

Music Store Database Project Report

Easy Level Queries

Q1: Most Senior Employee Based on Job Title

Explanation: This query retrieves all employee records and sorts them by the title column in descending order. The assumption is that more senior positions have titles that sort later alphabetically (like "Senior Manager" vs "Junior Associate"). The LIMIT 1 returns only the top result - the most senior employee.

```
SELECT *  
  
FROM employee  
  
ORDER BY title DESC  
  
LIMIT 1;
```

1
2
3
4

SELECT *
FROM employee
ORDER BY title DESC
LIMIT 1;

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

Fetch rows:

	employee_id	last_name	first_name	title	reports_to	levels	birthdate	hire_date	address	city
▶	3	Peacock	Jane	Sales Support Agent	2	L1	29-08-1973 00:00	01-04-2017 00:00	1111 6 Ave SW	Calga

Result Grid
Form Editor
Field Types
Query Stats
Execution Plan

Q2: Countries with Most Invoices

Explanation: This query groups all invoices by their billing_country and counts how many invoices exist for each country. The results are sorted from highest to lowest invoice count, showing which countries generate the most business.

```
SELECT billing_country, COUNT(*) AS invoice_count
```

```
FROM invoice
```

```
GROUP BY billing_country
```

```
ORDER BY invoice_count DESC;
```

The screenshot shows a SQL query editor with the following query:

```
1 • SELECT billing_country, COUNT(*) AS invoice_count
2 FROM invoice
3 GROUP BY billing_country
4 ORDER BY invoice_count DESC;
```

Below the query editor, the results are displayed in a table. The table has two columns: **billing_country** and **invoice_count**. The results are sorted in descending order of invoice count.

billing_country	invoice_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
Hungary	10
Sweden	10

The interface also includes a sidebar with options like **Result Grid**, **Form Editor**, **Field Types**, **Query Stats**, and **Execution Plan**. The bottom status bar indicates **Result 2** and **Read Only**.

Q3: Top 3 Invoice Totals

Explanation: This simple query sorts all invoices by their total amount in descending order and returns only the top 3 records. This identifies the three largest individual transactions in the database.

SELECT *

FROM invoice

ORDER BY total DESC

LIMIT 3;

The screenshot shows a database query editor interface. At the top, a SQL query is entered in a text area:

```
1 • SELECT *
2 FROM invoice
3 ORDER BY total DESC
4 LIMIT 3;
```

Below the query editor, the results are displayed in a table grid. The table has the following columns: invoice_id, customer_id, invoice_date, billing_address, billing_city, billing_state, billing_country, billing_postal_code, and total. The results are sorted by total amount in descending order, showing the top 3 invoices.

invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	billing_country	billing_postal_code	total
183	42	2018-02-09 00:00:00	9, Place Louis Barthou	Bordeaux	NULL	France	33000	23.76
92	32	2017-07-02 00:00:00	696 Osborne Street	Winnipeg	MB	Canada	R3L 2B9	19.80
31	3	2017-02-21 00:00:00	1498 rue Bélanger	Montréal	QC	Canada	H2G 1A7	19.80
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

The interface includes a toolbar with various icons for editing, exporting, and filtering. On the right side, there is a sidebar with options like 'Result Grid', 'Form Editor', 'Field Types', 'Query Stats', and 'Execution Plan'. The bottom of the interface shows a tab labeled 'invoice 3' and an 'Apply' button.

Q4: City with Highest Total Invoice Amount

Explanation: This query calculates the sum of all invoice totals for each city (using GROUP BY billing_city), then sorts these aggregated amounts to find the single city with the highest total sales. This helps identify the best location for promotional events.

SELECT billing_city, SUM(total) AS total_amount

FROM invoice

GROUP BY billing_city

ORDER BY total_amount DESC

LIMIT 1;

```
1 • SELECT billing_city, SUM(total) AS total_amount
2 FROM invoice
3 GROUP BY billing_city
4 ORDER BY total_amount DESC
5 LIMIT 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows: |

	billing_city	total_amount
▶	Prague	273.24

Result 4 x | Read Only

Form Editor
Field Types
Query Stats
Execution Plan

Q5: Customer Who Spent the Most Money

Explanation: This query joins the customer and invoice tables to calculate the total spending for each customer. By grouping on customer details and summing all their invoice totals, then sorting in descending order, we identify the top-spending customer.

```
SELECT c.customer_id, c.first_name, c.last_name, SUM(i.total) AS total_spent
```

```
FROM customer c
```

```
JOIN invoice i ON c.customer_id = i.customer_id
```

```
GROUP BY c.customer_id
```

```
ORDER BY total_spent DESC
```

```
LIMIT 1;
```

```
1 • SELECT c.customer_id, c.first_name, c.last_name, SUM(i.total) AS total_spent
2 FROM customer c
3 JOIN invoice i ON c.customer_id = i.customer_id
4 GROUP BY c.customer_id
5 ORDER BY total_spent DESC
6 LIMIT 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows: |

	customer_id	first_name	last_name	total_spent
▶	5	František	Wichterlová	144.54

Result 5 x | Read Only

Result Grid
Form Editor
Field Types
Query Stats
Execution Plan

Moderate Level Queries

Q1: Customers Who Listen to Rock Music

Explanation: This complex join connects customers to tracks they've purchased through invoices. The WHERE clause filters for only Rock genre tracks. DISTINCT ensures each customer is listed only once, even if they've purchased multiple Rock tracks. Results are ordered by email for easy reading.

```
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 i.invoice_id = il.invoice_id
```

```
JOIN track t ON il.track_id = t.track_id
```

```
JOIN genre g ON t.genre_id = g.genre_id
```

WHERE g.name = 'Rock'

ORDER BY c.email;

```
1 • SELECT DISTINCT c.email, c.first_name, c.last_name
2 FROM customer c
3 JOIN invoice i ON c.customer_id = i.customer_id
4 JOIN invoice_line il ON i.invoice_id = il.invoice_id
5 JOIN track t ON il.track_id = t.track_id
6 JOIN genre g ON t.genre_id = g.genre_id
7 WHERE g.name = 'Rock'
8 ORDER BY c.email;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	email	first_name	last_name
▶	aaronmitchell@yahoo.ca	Aaron	Mitchell
	alero@uol.com.br	Alexandre	Rocha
	astrid.gruber@apple.at	Astrid	Gruber
	bjorn.hansen@yahoo.no	Björn	Hansen
	camille.bernard@yahoo.fr	Camille	Bernard
	daan_peeters@apple.be	Daan	Peeters
	diego.gutierrez@yahoo.ar	Diego	Gutiérrez
	dmiller@comcast.com	Dan	Miller
	dominiquedefebvre@gmail.com	Dominique	Lefebvre
	edfrancis@yahoo.ca	Edward	Francis
	eduardo@woodstock.com.br	Eduardo	Martins
	ellie.sullivan@shaw.ca	Ellie	Sullivan
	emma_jones@hotmail.com	Emma	Jones
	enrique_munoz@yahoo.es	Enrique	Muñoz
	fernadaramos4@uol.com.br	Fernanda	Ramos
	fharris@google.com	Frank	Harris
	fralston@gmail.com	Frank	Ralston
	frantisekw@jetbrains.com	František	Wichterlová

Result 6 x | Read Only

Result Grid
Form Editor
Field Types
Query Stats
Execution Plan

Q2: Top 10 Rock Artists by Track Count

Explanation: This query navigates from artists through albums to tracks, counting how many Rock tracks each artist has. The GROUP BY aggregates tracks per artist, and ORDER BY track_count DESC with LIMIT 10 shows the most prolific Rock artists in the catalog

SELECT ar.artist_id, ar.name, COUNT(*) AS track_count

FROM artist ar

JOIN album al ON ar.artist_id = al.artist_id

JOIN track t ON al.album_id = t.album_id

JOIN genre g ON t.genre_id = g.genre_id

WHERE g.name = 'Rock'

GROUP BY ar.artist_id

ORDER BY track_count DESC

LIMIT 10;

```
1 • SELECT ar.artist_id, ar.name, COUNT(*) AS track_count
2 FROM artist ar
3 JOIN album al ON ar.artist_id = al.artist_id
4 JOIN track t ON al.album_id = t.album_id
5 JOIN genre g ON t.genre_id = g.genre_id
6 WHERE g.name = 'Rock'
7 GROUP BY ar.artist_id, ar.name
8 ORDER BY track_count DESC
9 LIMIT 10;
```

Result Grid

	artist_id	name	track_count
▶	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

Result 7 x Read Only

Q3: Tracks Longer Than Average Length

Explanation: The subquery (SELECT AVG(milliseconds) FROM track) calculates the average track length. The main query then finds all tracks longer than this average, presenting them from longest to shortest. This helps identify unusually long tracks

SELECT name, milliseconds

FROM track

WHERE milliseconds > (SELECT AVG(milliseconds) FROM track)

ORDER BY milliseconds DESC;

```

1 • SELECT name, milliseconds
2 FROM track
3 WHERE milliseconds > (SELECT AVG(milliseconds) FROM track)
4 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 The	494393
You Oughta Know (Alternate)	491885
Terra	482429
Snoopy's search-Red baron	456071
Sozinho (Hitmakers Classic Mix)	436636
Master Of Puppets	436453
Stone Crazy	433397
Snowblind	420022
Computadores Fazem Arte	404323

track 8 x Read Only

Advanced Level Queries

Q1: Customer Spending per Artist

Explanation: This uses a CTE (Common Table Expression) named artist_sales to first calculate how much was spent on each artist per invoice. The main query then joins this with customer data to show how much each customer spent on each artist. This provides detailed customer preference data.

WITH artist_sales AS (

SELECT

il.invoice_id,

ar.artist_id,

ar.name AS artist_name,


```
        SUM(il.unit_price * il.quantity) AS amount_spent

FROM invoice_line il

JOIN track t ON il.track_id = t.track_id

JOIN album al ON t.album_id = al.album_id

JOIN artist ar ON al.artist_id = ar.artist_id

GROUP BY il.invoice_id, ar.artist_id

)
```

```
SELECT

    c.customer_id,

    c.first_name,

    c.last_name,

    as.artist_name,

    SUM(as.amount_spent) AS total_spent

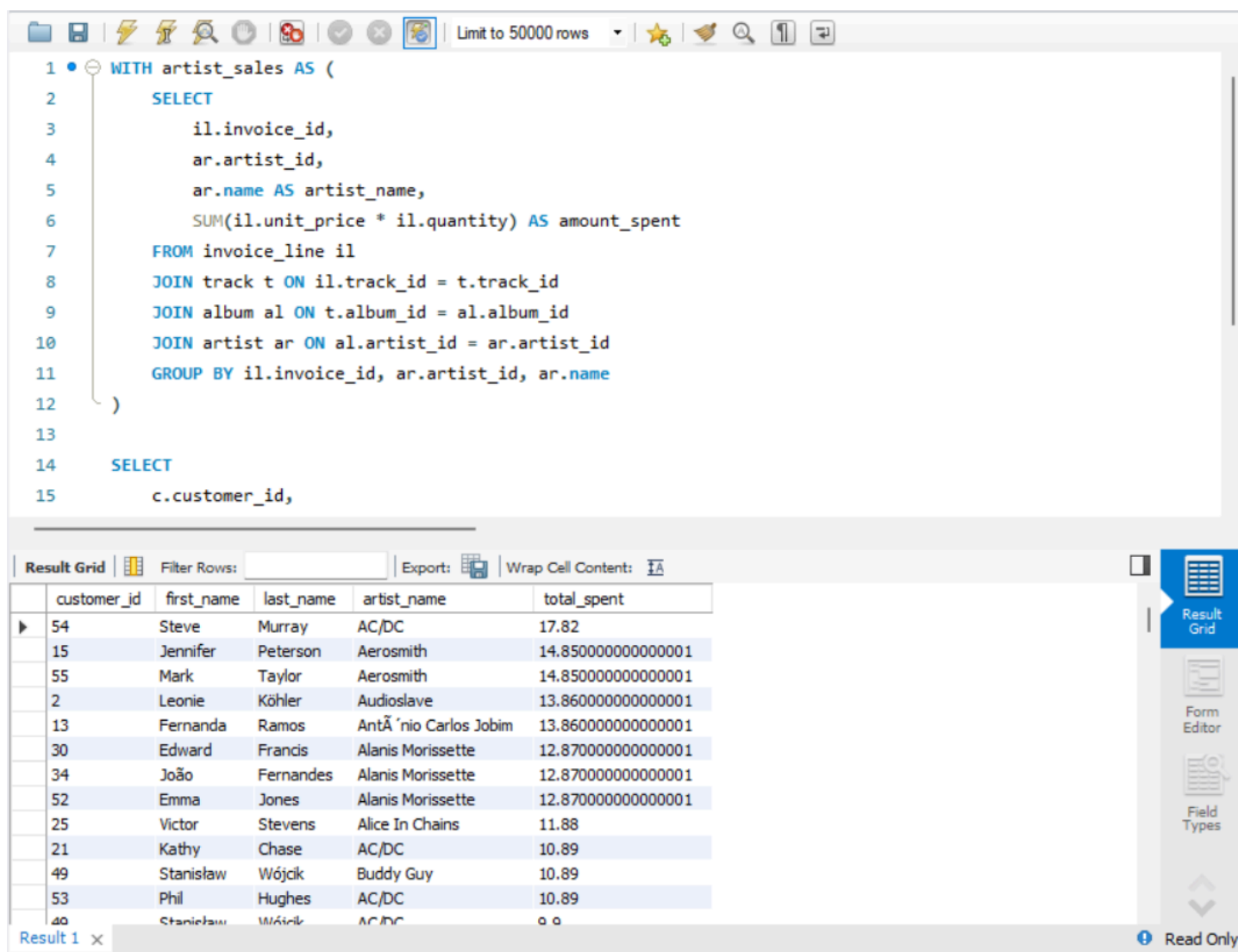
FROM customer c

JOIN invoice i ON c.customer_id = i.customer_id

JOIN artist_sales as ON i.invoice_id = as.invoice_id

GROUP BY c.customer_id, as.artist_id

ORDER BY total_spent DESC;
```



The screenshot shows a SQL IDE interface. At the top, there's a toolbar with various icons and a dropdown menu set to 'Limit to 50000 rows'. Below the toolbar is a text editor with a SQL query. The query is as follows:

```

1  WITH artist_sales AS (
2      SELECT
3          il.invoice_id,
4          ar.artist_id,
5          ar.name AS artist_name,
6          SUM(il.unit_price * il.quantity) AS amount_spent
7      FROM invoice_line il
8      JOIN track t ON il.track_id = t.track_id
9      JOIN album al ON t.album_id = al.album_id
10     JOIN artist ar ON al.artist_id = ar.artist_id
11     GROUP BY il.invoice_id, ar.artist_id, ar.name
12 )
13
14 SELECT
15     c.customer_id,

```

Below the text editor is a 'Result Grid' tab. It shows a table with 6 columns: customer_id, first_name, last_name, artist_name, and total_spent. The table contains 15 rows of data. The first row is highlighted. To the right of the table is a sidebar with icons for 'Result Grid', 'Form Editor', and 'Field Types'. At the bottom right, there is a 'Read Only' indicator.

	customer_id	first_name	last_name	artist_name	total_spent
▶	54	Steve	Murray	AC/DC	17.82
	15	Jennifer	Peterson	Aerosmith	14.850000000000001
	55	Mark	Taylor	Aerosmith	14.850000000000001
	2	Leonie	Köhler	Audioslave	13.860000000000001
	13	Fernanda	Ramos	Antônio Carlos Jobim	13.860000000000001
	30	Edward	Frands	Alanis Morissette	12.870000000000001
	34	João	Fernandes	Alanis Morissette	12.870000000000001
	52	Emma	Jones	Alanis Morissette	12.870000000000001
	25	Victor	Stevens	Alice In Chains	11.88
	21	Kathy	Chase	AC/DC	10.89
	49	Stanisław	Wójcik	Buddy Guy	10.89
	53	Phil	Hughes	AC/DC	10.89
	40	Stanisław	Wójcik	AC/DC	0.0

Q2: Most Popular Genre by Country

Explanation: The CTE calculates purchase counts for each genre in each country, using RANK() with PARTITION BY to rank genres within each country. The main query filters for only the top-ranked genre (rank = 1) in each country, showing regional music preferences.

WITH genre_sales_by_country AS (

SELECT

i.billing_country,

g.genre_id,

g.name AS genre_name,

COUNT(*) AS purchase_count,

RANK() OVER (PARTITION BY i.billing_country ORDER BY COUNT(*) DESC) AS genre_rank

```

FROM invoice i

JOIN invoice_line il ON i.invoice_id = il.invoice_id

JOIN track t ON il.track_id = t.track_id

JOIN genre g ON t.genre_id = g.genre_id

GROUP BY i.billing_country, g.genre_id

)

SELECT billing_country, genre_name, purchase_count

FROM genre_sales_by_country

WHERE genre_rank = 1

ORDER BY billing_country;

```

```

1  •  WITH genre_sales_by_country AS (
2      SELECT
3          i.billing_country,
4          g.genre_id,
5          g.name AS genre_name,
6          COUNT(*) AS purchase_count,
7          RANK() OVER (PARTITION BY i.billing_country ORDER BY COUNT(*) DESC) AS genre_rank
8      FROM invoice i
9      JOIN invoice_line il ON i.invoice_id = il.invoice_id
10     JOIN track t ON il.track_id = t.track_id
11     JOIN genre g ON t.genre_id = g.genre_id
12     GROUP BY i.billing_country, g.genre_id, g.name
13 )
14
15 SELECT billing_country, genre_name, purchase_count
16 FROM genre_sales_by_country
17 WHERE genre_rank = 1

```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	billing_country	genre_name	purchase_count
▶	Argentina	Rock	1
	Australia	Rock	18
	Austria	Rock	6
	Belgium	Rock	5
	Brazil	Rock	26
	Canada	Rock	57
	Chile	Rock	7
	Czech Republic	Rock	14
	Denmark	Rock	6

Result Grid

Form Editor

Read Only

Q3: Top-Spending Customer per Country

Explanation: Similar to Q2, this uses a CTE with window functions to rank customers by spending within each country. The main query selects only the top-spending customer (rank = 1) from each country, providing valuable information about key customers in different markets.

WITH customer_spending AS (

SELECT

c.customer_id,

c.first_name,

c.last_name,

i.billing_country,

SUM(i.total) AS total_spent,

RANK() OVER (PARTITION BY i.billing_country ORDER BY SUM(i.total) DESC) AS
spending_rank

FROM customer c

JOIN invoice i ON c.customer_id = i.customer_id

GROUP BY c.customer_id, i.billing_country

)

SELECT customer_id, first_name, last_name, billing_country, total_spent

FROM customer_spending

WHERE spending_rank = 1

ORDER BY billing_country;

```


3      c.customer_id,
4      c.first_name,
5      c.last_name,
6      i.billing_country,
7      SUM(i.total) AS total_spent,
8      RANK() OVER (PARTITION BY i.billing_country ORDER BY SUM(i.total) DESC) AS spending_rank
9  FROM customer c
10 JOIN invoice i ON c.customer_id = i.customer_id
11 GROUP BY c.customer_id, i.billing_country
12 )
13
14 SELECT customer_id, first_name, last_name, billing_country, total_spent
15 FROM customer_spending
16 WHERE spending_rank = 1
17 ORDER BY billing_country;

```

Result Grid  Filter Rows: Export:  Wrap Cell Content: 

	customer_id	first_name	last_name	billing_country	total_spent
▶	56	Diego	Gutiérrez	Argentina	39.60
	55	Mark	Taylor	Australia	81.18
	7	Astrid	Gruber	Austria	69.30
	8	Daan	Peeters	Belgium	60.39
	1	Luís	Gonçalves	Brazil	108.90
	3	François	Tremblay	Canada	99.99
	57	Luis	Rojas	Chile	97.02
	5	František	Wichterlová	Czech Republic	144.54
	9	Kara	Nielsen	Denmark	37.62
	44	Terhi	Hämäläinen	Finland	79.20
	42	Wyatt	Girard	France	99.99
	37	Fynn	Zimmermann	Germany	94.05
	45	Ladislav	Kovács	Hungary	78.21

Result 2 x

 Read Only



Result
Grid



Form
Editor



Field
Types

