customer_id	first_name			last_name	billing_country	total_spending	rowno	
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	60.38999999999...	1	
5	1	Luis			Gonçalves	Brazil	108.8999999999...	1	
6	3	François			Tremblay	Canada	99.99	1	
7	57	Luis			Rojas	Chile	97.0200000000...	1	

Results per page: 50 ▾ 1 – 24 of 24 |< < > >|

Results image:

*Untitled

Untitled

RUN SAVE DOWNLOAD SHARE SCHEDULE MORE Query completed.

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 top customer and how much they spent. For countries where the top amount spent is shared, provide all customers who spent this amount.
2
3 with recursive
4 | customer_with_country as (
5 | | select customer.customer_id , first_name , last_name , billing_country , sum(total) as total_spending
6 | | from `music_store_data.invoice` invoice

Processing location: US Press Alt+F1 for Accessibility Options.

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION	RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	billing_country	total_spending	first_name	last_name	customer_id	
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	108.89999999999999	Luís	Gonçalves	1	
6	Canada	99.99	François	Tremblay	3	
7	Chile	97.02000000000001	Luis	Rojas	57	
8	Czech Republic	144.54000000000001	František	Wichterlová	5	
9	Denmark	37.61999999999999	Kara	Nielsen	9	
10	Finland	79.2	Terhi	Hämäläinen	44	
11	France	99.99	Wvatt	Girard	42	

Results per page: 50 1 – 24 of 24