**Problem Statement**

**By**

**ISHA VERMA**

**Spotify Data Analysis Project**

1. Who is the senior most employee based on job title?

**Query :-**

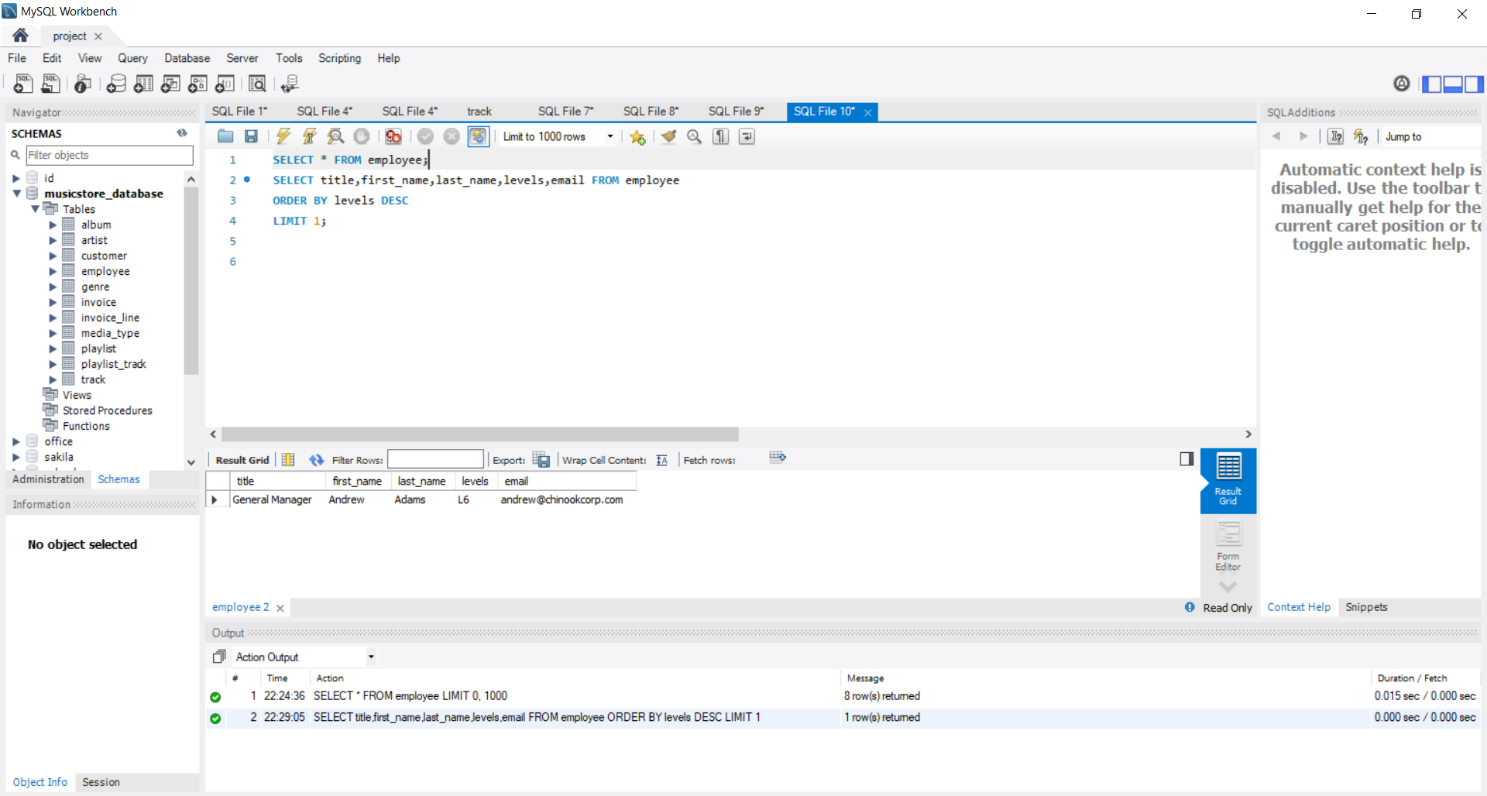
SELECT title ,first\_name,last\_name,levels,email FROM employee

ORDER BY levels DESC

LIMIT 1;

**Output :-**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Title** | **First\_name** | **Last\_name** | **Levels** | **Email** |
| General Manager | Andrew | Adams | L6 | andrew@chinookcorp.com |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

**Steps to Write and Execute the Query:**

1. **Start with SELECT Statement**  
   We use SELECT to choose the columns we want to display.  
   In this case, we selected:  
   title, first\_name, last\_name, levels, and email.
2. **Specify the Table**  
   Use FROM employee to tell SQL that we are getting data from the employee table.
3. **Sort the Data by Levels**  
   Use ORDER BY levels DESC to sort the employees from highest to lowest level.
4. **Limit the Result to 1 Row**  
   Use LIMIT 1 to show only the first row from the sorted list.  
   This gives us the employee with the highest level.
5. **Final Output**  
   The result shows the top-level employee. i.e

**General Manager Andrew Adams L6** [**andrew@chinookcorp.com**](mailto:andrew@chinookcorp.com)

1. Which countries have the most Invoices?

**Query :-**

SELECT billing\_country,count(\*) as count FROM invoice

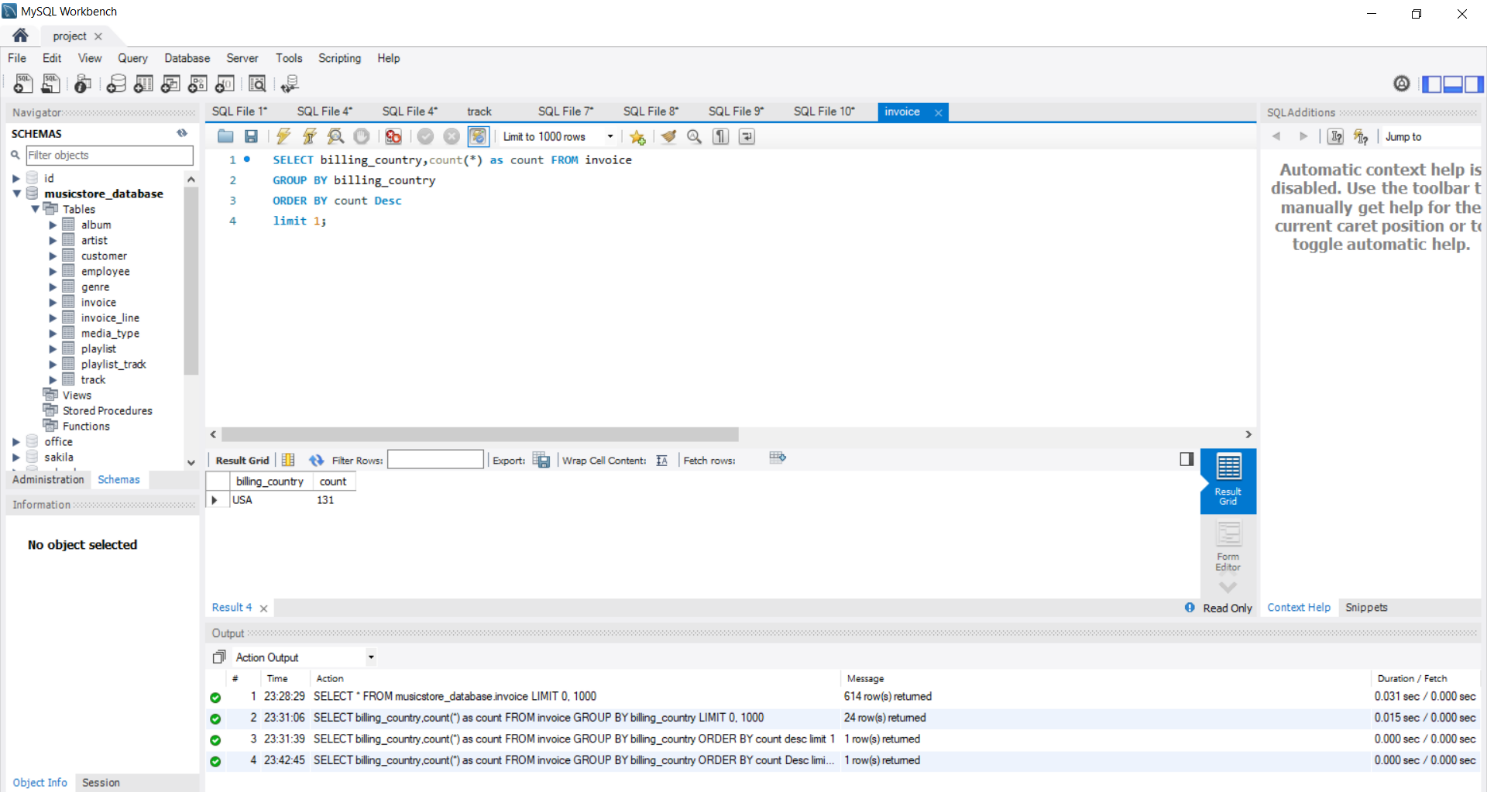
GROUP BY billing\_country

ORDER BY count Desc

limit 1;

**output :-**

|  |  |
| --- | --- |
| **Billing\_country** | **Count** |
| USA | 131 |

****

**Steps to Write and Execute the Query:**

1. **Start by Selecting the Needed Columns**

We write : SELECT billing\_country, COUNT(\*) AS count

This means : show the country name and how many invoices came from each country.

1. **Group the Results by Country**

We use : GROUP BY billing\_country

This groups all invoices based on each country, so we can count them separately.

1. **Sort the Results from Highest to Lowest**

We write : ORDER BY count DESC

This arranges the countries in order from the one with the most invoices to the least.

1. **Limit the Output to Only the Top One**

We use : LIMIT 1

This shows only the country with the **highest** number of invoices.

**5.** **Final Output**

USA 131

This means **USA has the most invoices — a total of 131**.

1. What are top 3 values of total invoice?

**Query :-**

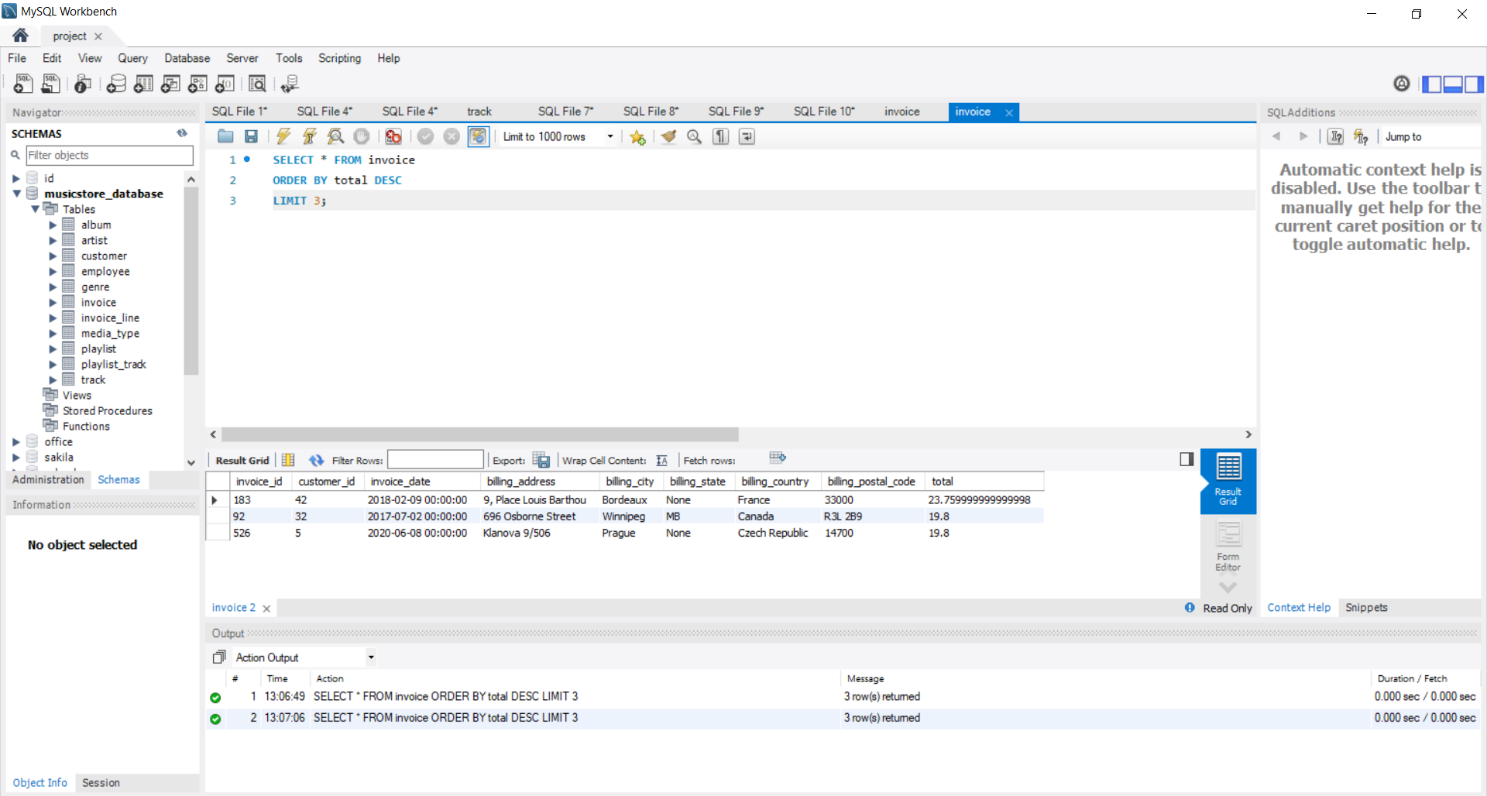
SELECT \* FROM invoice

ORDER BY total DESC

LIMIT 3;

**Output :-**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Invoice\_id** | **Customer\_id** | **Invoice\_date** | **Billing\_address** | **Billing\_city** | **Billing\_state** | **Billing\_country** | **Billing\_postal\_code** | **Tota**l |
| 183 | 42 | 2018-02-09 00:00:00 | 9, Place Louis Barthou | Bordeaux | None | France | 33000 | 23.760 |
| 92 | 32 | 2017-07-02 00:00:00 | 696 Osborne Street | Winnipeg | MB | Canada | R3L 2B9 | 19.8 |
| 526 | 5 | 2020-06-08 00:00:00 | Klanova 9/506 | Prague | None | Czech Republic | 14700 | 19.8 |



**Steps to Write and Execute the Query:**

1. **Select All Columns**  
   We write : SELECT \*
2. **Choose the Table**

We write : FROM invoice

1. **Sort the Data by Total**   
   We write : ORDER BY total DESC

This arranges the invoices starting from the **highest total amount to the lowest**.

1. **Limit the Results to Top 3**  
   We write : LIMIT 3

This shows only the **top 3 invoices** with the highest total amounts.

**5.** **Final Output**

| **Invoice ID** | **Billing Country** | **Total Amount** |
| --- | --- | --- |
| 183 | France | 23.760 |
| 92 | Canada | 19.800 |
| 526 | Czech Republic | 19.800 |

4. Which city has the best customers?

**Query :-**

SELECT billing\_city,sum(total) as Total\_Amount FROM invoice

GROUP BY billing\_city

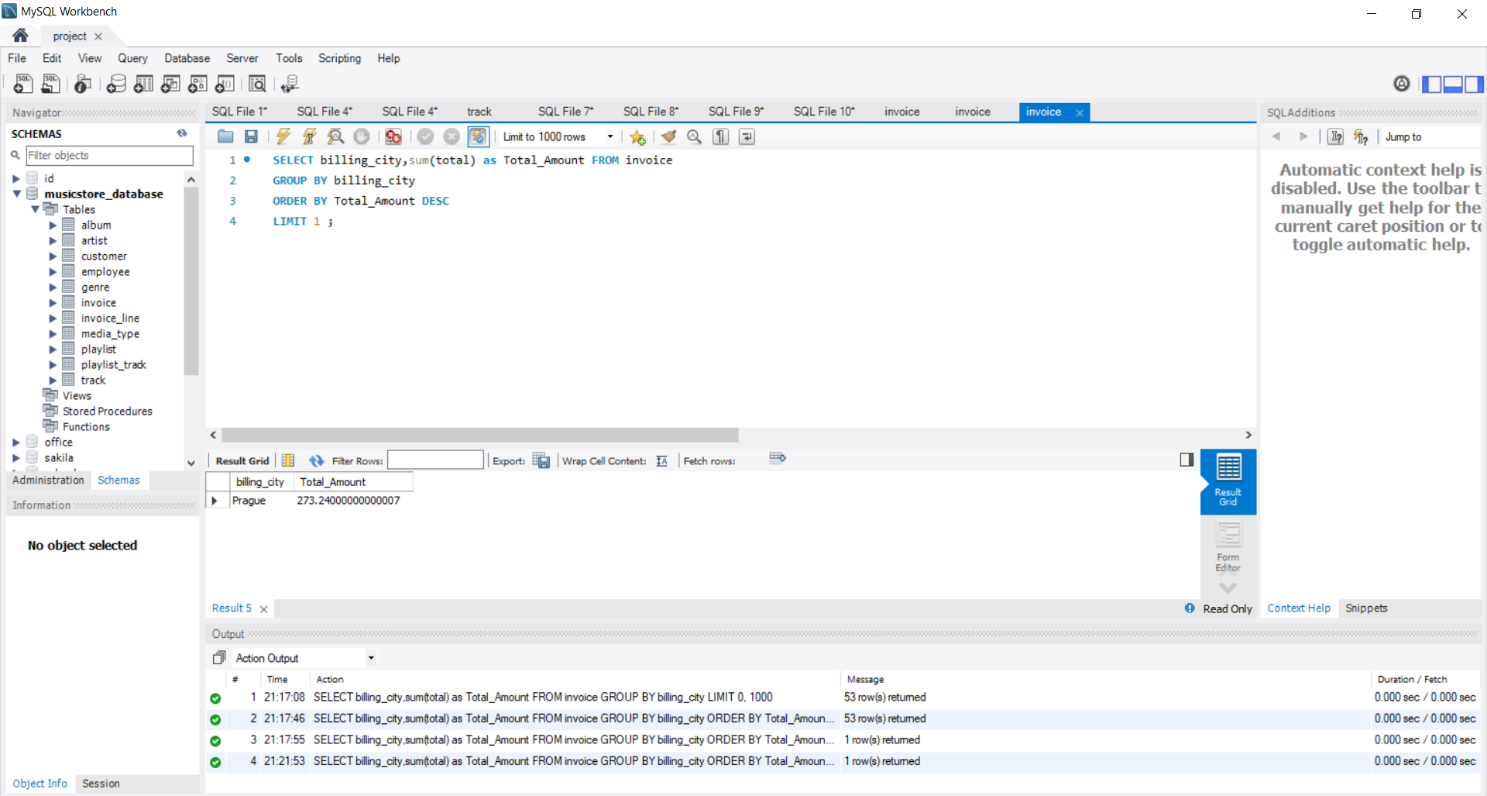
ORDER BY Total\_Amount DESC

LIMIT 1 ;

**Output :-**

|  |  |
| --- | --- |
| **Billing\_City** | **Total\_Amount** |
| Prague | 273.24000000000007 |

|  |
| --- |
|  |

a

**Steps to Write and Execute the Query:**

1. **Start with SELECT**  
    We use : SELECT billing\_city, SUM(total) AS Total\_Amount

This means : show the name of each city and add up all the invoice totals for that city.

1. **Group the Results by City**  
    We write : GROUP BY billing\_city

This groups the data city-wise so that the total amount is calculated separately for each city.

1. **Sort the Totals in Descending Order**  
    We use : ORDER BY Total\_Amount DESC

This arranges the cities from the **highest total amount to the lowest**.

1. **Limit the Result to Just One City**  
    We write : LIMIT 1

This shows **only the top city** — the one with the highest total invoice amount.

**5. Final Output:**

City: Prague Total Amount: 273.24

This means **Prague** has the **highest total sales amount** among all billing cities.

5.Who is the best customer?

**Query :-**

SELECT customer.customer\_id,first\_name,last\_name,

email,sum(total) AS Total\_Amount FROM customer

JOIN invoice

ON customer.customer\_id=invoice.customer\_id

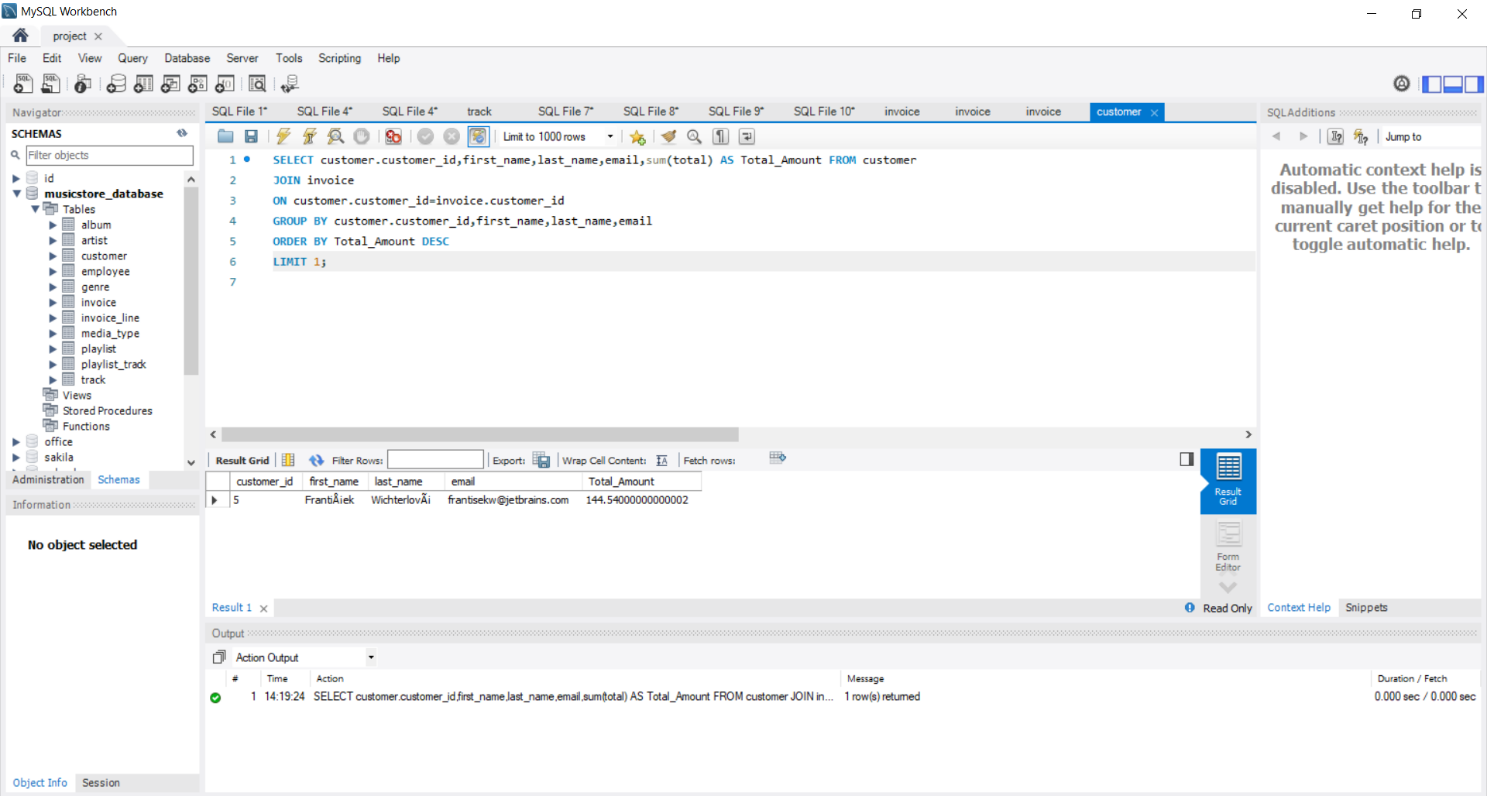
GROUP BY customer.customer\_id,first\_name,last\_name,email

ORDER BY Total\_Amount DESC

LIMIT 1;

**Output :-**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Customer\_id** | **First\_Name** | **Last\_Name** | **Email** | **Total\_Amount** |
| 5 | FrantiÅ¡ek | WichterlovÃ¡ | frantisekw@jetbrains.com | 144.54000000000002 |

****

**Steps to Write and Execute the Query:**

1. **Select the Customer Details and Total Amount**  
    We write : SELECT customer.customer\_id, first\_name, last\_name, email, SUM(total) AS Total\_Amount FROM customer

This selects the customer's ID, name, email, and the total amount they spent.

1. **Join the Customer and Invoice Tables**  
    We use : JOIN invoice ON customer.customer\_id = invoice.customer\_id

This connects each invoice to the correct customer using the customer\_id.

1. **Group the Data by Customer**  
    We write : GROUP BY customer.customer\_id, first\_name, last\_name, email

This groups all the invoices for each customer so that the total amount they spent can be calculated.

1. **Sort the Results by Total Amount**  
    We use : ORDER BY Total\_Amount DESC

This arranges the customers from the **highest spending to the lowest**.

1. **Show Only the Top Customer**  
    We write : LIMIT 1

This displays only the **customer who spent the most money**.

**6. Final Output:**

| **Customer ID** | **First Name** | **Last Name** | **Email** | **Total Amount** |
| --- | --- | --- | --- | --- |
| 5 | František | Wichterlová | [frantisekw@jetbrains.com](mailto:frantisekw@jetbrains.com) | 144.54 |

This means **František Wichterlová** is the **top spending customer**, with a total

purchase amount of **144.54**.

6. Details of customers who listen Rock music.

**Query :-**

SELECT distinct first\_name,last\_name,email FROM customer

JOIN invoice

ON customer.customer\_id=invoice.customer\_id

JOIN invoice\_line

ON invoice.invoice\_id=invoice\_line.invoice\_id

WHERE track\_id IN (

SELECT track\_id FROM track

JOIN genre

ON track.genre\_id=genre.genre\_id

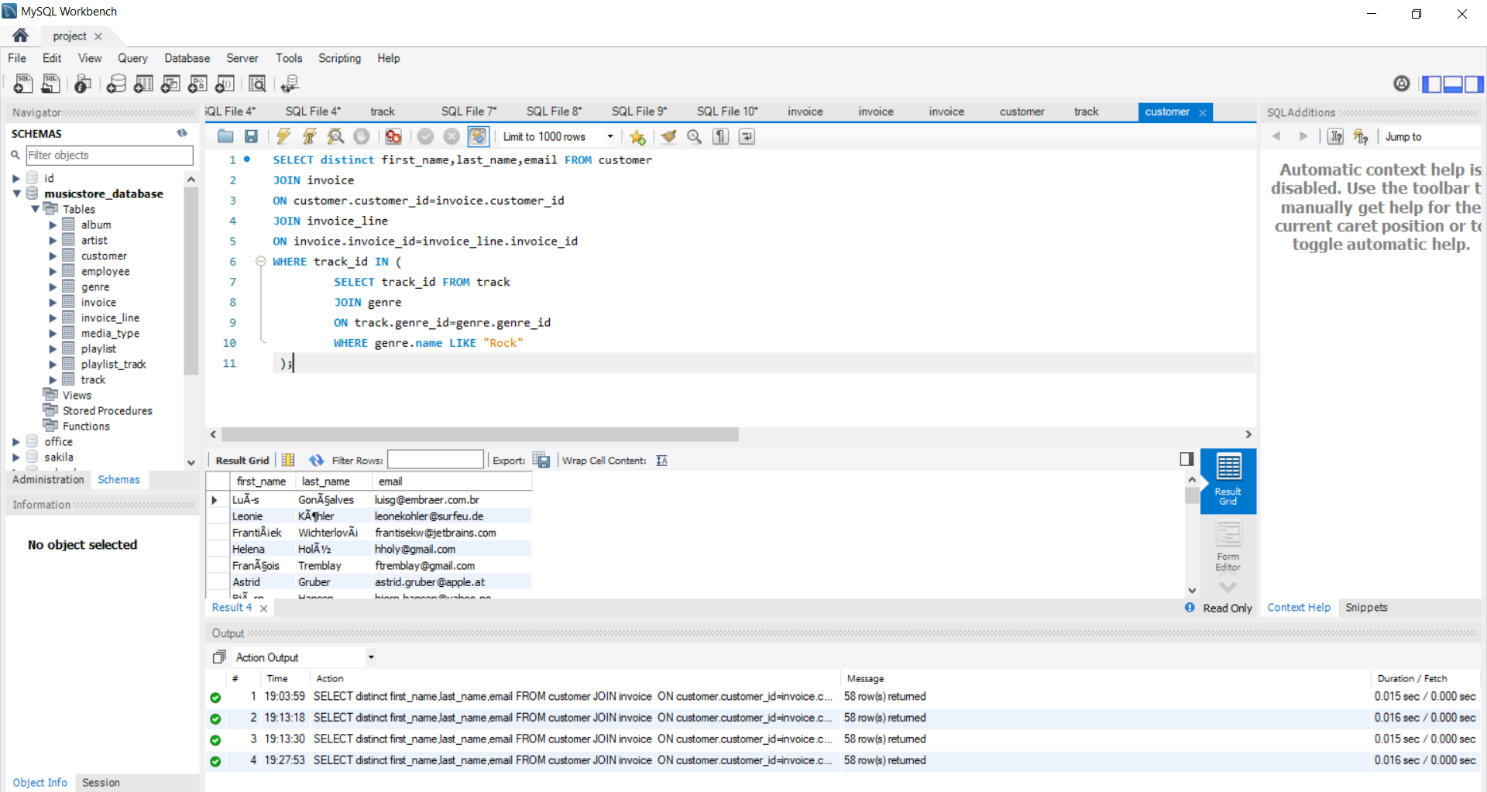
WHERE genre.name LIKE "Rock"

);

**Output :-**

**First\_name Last\_name Email**

|  |  |  |
| --- | --- | --- |
| LuÃ­s | GonÃ§alves | luisg@embraer.com.br |
| Leonie | KÃ¶hler | leonekohler@surfeu.de |
| FrantiÅ¡ek | WichterlovÃ¡ | frantisekw@jetbrains.com |
| Helena | HolÃ½ | hholy@gmail.com |
| FranÃ§ois | Tremblay | ftremblay@gmail.com |
| Astrid | Gruber | astrid.gruber@apple.at |
| BjÃ¸rn | Hansen | bjorn.hansen@yahoo.no |
| Kara | Nielsen | kara.nielsen@jubii.dk |
| Roberto | Almeida | roberto.almeida@riotur.gov.br |
| Daan | Peeters | daan\_peeters@apple.be |
| Fernanda | Ramos | fernadaramos4@uol.com.br |
| Jennifer | Peterson | jenniferp@rogers.ca |
| Mark | Philips | mphilips12@shaw.ca |
| Alexandre | Rocha | alero@uol.com.br |
| Eduardo | Martins | eduardo@woodstock.com.br |
| Dan | Miller | dmiller@comcast.com |
| Tim | Goyer | tgoyer@apple.com |
| Frank | Harris | fharris@google.com |
| Heather | Leacock | hleacock@gmail.com |
| Jack | Smith | jacksmith@microsoft.com |
| Kathy | Chase | kachase@hotmail.com |
| John | Gordon | johngordon22@yahoo.com |
| Edward | Francis | edfrancis@yachoo.ca |
| Robert | Brown | robbrown@shaw.ca |
| Frank | Ralston | fralston@gmail.com |
| Victor | Stevens | vstevens@yahoo.com |
| Patrick | Gray | patrick.gray@aol.com |
| Julia | Barnett | jubarnett@gmail.com |
| Richard | Cunningham | ricunningham@hotmail.com |
| Martha | Silk | marthasilk@gmail.com |
| JoÃ£o | Fernandes | jfernandes@yahoo.pt |
| Ellie | Sullivan | ellie.sullivan@shaw.ca |
| Hannah | Schneider | hannah.schneider@yahoo.de |
| Aaron | Mitchell | aaronmitchell@yahoo.ca |
| Madalena | Sampaio | masampaio@sapo.pt |
| Niklas | SchrÃ¶der | nschroder@surfeu.de |
| Fynn | Zimmermann | fzimmermann@yahoo.de |
| Wyatt | Girard | wyatt.girard@yahoo.fr |
| Isabelle | Mercier | isabelle\_mercier@apple.fr |
| Camille | Bernard | camille.bernard@yahoo.fr |
| Terhi | HÃ¤mÃ¤lÃ¤inen | terhi.hamalainen@apple.fi |
| Ladislav | KovÃ¡cs | ladislav\_kovacs@apple.hu |
| Marc | Dubois | marc.dubois@hotmail.com |
| Dominique | Lefebvre | dominiquelefebvre@gmail.com |
| StanisÅ‚aw | WÃ³jcik | stanisÅ‚aw.wÃ³jcik@wp.pl |
| Hugh | O'Reilly | hughoreilly@apple.ie |
| Emma | Jones | emma\_jones@hotmail.com |
| Lucas | Mancini | lucas.mancini@yahoo.it |
| Joakim | Johansson | joakim.johansson@yahoo.se |
| Johannes | Van der Berg | johavanderberg@yahoo.nl |
| Enrique | MuÃ±oz | enrique\_munoz@yahoo.es |
| Phil | Hughes | phil.hughes@gmail.com |
| Mark | Taylor | mark.taylor@yahoo.au |
| Luis | Rojas | luisrojas@yahoo.cl |
| Manoj | Pareek | manoj.pareek@rediff.com |
| Steve | Murray | steve.murray@yahoo.uk |
| Rishabh | Mishra | rishabh\_mishra@yahoo.in |
| Diego | GutiÃ©rrez | diego.gutierrez@yahoo.ar |



**Steps to Write and Execute the Query:**

1. **Start by Selecting Customer Details**  
   We use : SELECT DISTINCT first\_name, last\_name, email

This selects the **unique names and email addresses** of customers who bought Rock tracks. DISTINCT ensures that duplicate entries are removed.

1. **Join the customer and invoice Tables**  
   We write : FROM customer JOIN invoice ON customer.customer\_id = invoice.customer\_id

This connects each customer to their invoices using customer\_id.

1. **Join the invoice and invoice\_line Tables**  
   We write : JOIN invoice\_line ON invoice.invoice\_id = invoice\_line.invoice\_id

This connects each invoice to the specific tracks purchased using invoice\_id.

1. **Filter Only Rock Genre Tracks Using a Subquery**  
   We use a subquery in the WHERE clause:

WHERE track\_id IN (

SELECT track\_id FROM track

JOIN genre ON track.genre\_id = genre.genre\_id

WHERE genre.name LIKE "Rock" )

* + This subquery finds all the track IDs that belong to the **Rock genre**.
  + It joins the track and genre tables using genre\_id.
  + Then it filters for only those tracks where the genre is **"Rock"**.

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

**Query :-**

SELECT artist.artist\_id,artist.name AS Artist\_Name,genre.name AS genre,count(artist.artist\_id) AS No\_Of\_Songs FROM artist

JOIN album

ON artist.artist\_id = album.artist\_id

JOIN track

ON album.album\_id = track.album\_id

JOIN genre

ON track.genre\_id = genre.genre\_id

WHERE genre.name like "Rock"

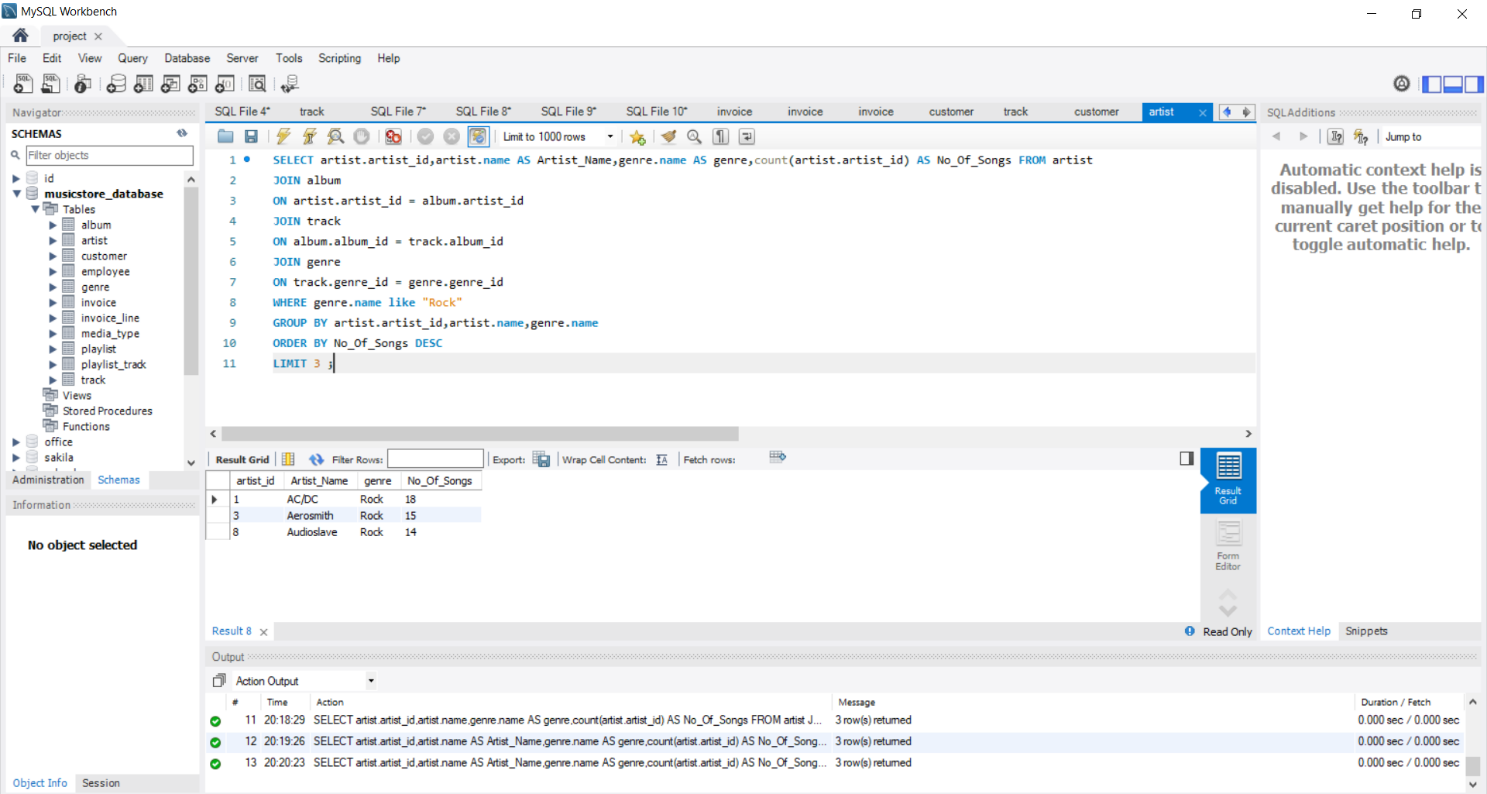
GROUP BY artist.artist\_id,artist.name,genre.name

ORDER BY No\_Of\_Songs DESC

LIMIT 3 ;

**Output :-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Artist\_id** | **Artist\_name** | **Genre** | **No\_Of\_Songs** |
| 1 | AC/DC | Rock | 18 |
| 3 | Aerosmith | Rock | 15 |
| 8 | Audioslave | Rock | 14 |



**Steps to Write and Execute the Query:**

1. **Start by Selecting Artist Details and Song Count**  
   We selected the artist ID, artist name, genre name, and used COUNT to find out how many Rock songs each artist has:

SELECT artist.artist\_id, artist.name AS Artist\_Name, genre.name AS genre, COUNT(artist.artist\_id) AS No\_Of\_Songs

1. **Join Artist Table with Album Table**  
   We used : JOIN album ON artist.artist\_id = album.artist\_id

This connects each artist to their albums.

1. **Join Album Table with Track Table**  
   We used : JOIN track ON album.album\_id = track.album\_id

This connects albums to the tracks they contain.

1. **Join Track Table with Genre Table**  
   We used : JOIN genre ON track.genre\_id = genre.genre\_id

This connects each track to its genre name.

1. **Filter Only Rock Genre**  
   We wrote : WHERE genre.name LIKE "Rock"

This filters the results to only include songs from the **Rock** genre.

1. **Group Results by Artist and Genre**  
   We wrote : GROUP BY artist.artist\_id, artist.name, genre.name

This groups the songs for each artist so we can count how many Rock songs they have.

1. **Sort Results by Number of Songs**  
   We wrote : ORDER BY No\_Of\_Songs DESC

This sorts the artists from the one with the **most Rock songs to the least**.

1. **Limit the Output to Top 3 Artists**  
   We used : LIMIT 3

This shows only the **top 3 artists** who have the **highest number of Rock songs**.

**9. Final Output:**

| **Artist ID** | **Artist Name** | **Genre** | **No. of Songs** |
| --- | --- | --- | --- |
| 1 | AC/DC | Rock | 18 |
| 3 | Aerosmith | Rock | 15 |
| 8 | Audioslave | Rock | 14 |

This shows that **AC/DC** has the highest number of Rock songs in the database, followed by **Aerosmith** and **Audioslave**.

8. We want to find out the most popular music Genre for each country.

**Query :-**

With popular\_genre AS

(SELECT invoice.billing\_country,count(invoice\_line.quantity) AS Quantity,genre.name AS genre,

ROW\_NUMBER() OVER(PARTITION BY invoice.billing\_country ORDER BY count(invoice\_line.quantity) DESC ) AS row\_num

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

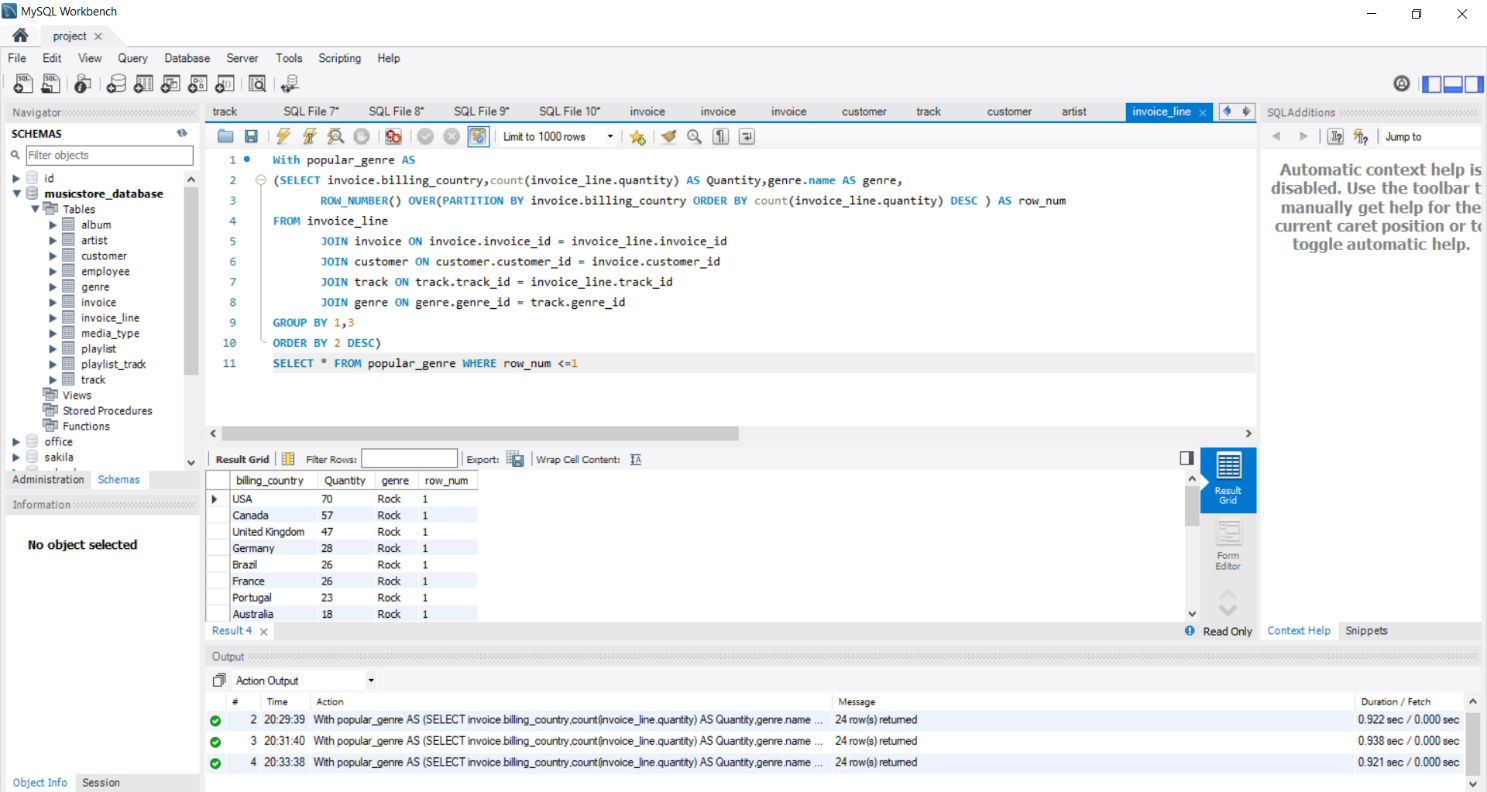
GROUP BY 1,3

ORDER BY 2 DESC)

SELECT \* FROM popular\_genre WHERE row\_num <=1;

**Output :-**

|  |  |  |  |
| --- | --- | --- | --- |
| USA | 70 | Rock | 1 |
| Canada | 57 | Rock | 1 |
| United Kingdom | 47 | Rock | 1 |
| Germany | 28 | Rock | 1 |
| Brazil | 26 | Rock | 1 |
| France | 26 | Rock | 1 |
| Portugal | 23 | Rock | 1 |
| Australia | 18 | Rock | 1 |
| Poland | 14 | Rock | 1 |
| Czech Republic | 14 | Rock | 1 |
| India | 13 | Rock | 1 |
| Chile | 7 | Rock | 1 |
| Austria | 6 | Rock | 1 |
| Netherlands | 6 | Rock | 1 |
| Denmark | 6 | Rock | 1 |
| Finland | 6 | Rock | 1 |
| Belgium | 5 | Rock | 1 |
| Sweden | 5 | Rock | 1 |
| Spain | 4 | Metal | 1 |
| Hungary | 4 | Rock | 1 |
| Italy | 3 | Rock | 1 |
| Norway | 2 | Metal | 1 |
| Ireland | 2 | Rock | 1 |
| Argentina | 1 | Rock | 1 |



**Steps to Write and Execute the Query:**

1. **Start by Creating a Temporary Table (CTE):**  
    We used WITH popular\_genre AS (...) to create a temporary result, so we can work step by step in an organized way. .
2. **Join the Necessary Tables: .**
   * invoice\_line (shows which tracks were sold)
   * invoice (tells us the billing country)
   * customer (links invoice to customer)
   * track (contains track details)
   * genre (tells the genre of each track)

All these tables are connected to find **which genres were sold in which country**.

1. **Group by Country and Genre: .**  
   We grouped the data by: .
   * billing\_country
   * genre.name

This helps us **count how many songs** of each genre were sold in each country.

1. **Count the Quantity of Tracks Sold:**  
    We used : COUNT(invoice\_line.quantity) AS Quantity

This gives the **total number of tracks sold** for each genre in each country.

1. **Use ROW\_NUMBER to Rank Genres in Each Country:**  
    We added : ROW\_NUMBER() OVER(PARTITION BY billing\_country ORDER BY COUNT DESC) AS row\_num

This **ranks the genres** from most sold to least, **separately for each country**.

1. **Filter to Get Only the Top Genre per Country:**  
    In the final step, we wrote : SELECT \* FROM popular\_genre WHERE row\_num <= 1

This keeps **only the most popular genre** (rank 1) in each country.

9. Determine which customer has spent the most on music for each country.

**Query :-**

WITH TOP\_COUNTRY\_CUSTOMER AS

(SELECT customer.customer\_id,first\_name,last\_name,email,

billing\_country AS country,sum(total) AS total\_amount,

ROW\_NUMBER() OVER(PARTITION BY billing\_country ORDER BY sum(total) DESC) AS row\_num

FROM customer

JOIN invoice ON customer.customer\_id = invoice.customer\_id

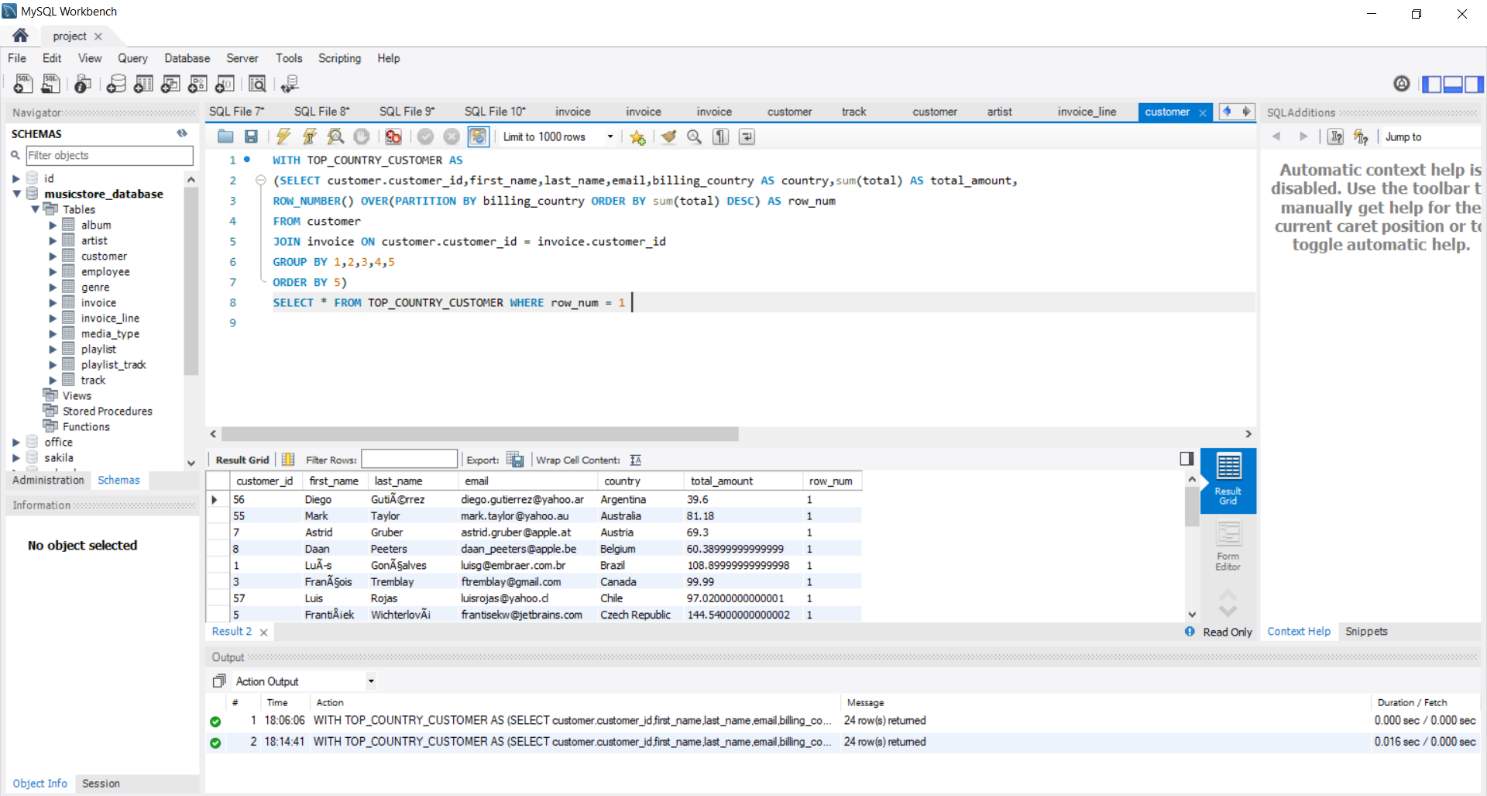
GROUP BY 1,2,3,4,5

ORDER BY 5)

SELECT \* FROM TOP\_COUNTRY\_CUSTOMER WHERE row\_num = 1;

**Output :-**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 56 | Diego | GutiÃ©rrez | diego.gutierrez@yahoo.ar | Argentina | 39.6 | 1 |
| 55 | Mark | Taylor | mark.taylor@yahoo.au | Australia | 81.18 | 1 |
| 7 | Astrid | Gruber | astrid.gruber@apple.at | Austria | 69.3 | 1 |
| 8 | Daan | Peeters | daan\_peeters@apple.be | Belgium | 60.38999999999999 | 1 |
| 1 | LuÃ­s | GonÃ§alves | luisg@embraer.com.br | Brazil | 108.89999999999998 | 1 |
| 3 | FranÃ§ois | Tremblay | ftremblay@gmail.com | Canada | 99.99 | 1 |
| 57 | Luis | Rojas | luisrojas@yahoo.cl | Chile | 97.02000000000001 | 1 |
| 5 | FrantiÅ¡ek | WichterlovÃ¡ | frantisekw@jetbrains.com | Czech Republic | 144.54000000000002 | 1 |
| 9 | Kara | Nielsen | kara.nielsen@jubii.dk | Denmark | 37.61999999999999 | 1 |
| 44 | Terhi | HÃ¤mÃ¤lÃ¤inen | terhi.hamalainen@apple.fi | Finland | 79.2 | 1 |
| 42 | Wyatt | Girard | wyatt.girard@yahoo.fr | France | 99.99 | 1 |
| 37 | Fynn | Zimmermann | fzimmermann@yahoo.de | Germany | 94.05000000000001 | 1 |
| 45 | Ladislav | KovÃ¡cs | ladislav\_kovacs@apple.hu | Hungary | 78.21 | 1 |
| 58 | Manoj | Pareek | manoj.pareek@rediff.com | India | 111.86999999999999 | 1 |
| 46 | Hugh | O'Reilly | hughoreilly@apple.ie | Ireland | 114.83999999999997 | 1 |
| 47 | Lucas | Mancini | lucas.mancini@yahoo.it | Italy | 50.49 | 1 |
| 48 | Johannes | Van der Berg | johavanderberg@yahoo.nl | Netherlands | 65.34 | 1 |
| 4 | BjÃ¸rn | Hansen | bjorn.hansen@yahoo.no | Norway | 72.27000000000001 | 1 |
| 49 | StanisÅ‚aw | WÃ³jcik | stanisÅ‚aw.wÃ³jcik@wp.pl | Poland | 76.22999999999999 | 1 |
| 34 | JoÃ£o | Fernandes | jfernandes@yahoo.pt | Portugal | 102.96000000000001 | 1 |
| 50 | Enrique | MuÃ±oz | enrique\_munoz@yahoo.es | Spain | 98.01 | 1 |
| 51 | Joakim | Johansson | joakim.johansson@yahoo.se | Sweden | 75.24 | 1 |
| 53 | Phil | Hughes | phil.hughes@gmail.com | United Kingdom | 98.01 | 1 |
| 17 | Jack | Smith | jacksmith@microsoft.com | USA | 98.01 | 1 |



**Steps to Write and Execute the Query:**

1. **Start with a temporary table (CTE): .**  
   We used WITH TOP\_COUNTRY\_CUSTOMER AS (...) to store the result temporarily and work step by step. .
2. **Join two tables:**
   * customer (gives customer details like name, email, country)
   * invoice (gives the amount each customer spent)  
     We joined them using customer\_id to connect customer with their invoices. .
3. **Group the data: .**  
   We grouped the result by:
   * Customer ID
   * First name, last name, email
   * Billing country  
     This helped us calculate the total spending of each customer in each country. .
4. **Calculate total amount spent: .**  
   We used SUM(total) to get the total money spent by each customer. .
5. **Rank customers in each country: .**  
   We used ROW\_NUMBER() to give a rank to each customer in their country, based on total spending.  
   The one who spent the most gets **rank 1**. .
6. **Get only the top spender per country: .**  
   In the final step, we filtered the result using : WHERE row\_num = 1

This gave us only the **top 1 customer from each country** who spent the most.

**Assignment Question**

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

**Query :-**

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 1,2

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

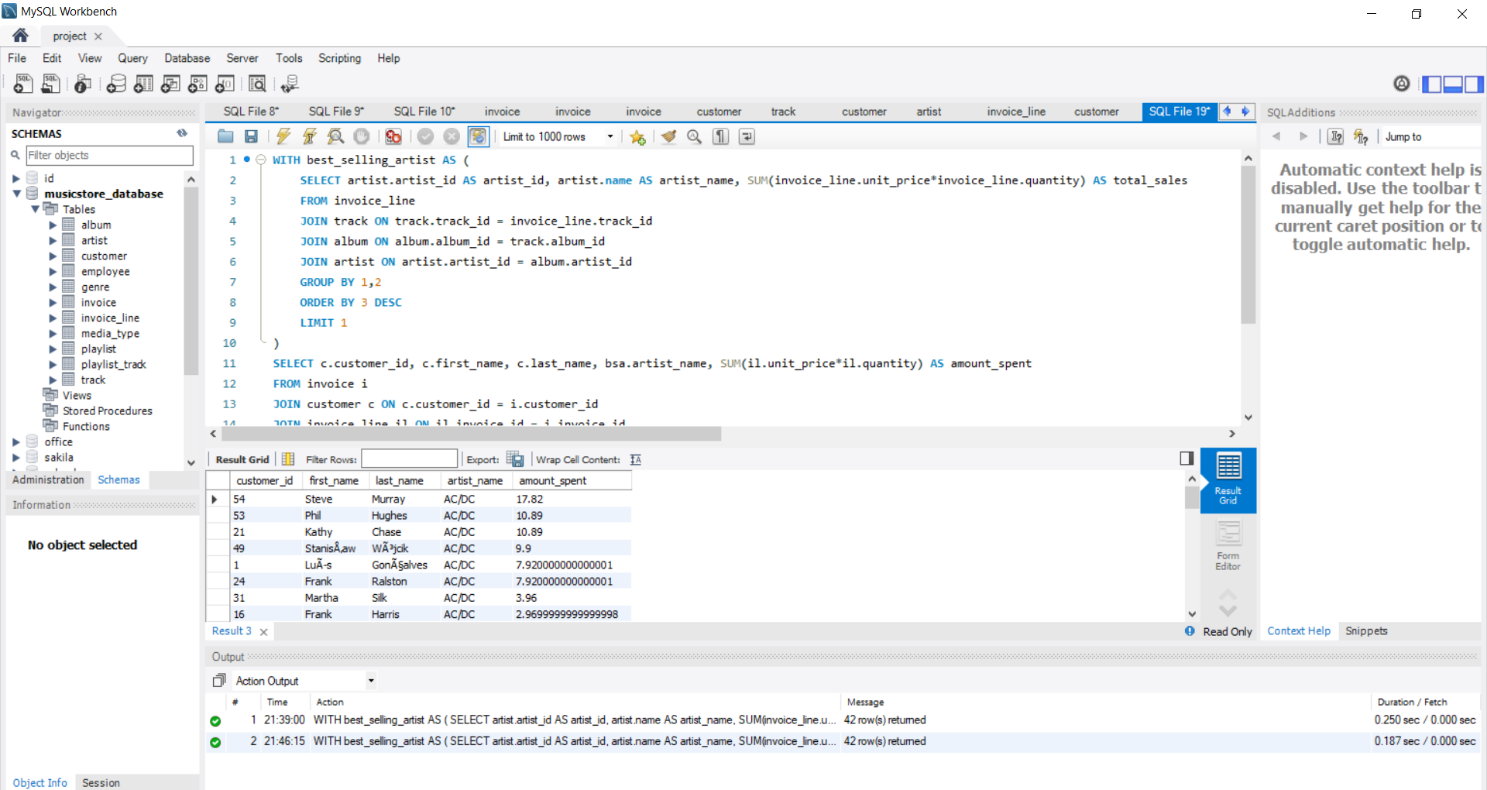
JOIN best\_selling\_artist bsa ON bsa.artist\_id = alb.artist\_id

GROUP BY 1,2,3,4

ORDER BY 5 DESC;

**Output :-**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 | LuÃ­s | GonÃ§alves | AC/DC | 7.920000000000001 |
| 24 | Frank | Ralston | AC/DC | 7.920000000000001 |
| 31 | Martha | Silk | AC/DC | 3.96 |
| 16 | Frank | Harris | AC/DC | 2.9699999999999998 |
| 42 | Wyatt | Girard | AC/DC | 2.9699999999999998 |
| 6 | Helena | HolÃ½ | AC/DC | 2.9699999999999998 |
| 38 | Niklas | SchrÃ¶der | AC/DC | 2.9699999999999998 |
| 35 | Madalena | Sampaio | AC/DC | 2.9699999999999998 |
| 44 | Terhi | HÃ¤mÃ¤lÃ¤inen | AC/DC | 2.9699999999999998 |
| 9 | Kara | Nielsen | AC/DC | 1.98 |
| 34 | JoÃ£o | Fernandes | AC/DC | 1.98 |
| 57 | Luis | Rojas | AC/DC | 1.98 |
| 27 | Patrick | Gray | AC/DC | 1.98 |
| 20 | Dan | Miller | AC/DC | 1.98 |
| 30 | Edward | Francis | AC/DC | 1.98 |
| 5 | FrantiÅ¡ek | WichterlovÃ¡ | AC/DC | 1.98 |
| 47 | Lucas | Mancini | AC/DC | 0.99 |
| 43 | Isabelle | Mercier | AC/DC | 0.99 |
| 19 | Tim | Goyer | AC/DC | 0.99 |
| 39 | Camille | Bernard | AC/DC | 0.99 |
| 8 | Daan | Peeters | AC/DC | 0.99 |
| 15 | Jennifer | Peterson | AC/DC | 0.99 |
| 58 | Manoj | Pareek | AC/DC | 0.99 |
| 46 | Hugh | O'Reilly | AC/DC | 0.99 |
| 32 | Aaron | Mitchell | AC/DC | 0.99 |
| 45 | Ladislav | KovÃ¡cs | AC/DC | 0.99 |
| 29 | Robert | Brown | AC/DC | 0.99 |
| 26 | Richard | Cunningham | AC/DC | 0.99 |
| 17 | Jack | Smith | AC/DC | 0.99 |
| 14 | Mark | Philips | AC/DC | 0.99 |
| 2 | Leonie | KÃ¶hler | AC/DC | 0.99 |
| 56 | Diego | GutiÃ©rrez | AC/DC | 0.99 |
| 48 | Johannes | Van der Berg | AC/DC | 0.99 |
| 13 | Fernanda | Ramos | AC/DC | 0.99 |
| 55 | Mark | Taylor | AC/DC | 0.99 |
| 7 | Astrid | Gruber | AC/DC | 0.99 |
| 59 | Rishabh | Mishra | AC/DC | 0.99 |
| 10 | Eduardo | Martins | AC/DC | 0.99 |



**Steps to Write and Execute the Query:**

#### **1. Create a Temporary Table for the Best-Selling Artist**

* First, we created a temporary table called best\_selling\_artist.
* In this part:
  + We calculated total sales for each artist by multiplying **unit price × quantity** of tracks sold.
  + We joined these tables:  
    invoice\_line → track → album → artist to connect sales to each artist.
  + We grouped the results by artist and sorted them by total sales.
  + Then we picked the **top-selling artist** using LIMIT 1.

**2. Find Customers Who Bought Music from That Artist**

* We joined the invoice, customer, and invoice\_line tables to get the **purchase details of each customer**.
* We also joined the track and album tables to find out which tracks belong to the best-selling artist.
* Then we joined with the temporary table best\_selling\_artist to **only include purchases related to that artist**.

**3. Group and Calculate Spending**

* We grouped the data by customer to calculate how much each person spent on the top artist's music.
* We used the same formula again: **unit price × quantity** to calculate the amount.
* Finally, we sorted the results by amount spent in descending order to find the customer who spent the most.