

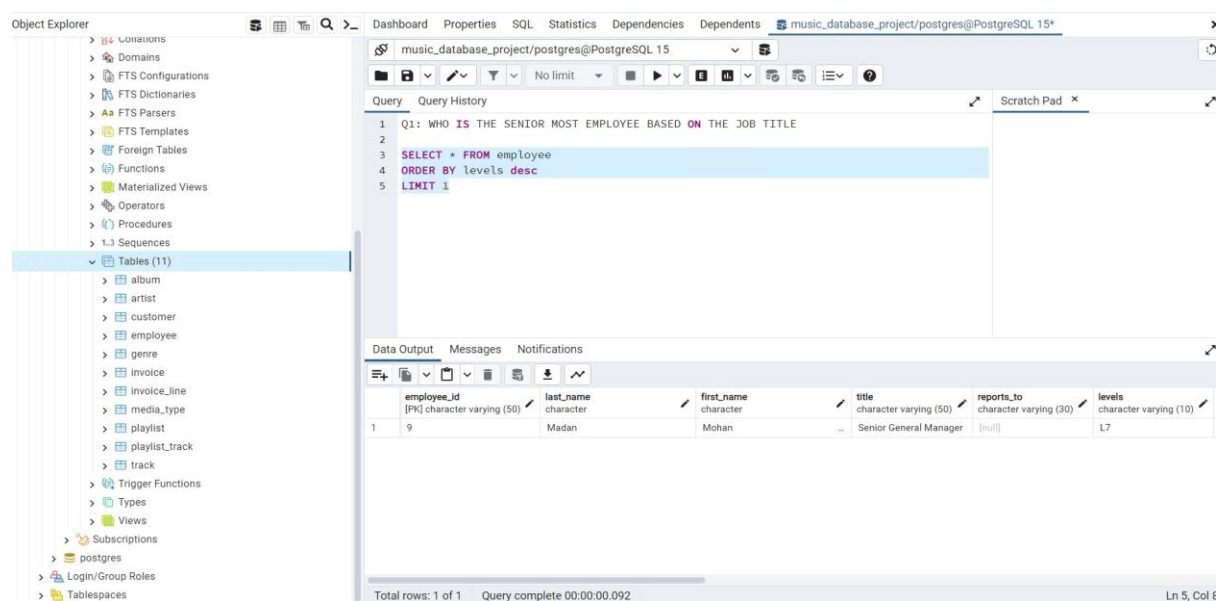
SQL PROJECT: -

This project uses a database of music CD sales data to answer a variety of questions about the business. The questions are divided into two categories:

- Business intelligence questions: These questions are designed to help the business understand its customers, products, and sales. For example, the questions in this category can be used to identify the most popular music genres, the best-selling artists, and the most profitable cities.
- Customer relationship management (CRM) questions: These questions are designed to help the business understand its customers and their buying habits. For example, the questions in this category can be used to identify the most loyal customers, the customers who are most likely to churn, and the customers who are most likely to spend more money.

The project uses a variety of SQL queries to answer the questions.

Q1: Who is the senior most employee based on job?



The screenshot shows a PostgreSQL IDE interface. On the left, the 'Object Explorer' pane displays a tree view of database objects, with 'Tables (11)' expanded. The main query editor displays the following SQL query:

```
1 Q1: WHO IS THE SENIOR MOST EMPLOYEE BASED ON THE JOB TITLE
2
3 SELECT * FROM employee
4 ORDER BY levels desc
5 LIMIT 1
```

Below the query editor, the 'Data Output' pane shows the results of the query in a table format:

	employee_id [PK] character varying (50)	last_name character	first_name character	title character varying (50)	reports_to character varying (30)	levels character varying (10)
1	9	Madan	Mohan	Senior General Manager	[null]	L7

The status bar at the bottom indicates 'Total rows: 1 of 1' and 'Query complete 00:00:00.092'.

- This query uses the job_title and employee_id tables to find the employee with the highest-ranking job title.

Q2: Which countries have the most Invoices?

The screenshot shows a PostgreSQL IDE interface. On the left, the 'Object Explorer' pane lists database objects, with 'Tables (11)' expanded. The main query editor displays the following SQL query:

```

1 Q2: WHICH COUNTRY HAVE MOST INVOICE
2
3
4 SELECT COUNT(*) as c, billing_country
5 from invoice
6 group by billing_country
7 order by c desc

```

The 'Data Output' pane shows the results of the query, sorted by the number of invoices (c) in descending order:

c	billing_country
131	USA
76	Canada
61	Brazil
50	France
41	Germany
30	Czech Republic
29	Portugal
28	United Kingdom
21	India
13	Chile
13	Ireland
11	Spain
11	Finland
10	Australia

The status bar at the bottom indicates 'Total rows: 24 of 24' and 'Query complete 00:00:00.096'.

- This query uses the invoices table to count the number of invoices from each country. The results are sorted by the number of invoices, with the country with the most invoices listed first.

Q3: What are top 3 values of total invoice?

The screenshot shows the same PostgreSQL IDE interface. The query editor displays the following SQL query:

```

1 Q3:WHAT ARE TOP 3 VALUES OF TOTAL INVOICE
2
3
4 SELECT total FROM invoice
5 ORDER BY total desc
6 limit 5

```

The 'Data Output' pane shows the results of the query, sorted by the total invoice value in descending order:

total
23.759999999999998
19.8
19.8
19.8
19.8

The status bar at the bottom indicates 'Total rows: 5 of 5' and 'Query complete 00:00:00.090'.

- This query uses the invoices table to find the top 3 values of the total column. The results are sorted by the total column, with the highest value listed first.

Q4: Which city has the best customers?

The screenshot shows a database IDE with the following components:

- Object Explorer:** A tree view on the left showing the database structure. The 'Tables (11)' folder is expanded, listing tables like album, artist, customer, employee, genre, invoice, invoice_line, media_type, playlist, playlist_track, track, and trigger_functions.
- Query Editor:** A central pane containing the following SQL query:


```
1 SELECT customer.customer_id, customer.first_name, customer.last_name,
2 SUM(invoice.total) AS total
3 FROM customer
4 JOIN invoice ON customer.customer_id = invoice.customer_id
5 GROUP BY customer.customer_id
6 ORDER BY total desc
7 LIMIT 5;
```
- Data Output:** A table at the bottom showing the results of the query. It has 5 rows and 4 columns: customer_id, first_name, last_name, and total.

customer_id	first_name	last_name	total
5	R	Madhav	144.54000000000002
6	Helena	Holy	128.7
46	Hugh	O'Reilly	114.83999999999997
58	Manoj	Pareek	111.86999999999999
1	Luis	Gonçalves	108.89999999999998
- Status Bar:** At the bottom, it indicates 'Total rows: 5 of 5' and 'Query complete 00:00:00.064'.

- This query uses the invoices and customers tables to find the city with the highest sum of invoice totals. The results are sorted by the sum of invoice totals, with the city with the highest total listed first

Q5: Who is the best customer?

The screenshot shows a database IDE with the following components:

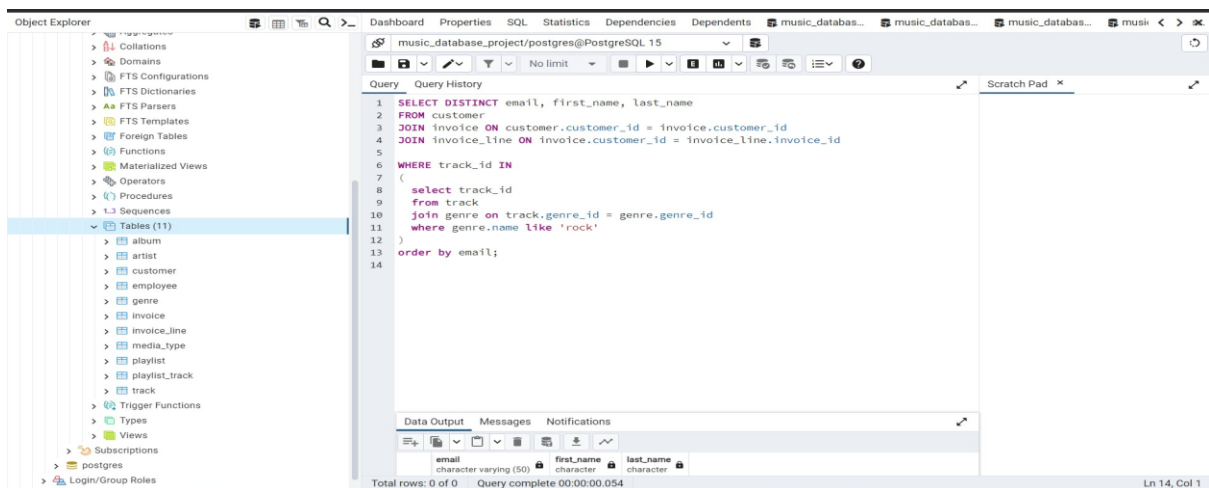
- Object Explorer:** A tree view on the left showing the database structure. The 'Tables (11)' folder is expanded, listing tables like album, artist, customer, employee, genre, invoice, invoice_line, media_type, playlist, playlist_track, track, and trigger_functions.
- Query Editor:** A central pane containing the following SQL query:


```
1 SELECT SUM(total) as invoice_total, billing_city
2 FROM invoice
3 GROUP BY billing_city
4 ORDER BY invoice_total DESC
5 limit 3;
```
- Data Output:** A table at the bottom showing the results of the query. It has 3 rows and 2 columns: invoice_total and billing_city.

invoice_total	billing_city
273.24000000000007	Prague
169.29	Mountain View
166.32	London
- Status Bar:** At the bottom, it indicates 'Total rows: 3 of 3' and 'Query complete 00:00:00.093'.

- This query uses the invoices and customers tables to find the customer who has spent the most money. The results are sorted by the total amount spent, with the customer who has spent the most money listed first.

Q6: 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



The screenshot shows a PostgreSQL IDE interface. On the left, the 'Object Explorer' pane lists database objects, with 'Tables (11)' expanded. The main query editor contains the following SQL code:

```

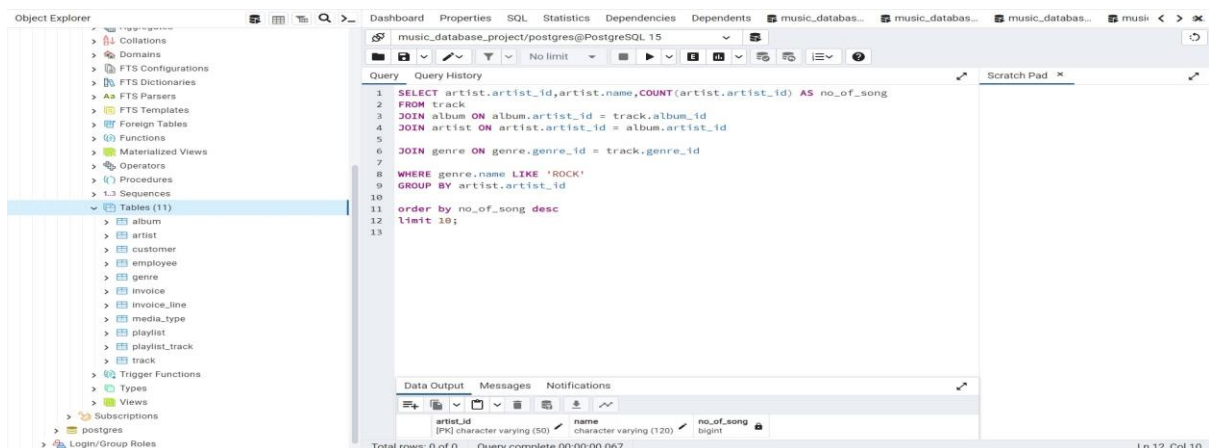
1 SELECT DISTINCT email, first_name, last_name
2 FROM customer
3 JOIN invoice ON customer.customer_id = invoice.customer_id
4 JOIN invoice_line ON invoice.customer_id = invoice_line.invoice_id
5
6 WHERE track_id IN
7 (
8     select track_id
9     from track
10    join genre on track.genre_id = genre.genre_id
11   where genre.name like 'rock'
12 )
13 order by email;
14

```

The 'Data Output' pane at the bottom shows the column headers: email, first_name, last_name. The status bar indicates 'Total rows: 0 of 0' and 'Query complete 00:00:00.054'.

- This query uses the customers and genres tables to find all customers who listen to rock music. The results are sorted by the customer's email address, starting with the letter "A".

Q7: Let's invite the artists who have written the most rock music in our dataset. Write a query that returns the Artist name and total track count of the top 10 rock bands



The screenshot shows a PostgreSQL IDE interface. On the left, the 'Object Explorer' pane lists database objects, with 'Tables (11)' expanded. The main query editor contains the following SQL code:

```

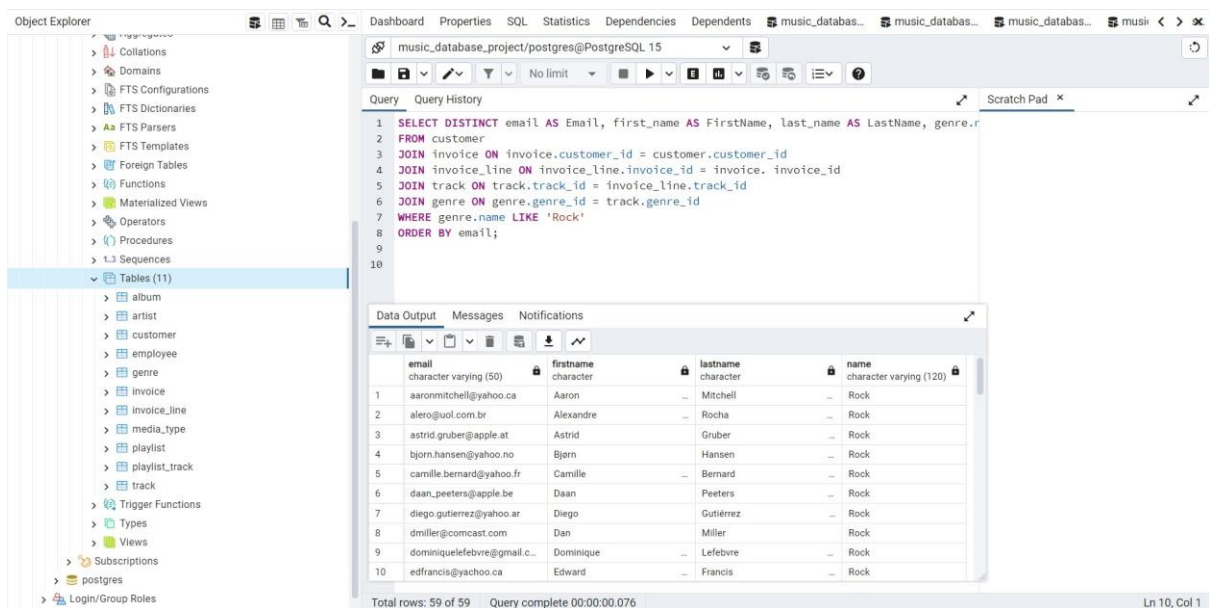
1 SELECT artist.artist_id, artist.name, COUNT(artist.artist_id) AS no_of_song
2 FROM track
3 JOIN album ON album.artist_id = track.album_id
4 JOIN artist ON artist.artist_id = album.artist_id
5
6 JOIN genre ON genre.genre_id = track.genre_id
7
8 WHERE genre.name LIKE 'ROCK'
9 GROUP BY artist.artist_id
10
11 order by no_of_song desc
12 limit 10;
13

```

The 'Data Output' pane at the bottom shows the column headers: artist_id, name, no_of_song. The status bar indicates 'Total rows: 0 of 0' and 'Query complete 00:00:00.067'.

- This query uses the artists and tracks tables to find the top 10 artists who have written the most rock music. The results are sorted by the number of tracks, with the artist with the most tracks listed first

Q8: 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



The screenshot shows a PostgreSQL IDE interface. On the left is the 'Object Explorer' showing a database schema with tables like album, artist, customer, employee, genre, invoice, invoice_line, media_type, playlist, playlist_track, track, and trigger_functions. The main window displays a SQL query in the 'Query' tab:

```

1 SELECT DISTINCT email AS Email, first_name AS FirstName, last_name AS LastName, genre.r
2 FROM customer
3 JOIN invoice ON invoice.customer_id = customer.customer_id
4 JOIN invoice_line ON invoice_line.invoice_id = invoice.invoice_id
5 JOIN track ON track.track_id = invoice_line.track_id
6 JOIN genre ON genre.genre_id = track.genre_id
7 WHERE genre.name LIKE 'Rock'
8 ORDER BY email;
9
10

```

Below the query editor, the 'Data Output' tab shows the results of the query. The results are sorted by email. The table has columns: email, first_name, last_name, and name. The data is as follows:

email	first_name	last_name	name
1 aaronmitchell@yahoo.ca	Aaron	Mitchell	Rock
2 alero@uol.com.br	Alexandre	Rocha	Rock
3 astrid.gruber@apple.at	Astrid	Gruber	Rock
4 bjorn.hansen@yahoo.no	Bjorn	Hansen	Rock
5 camille.bernard@yahoo.fr	Camille	Bernard	Rock
6 daan.peeters@apple.be	Daan	Peeters	Rock
7 diego.gutierrez@yahoo.ar	Diego	Gutiérrez	Rock
8 dmiller@comcast.com	Dan	Miller	Rock
9 dominiquelefebvre@gmail.c...	Dominique	Lefebvre	Rock
10 edfrancis@yahoo.ca	Edward	Francis	Rock

At the bottom, it shows 'Total rows: 59 of 59' and 'Query complete 00:00:00.076'.

- This query uses the tracks table to find all tracks that are longer than the average song length. The results are sorted by the song length, with the longest songs listed first.

Q9: Find how much amount spent by each customer on artists? Write a query to return customer name, artist name and total spent

- This query uses the invoices, customers, and tracks tables to find how much each customer has spent on each artist. The results are sorted by the customer name, with the customer who has spent the most money on each artist listed first.

Object Explorer

- Collations
- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (11)**
 - album
 - artist
 - customer
 - employee
 - genre
 - invoice
 - invoice_line
 - media_type
 - playlist
 - playlist_track
 - track
- Trigger Functions
- Types
- Views
- Subscriptions
- postgres
- Login/Group Roles

music_database_project/postgres@PostgreSQL 15

Query Query History

```

1 WITH best_selling_artist AS (
2 SELECT artist.artist_id AS artist_id, artist.name AS artist_name,
3 SUM(invoice_line.unit_price * invoice_line.quantity) AS total_sales
4 FROM invoice_line
5 JOIN track ON track.track_id = invoice_line.track_id
6 JOIN album ON album.album_id = track.album_id
7 JOIN artist ON artist.artist_id = album.artist_id
8 GROUP BY 1
9 ORDER BY 3 DESC
10 LIMIT 1)
11 SELECT c.customer_id, c.first_name, c.last_name, bsa.artist_name,
12 SUM(il.unit_price*il.quantity) AS ROUND(amount_spent,2)
13 FROM invoice i
14 JOIN customer c ON c.customer_id = i.customer_id
15 JOIN invoice_line il ON il.invoice_id = i.invoice_id
16 JOIN track t ON t.track_id = il.track_id
17 JOIN album alb ON alb.album_id = t.album_id
18 JOIN best_selling_artist bsa ON bsa.artist_id = alb.artist_id
19 GROUP BY 1,2,3,4
20 ORDER BY 5 DESC
21 LIMIT 3;

```

Data Output Messages Notifications

	customer_id	first_name	last_name	artist_name	amount_spent
	integer	character	character	character varying (120)	double precision
1	46	Hugh	O'Reilly	Queen	27.7199999999999985
2	38	Niklas	Schröder	Queen	18.81
3	3	François	Tremblay	Queen	17.82

Total rows: 3 of 3 Query complete 00:00:00.056 Ln 12, Col 54

- This is just a brief overview of the SQL project you have created