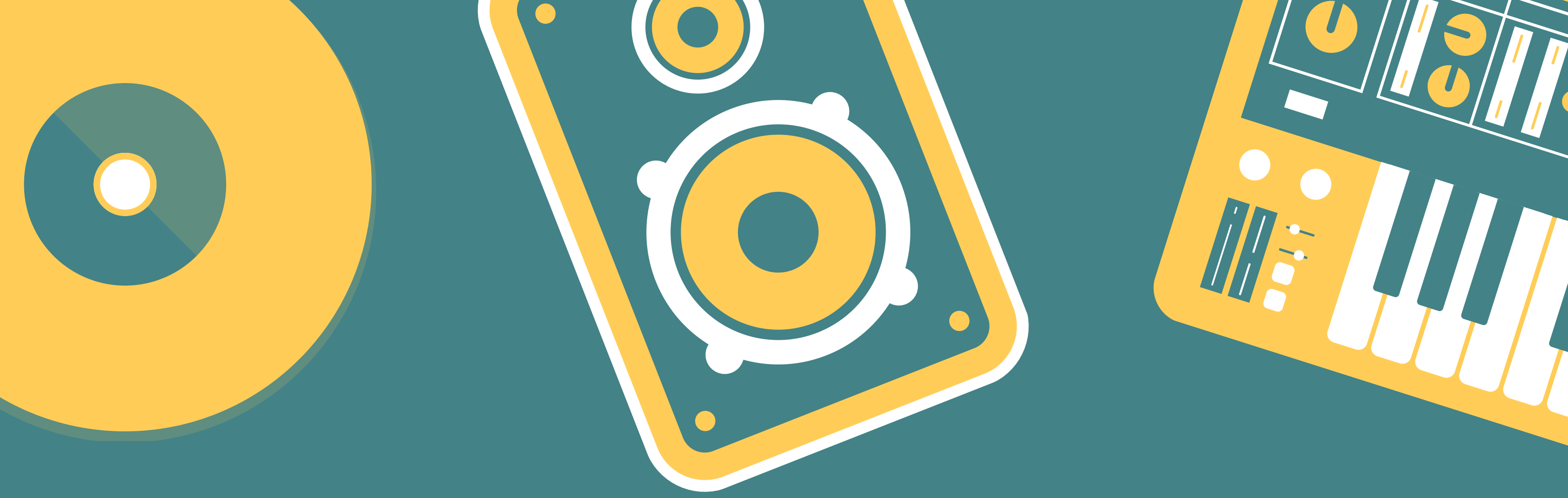
YOUR UNIVERSE OF MELODIES

TUNE SPHERE

Digital Music Store Analysis Project Report

ADVANCED SQL PROJECT

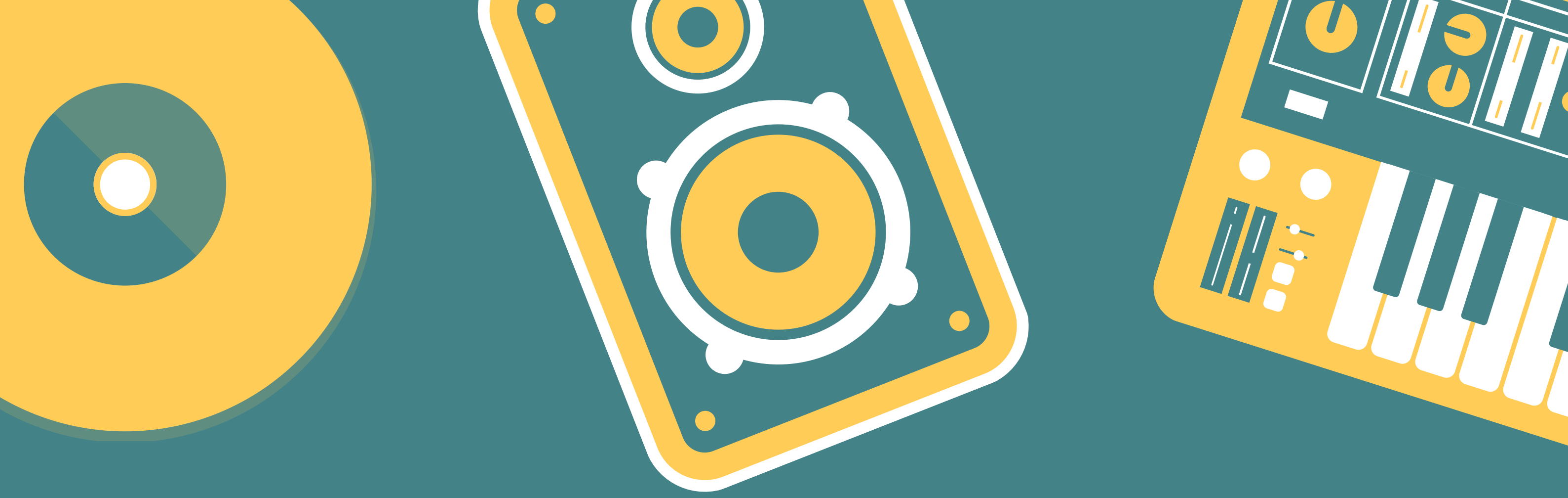
The primary goal of this project is to analyze the sales, customer behavior, and music preferences in a digital music store, Tune Sphere, using SQL. This analysis will help in understanding the store's performance, identifying popular genres and artists, and making data-driven decisions to enhance business strategies.



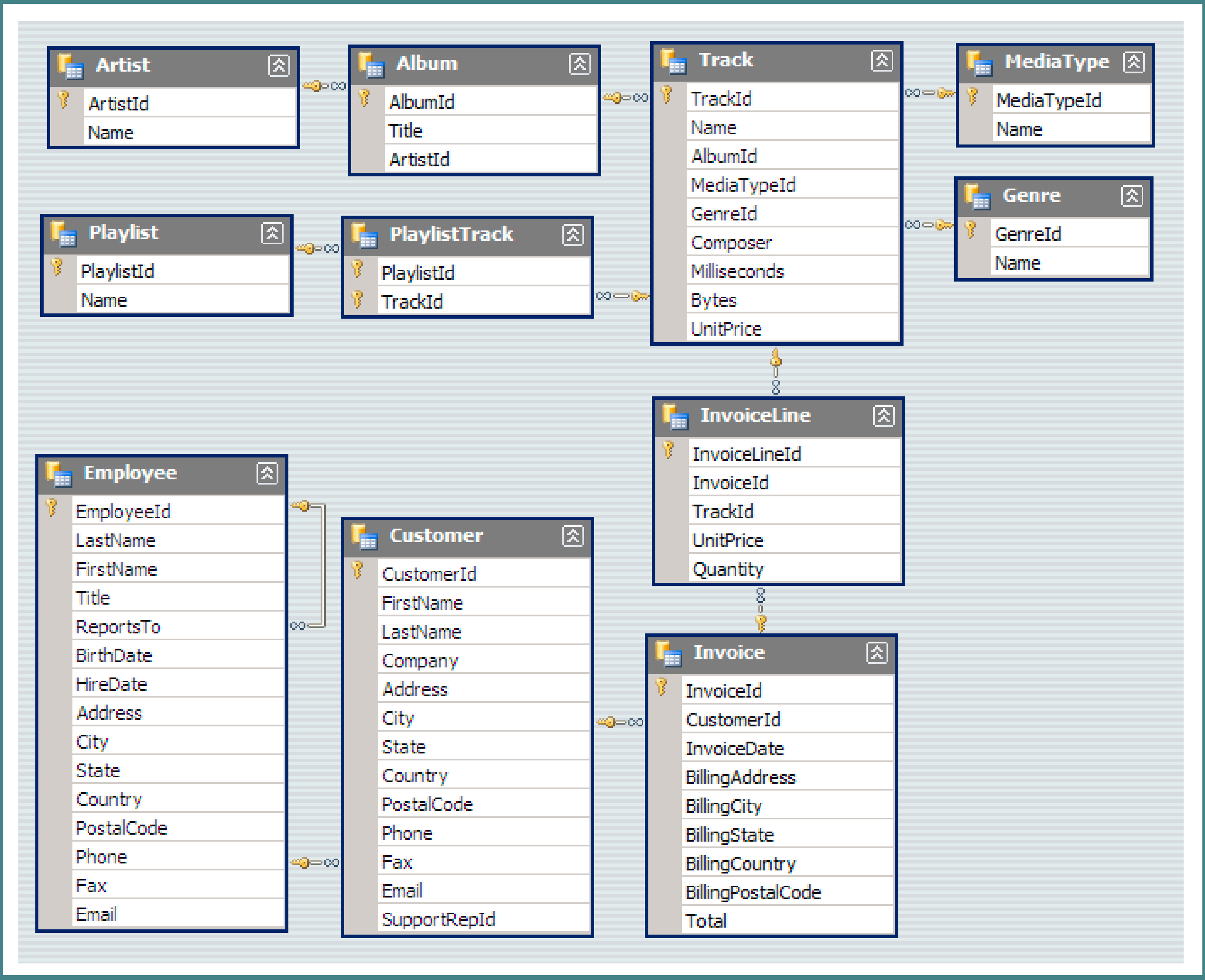
DATA SOURCES

The project will involve multiple datasets typically available in a digital music store database. I have used the following 11 tables for the same:

- **ALBUM**: Details of music albums including album ID, title, artist ID.
- **ARTIST**: Information about artists such as artist ID, name.
- **CUSTOMER**: Information about customers such as customer ID, name, email, phone, location.
- **EMPLOYEE**: Information about employees such as employee ID, name, title, hire date, birth date, email, phone, location.
- **GENRE**: Details of music genres such as genre ID, genre name
- **INVOICE**: Details of each transaction, including invoice ID, customer ID, purchase date, billing address, total amount.
- **INVOICE LINE**: Line items associated with each invoice, detailing the purchased tracks including invoice line ID, track ID, unit price, quantity.
- **MEDIA TYPE**: Information about file media such as media type name.
- **PLAYLIST**: Information about playlists such as playlist ID, playlist name.
- **PLAYLIST TRACK**: Information about playlist tracks such as playlist ID, track ID.
- **TRACK**: Details of tracks such as track ID, track name, composer, track size, track duration, price.



ER DIAGRAM



Artist
- ArtistId
- Name

Album
- AlbumId
- Title
- ArtistId

Track
- TrackId
- Name
- AlbumId
- MediaTypeId
- GenreId
- Composer
- Milliseconds
- Bytes
- UnitPrice

MediaType
- MediaTypeId
- Name

Genre
- GenreId
- Name

Playlist
- PlaylistId
- Name

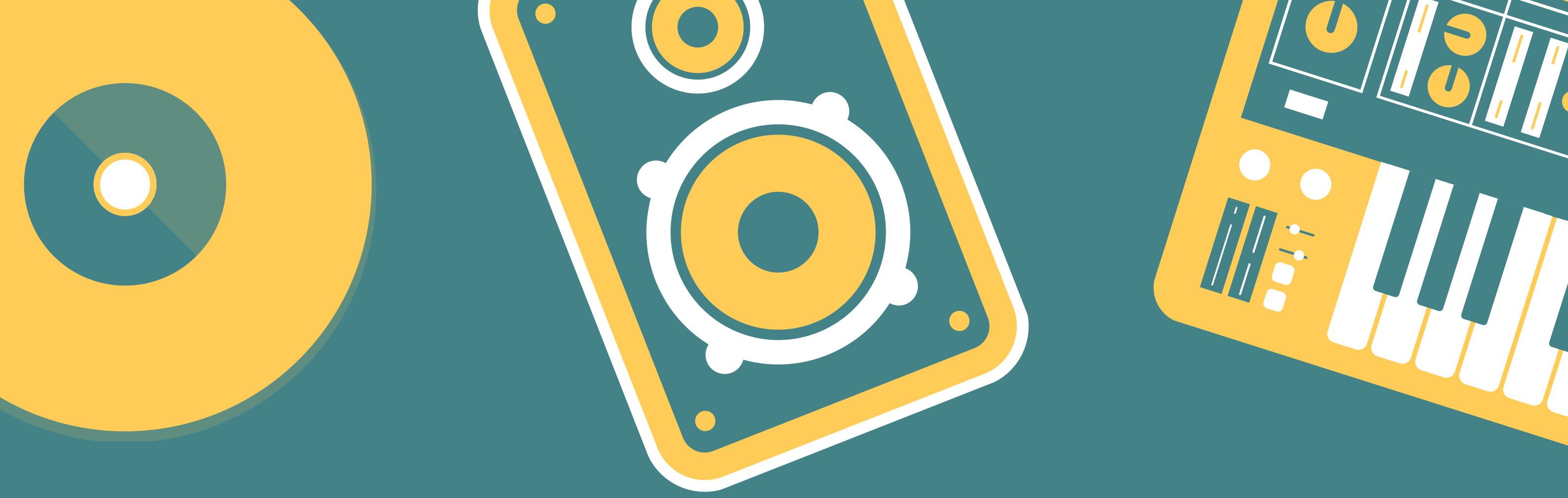
PlaylistTrack
- PlaylistId
- TrackId

InvoiceLine
- InvoiceLineId
- InvoiceId
- TrackId
- UnitPrice
- Quantity

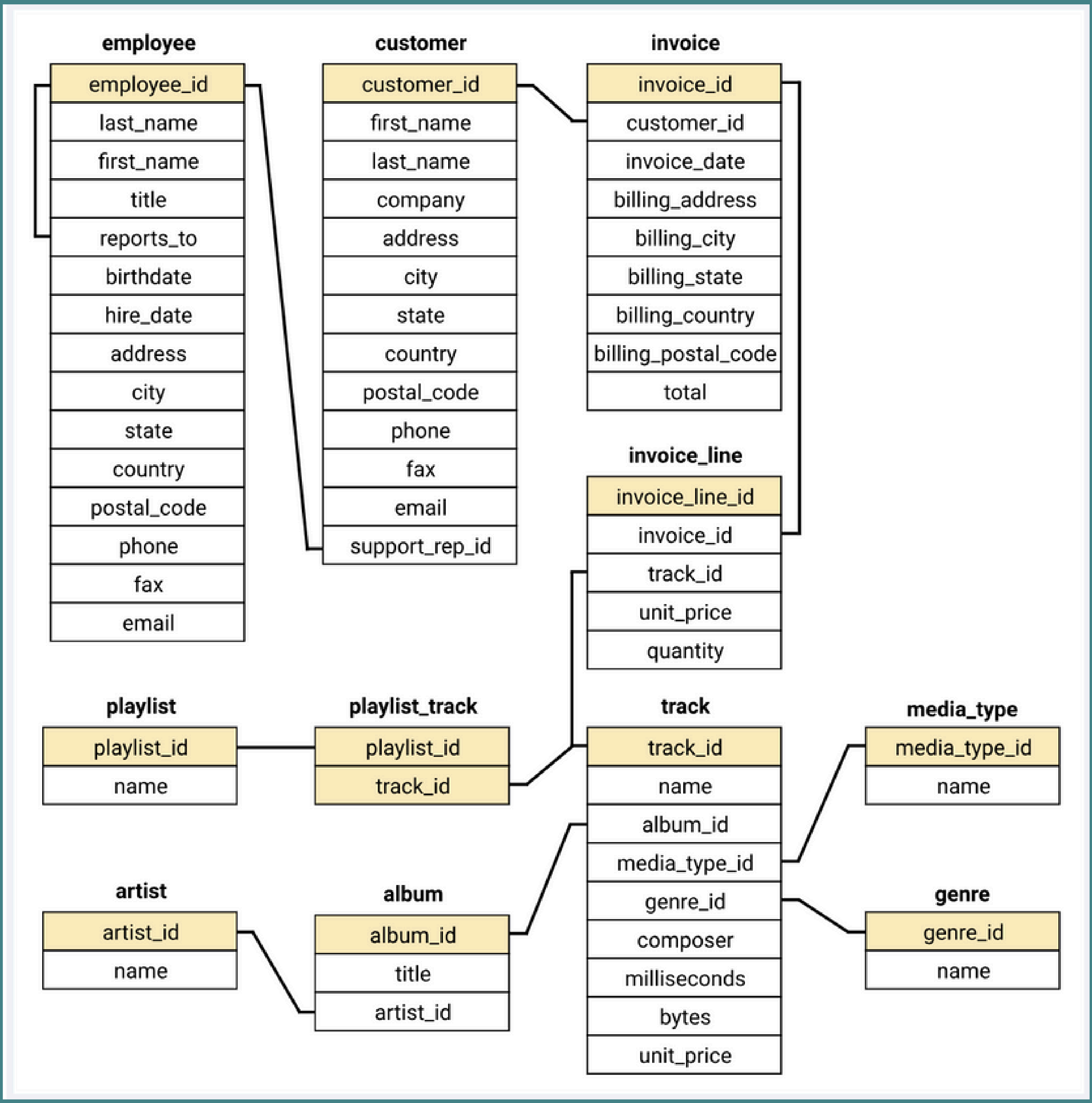
Employee
- EmployeeId
- LastName
- FirstName
- Title
- ReportsTo
- BirthDate
- HireDate
- Address
- City
- State
- Country
- PostalCode
- Phone
- Fax
- Email

Customer
- CustomerId
- FirstName
- LastName
- Company
- Address
- City
- State
- Country
- PostalCode
- Phone
- Fax
- Email
- SupportRepId

Invoice
- InvoiceId
- CustomerId
- InvoiceDate
- BillingAddress
- BillingCity
- BillingState
- BillingCountry
- BillingPostalCode
- Total



SCHEMA



employee

- employee_id
- last_name
- first_name
- title
- reports_to
- birthdate
- hire_date
- address
- city
- state
- country
- postal_code
- phone
- fax
- email

customer

- customer_id
- first_name
- last_name
- company
- address
- city
- state
- country
- postal_code
- phone
- fax
- email
- support_rep_id

invoice

- invoice_id
- customer_id
- invoice_date
- billing_address
- billing_city
- billing_state
- billing_country
- billing_postal_code
- total

invoice_line

- invoice_line_id
- invoice_id
- track_id
- unit_price
- quantity

playlist

- playlist_id
- name

playlist_track

- playlist_id
- track_id

track

- track_id
- name
- album_id
- media_type_id
- genre_id
- composer
- milliseconds
- bytes
- unit_price

media_type

- media_type_id
- name

artist

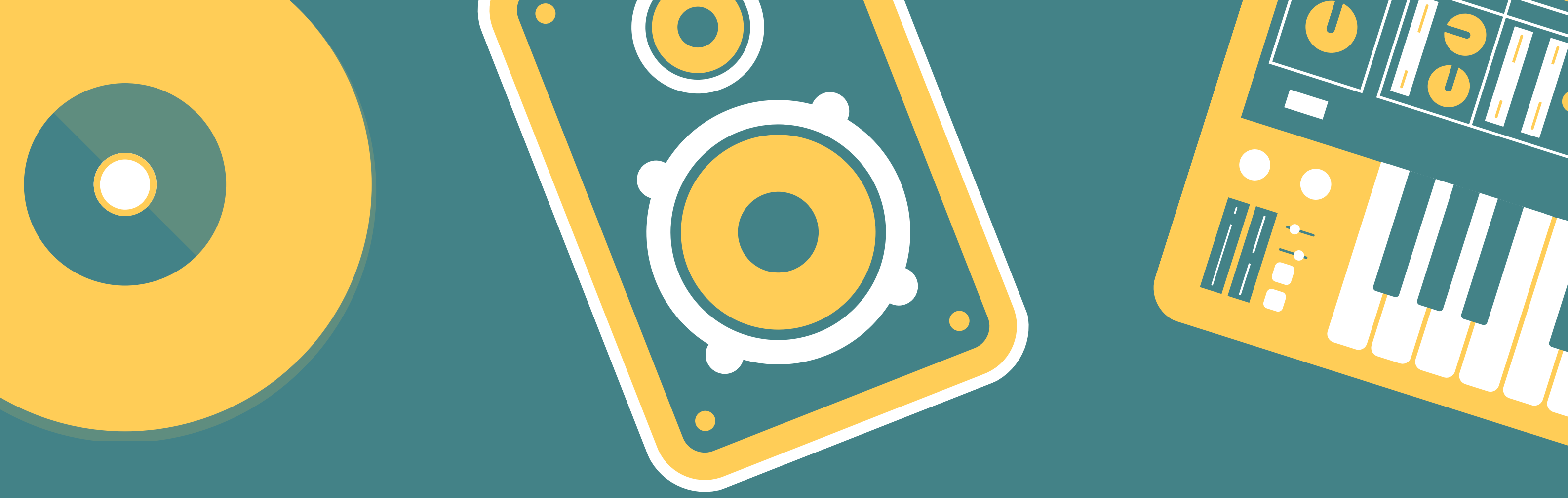
- artist_id
- name

album

- album_id
- title
- artist_id

genre

- genre_id
- name



KEY ANALYSIS AREAS

1. SALES ANALYSIS

- Total revenue generated over a specific period.
- Trends in sales over time (daily, weekly, monthly).
- Breakdown of sales by genre, artist, album, and track.
- Identification of top-selling items and their contribution to revenue.

2. CUSTOMER ANALYSIS

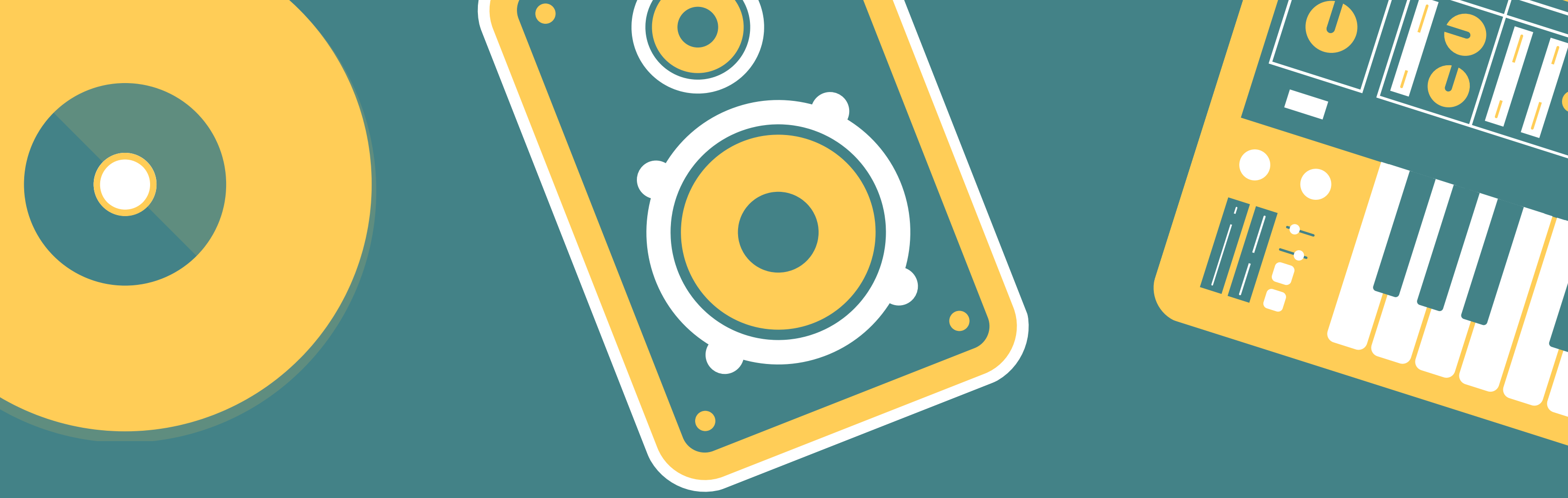
- Demographic analysis of customers (age, gender, location).
- Customer purchasing behavior (frequency of purchases, average purchase value).
- Identification of high-value customers and their buying patterns.
- Customer retention analysis (repeat customers, churn rate).

3. GENRE & ARTIST ANALYSIS

- Popularity of music genres based on sales and customer preferences.
- Analysis of top-selling artists and their impact on sales.
- Correlation between genres, artists, and customer demographics.

4. OPERATIONAL METRICS

- Average revenue per transaction.
- Inventory management metrics (e.g., track popularity, album inventory turnover).
- Analysis of discounts and their impact on sales.



QUESTIONS (SET-1)

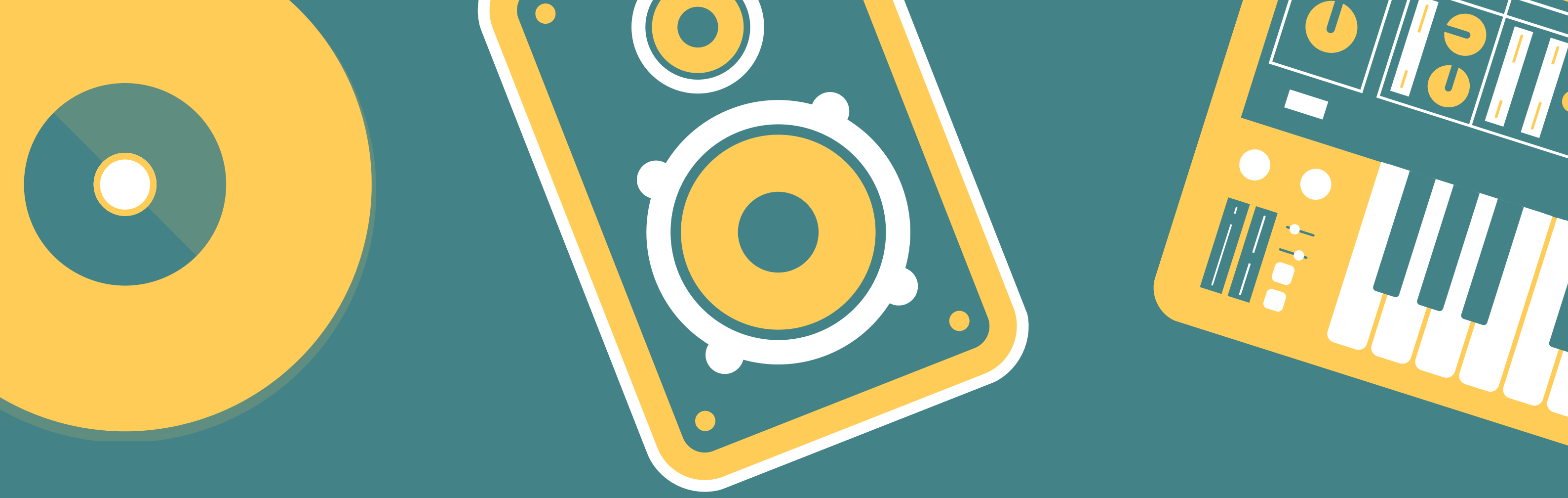
Q.1 Who is the senior most employee based on job title?

Q.2 Which countries have the most invoices?

Q.3 What are top 3 values of total invoice

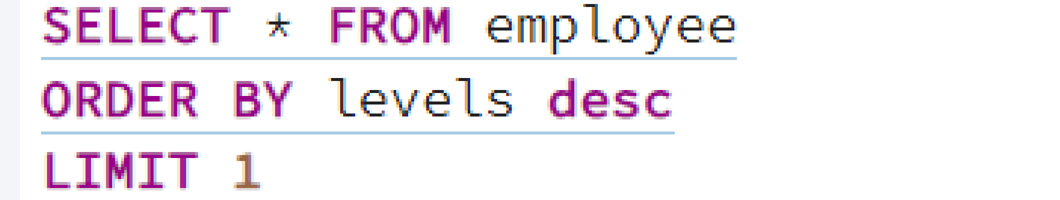
Q.4 Which city has the best customers? We would like to throw a promotional Music Festival in the city we made the most money. Write a query that returns one city that has the highest sum of invoice totals. Return both the city name & sum of all invoice totals.

Q.5 Who is the best customer? The customer who has spent the most money will be declared the best customer. Write a query that returns the person who has spent the most money.



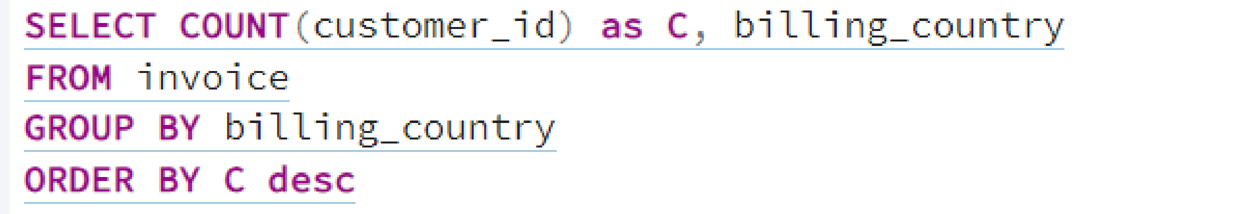
ANSWERS (SET-1)

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



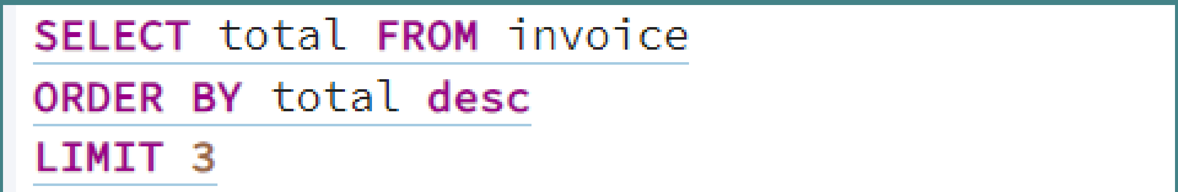
```
SELECT * FROM employee
ORDER BY levels desc
LIMIT 1
```

Q.2 Which countries have the most invoices?

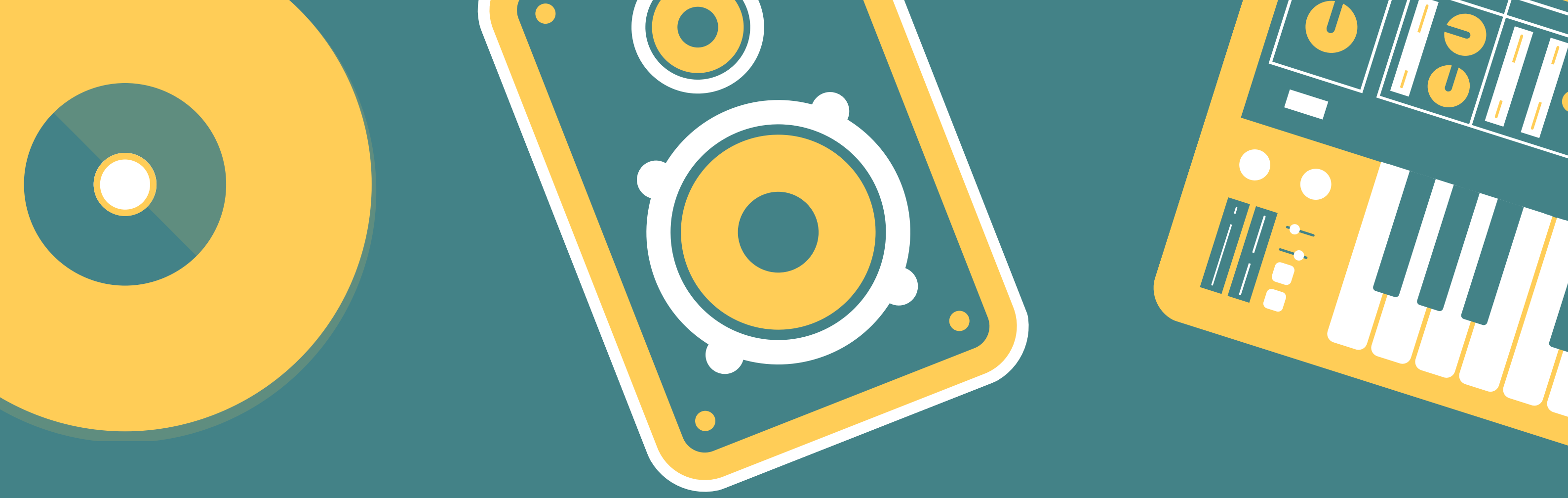


```
SELECT COUNT(customer_id) as C, billing_country
FROM invoice
GROUP BY billing_country
ORDER BY C desc
```

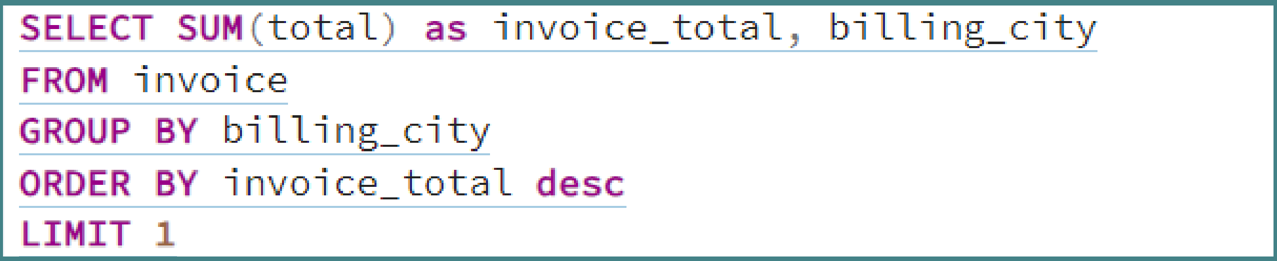
Q.3 What are top 3 values of total invoice?



```
SELECT total FROM invoice
ORDER BY total desc
LIMIT 3
```

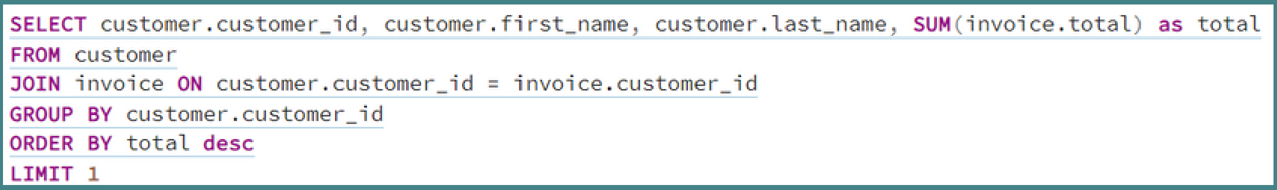


Q.4 Which city has the best customers? We would like to throw a promotional Music Festival in the city we made the most money. Write a query that returns one city that has the highest sum of invoice totals. Return both the city name & sum of all invoice totals.

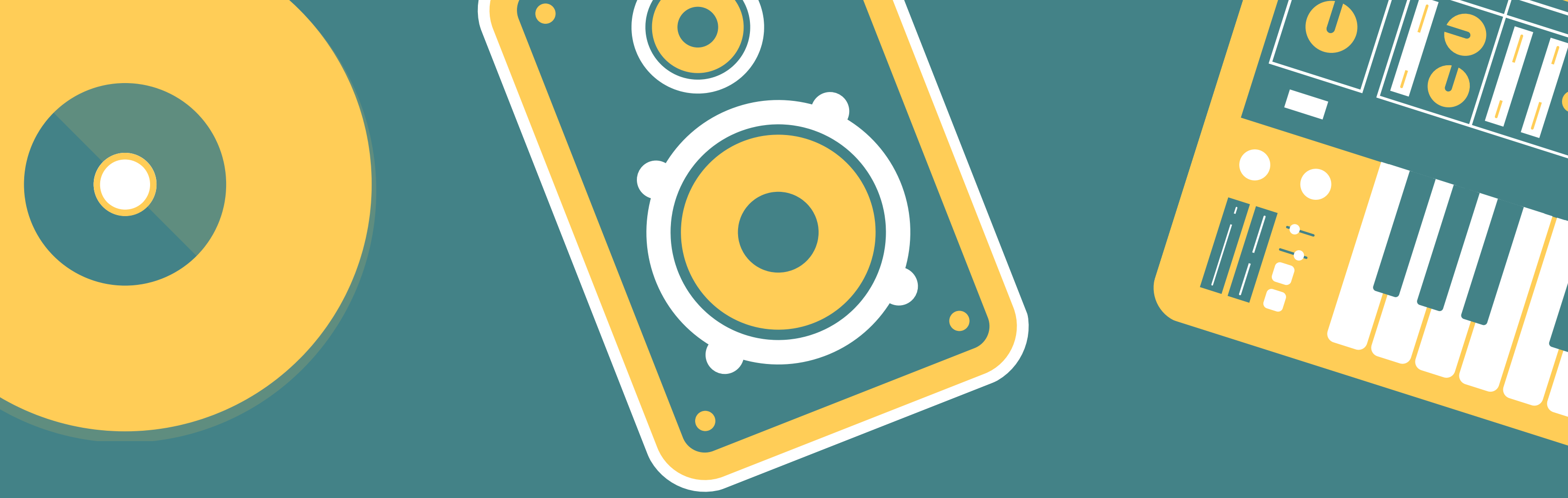


```
SELECT SUM(total) as invoice_total, billing_city
FROM invoice
GROUP BY billing_city
ORDER BY invoice_total desc
LIMIT 1
```

Q.5 Who is the best customer? The customer who has spent the most money will be declared the best customer. Write a query that returns the person who has spent the most money.



```
SELECT customer.customer_id, customer.first_name, customer.last_name, SUM(invoice.total) as total
FROM customer
JOIN invoice ON customer.customer_id = invoice.customer_id
GROUP BY customer.customer_id
ORDER BY total desc
LIMIT 1
```

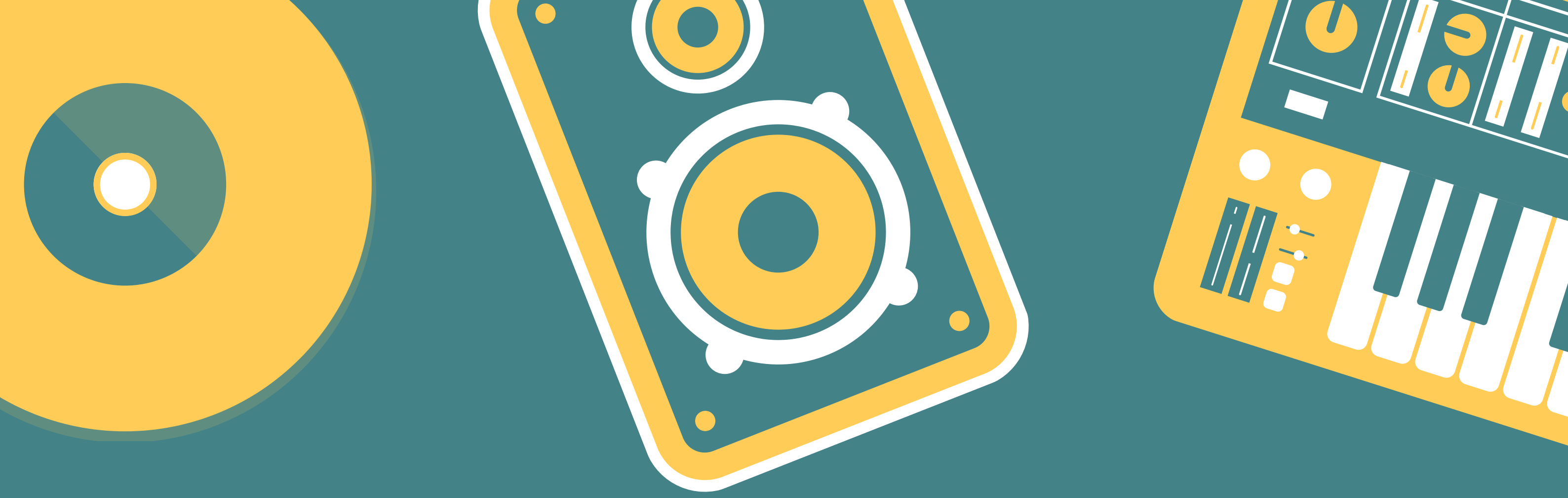


QUESTIONS (SET-2)

Q.1 Write query to return the email, first name, last name and genre of all rock music listeners. Return your list ordered alphabetically by email starting with ‘A’.

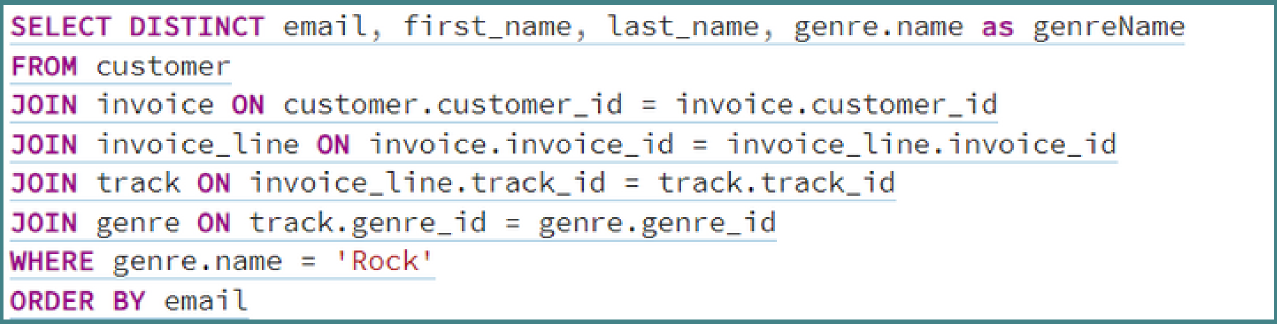
Q.2 Let's invite the artists who have written the most rock music in our dataset. Write a query that returns the artist name and total track count of the top 10 rock bands.

Q.3 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.



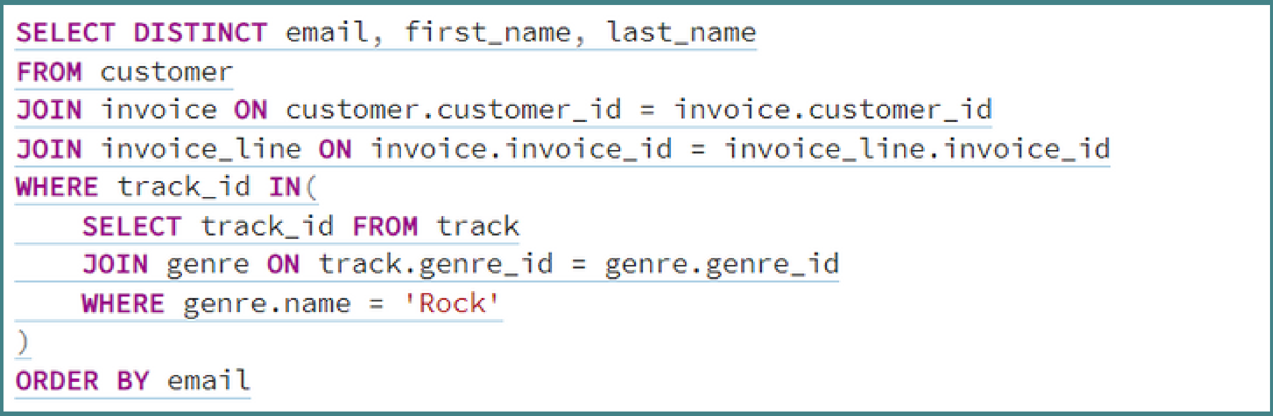
ANSWERS (SET-2)

Q.1 Write query to return the email, first name, last name and genre of all rock music listeners. Return your list ordered alphabetically by email starting with ‘A’.

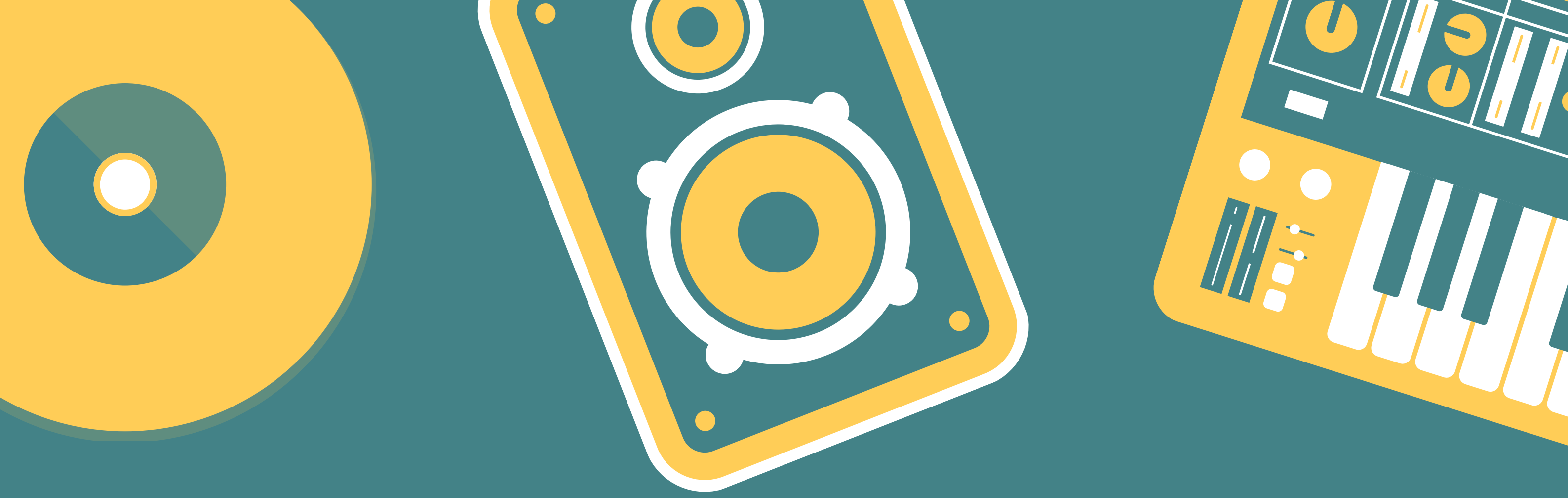


```sql
SELECT DISTINCT email, first_name, last_name, genre.name as genreName
FROM customer
JOIN invoice ON customer.customer_id = invoice.customer_id
JOIN invoice_line ON invoice.invoice_id = invoice_line.invoice_id
JOIN track ON invoice_line.track_id = track.track_id
JOIN genre ON track.genre_id = genre.genre_id
WHERE genre.name = 'Rock'
ORDER BY email
```

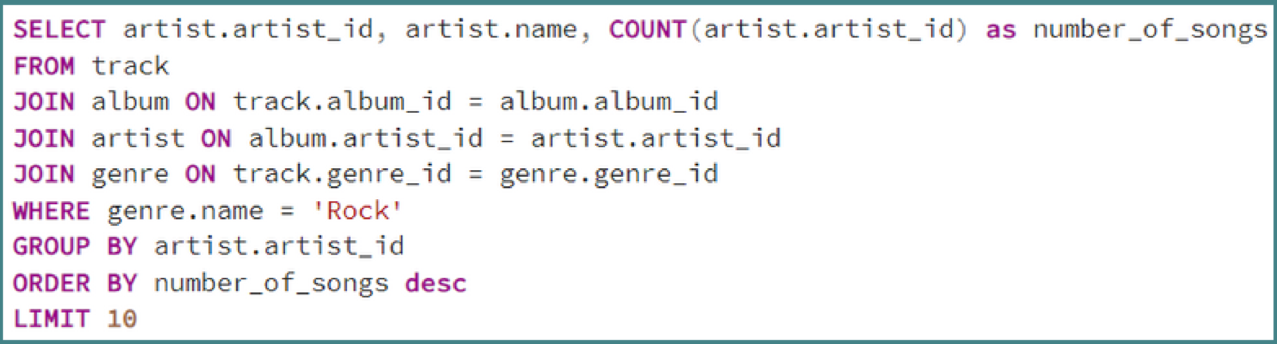
OR



```sql
SELECT DISTINCT email, first_name, last_name
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 = 'Rock'
)
ORDER BY email
```

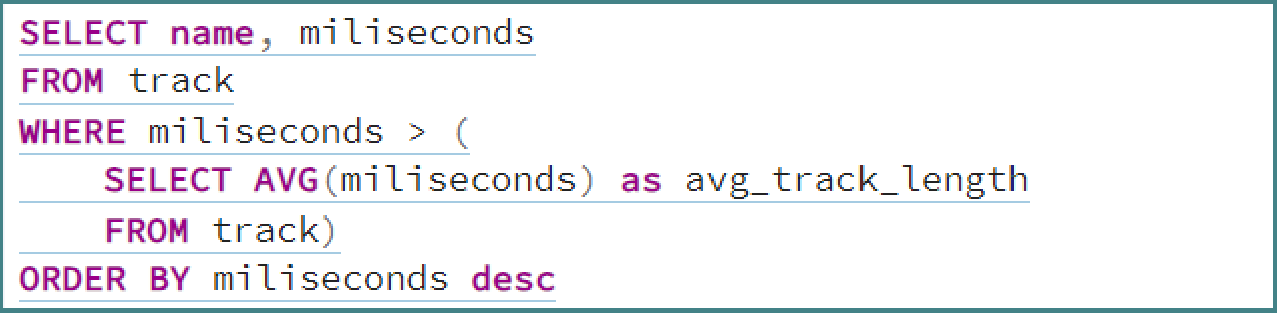


Q.2 Let's invite the artists who have written the most rock music in our dataset. Write a query that returns the artist name and total track count of the top 10 rock bands.

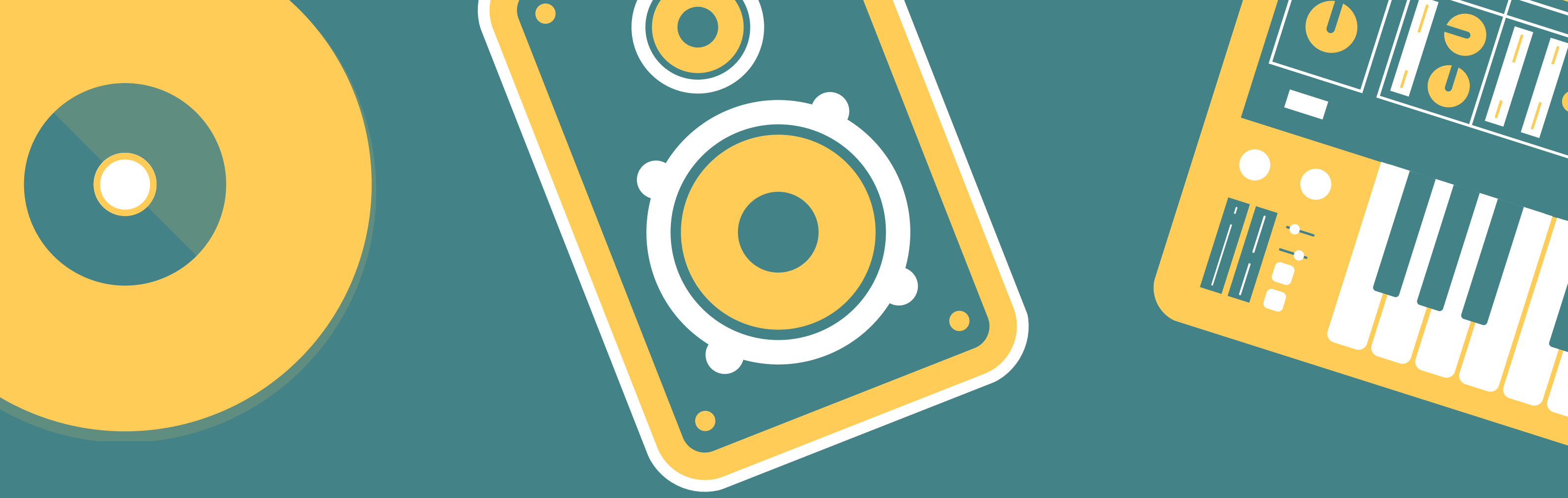


```
SELECT artist.artist_id, artist.name, COUNT(artist.artist_id) as number_of_songs
FROM track
JOIN album ON track.album_id = album.album_id
JOIN artist ON album.artist_id = artist.artist_id
JOIN genre ON track.genre_id = genre.genre_id
WHERE genre.name = 'Rock'
GROUP BY artist.artist_id
ORDER BY number_of_songs desc
LIMIT 10
```

Q.3 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.



```
SELECT name, miliseconds
FROM track
WHERE miliseconds > (
    SELECT AVG(miliseconds) as avg_track_length
    FROM track)
ORDER BY miliseconds desc
```

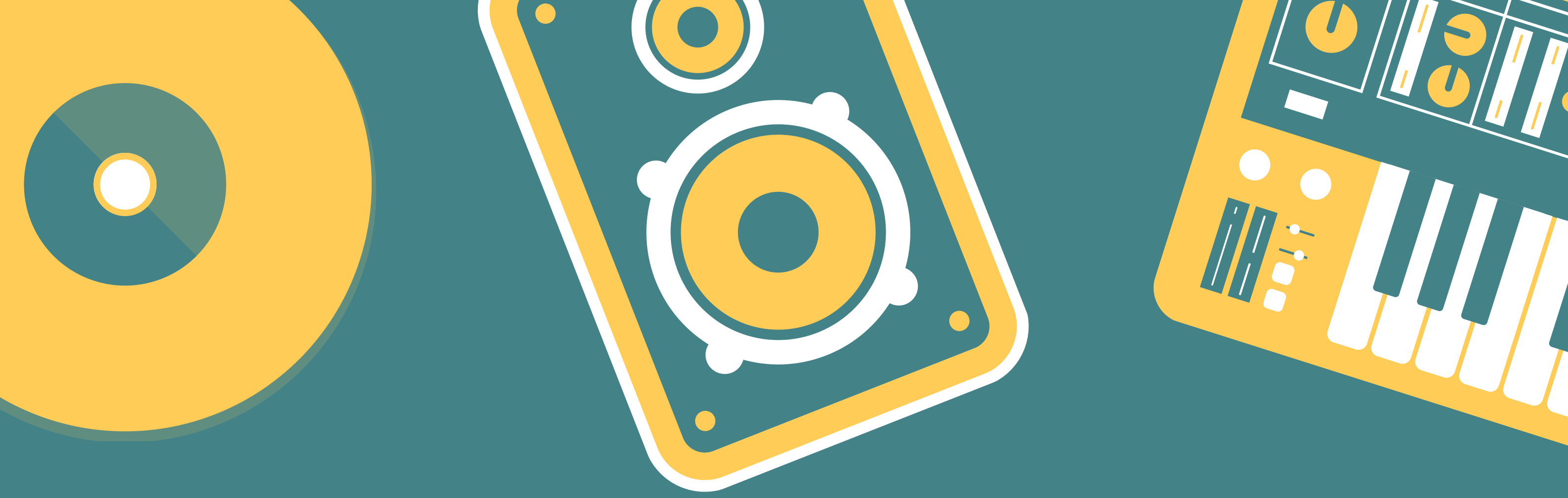


QUESTIONS (SET-3)

Q.1 Find how much amount is spent by each customer on artists? Write a query to return customer name, artist name and total amount spent.

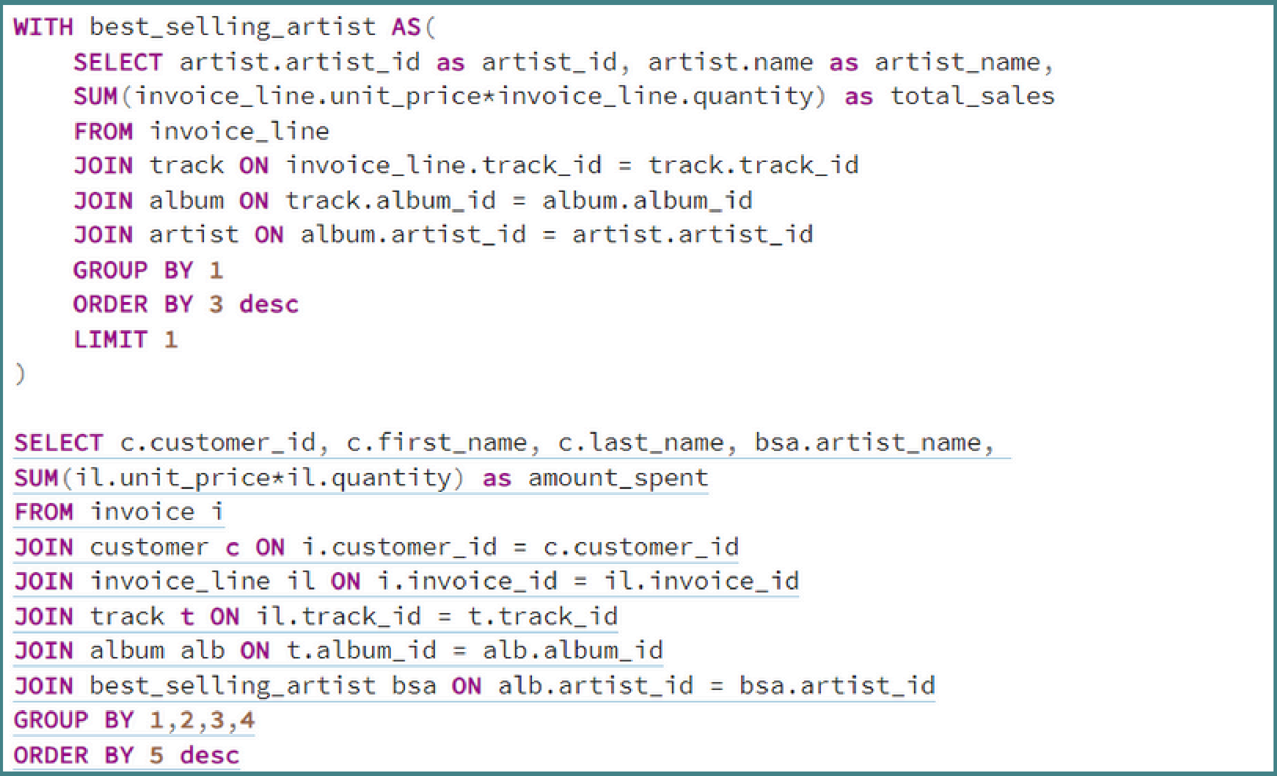
Q.2 We want to find out the most popular music genre for each country. We determine the most popular genre as the genre with the highest amount of purchases. Write a query that returns each country along with the top genre. For countries where the maximum number of purchases is same for two or more genres return all genres.

Q.3 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 the top customer and how much they spent. For countries where the top amount spent is same, provide all customers who spent this amount.



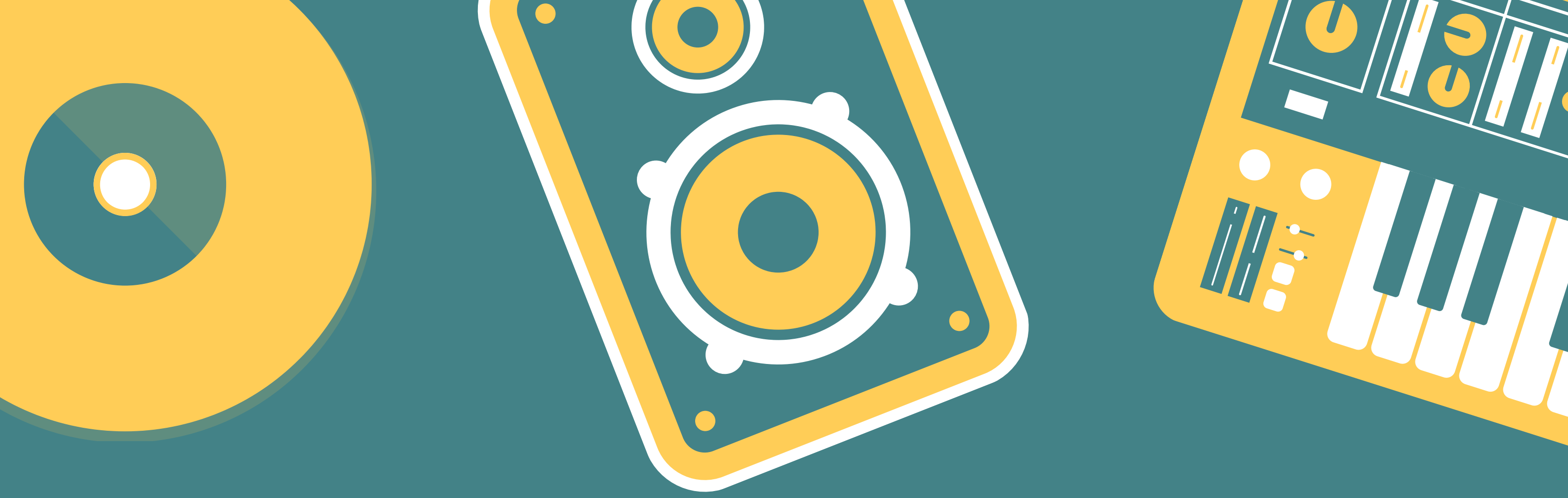
ANSWERS (SET-3)

Q.1 Find how much amount is spent by each customer on artists? Write a query to return customer name, artist name and total amount spent.

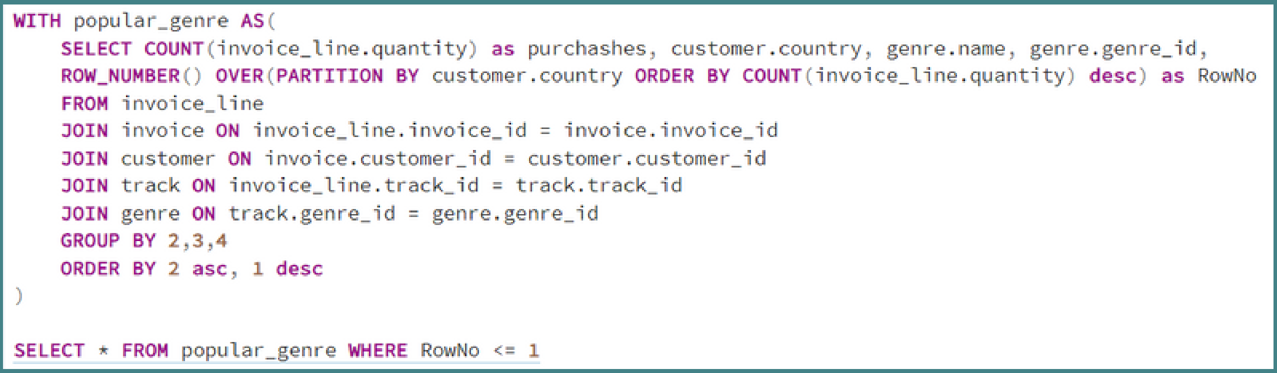


```
WITH best_selling_artist AS(
    SELECT artist.artist_id as artist_id, artist.name as artist_name,
    SUM(invoice_line.unit_price*invoice_line.quantity) as total_sales
    FROM invoice_line
    JOIN track ON invoice_line.track_id = track.track_id
    JOIN album ON track.album_id = album.album_id
    JOIN artist ON album.artist_id = artist.artist_id
    GROUP BY 1
    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 i.customer_id = c.customer_id
JOIN invoice_line il ON i.invoice_id = il.invoice_id
JOIN track t ON il.track_id = t.track_id
JOIN album alb ON t.album_id = alb.album_id
JOIN best_selling_artist bsa ON alb.artist_id = bsa.artist_id
GROUP BY 1,2,3,4
ORDER BY 5 desc
```



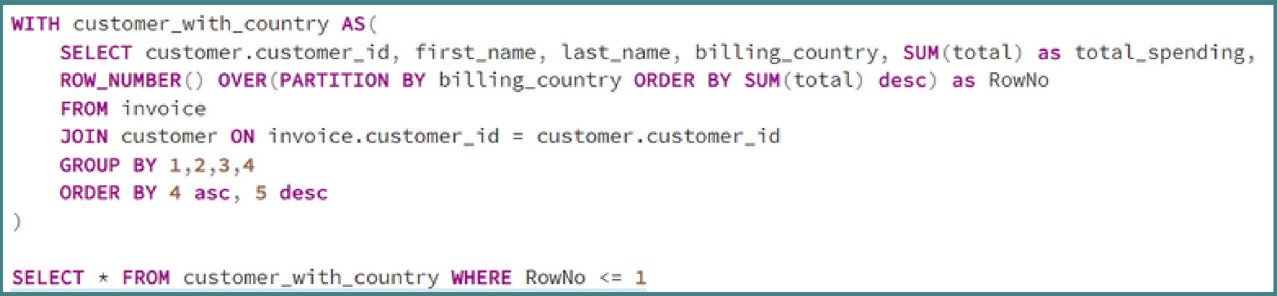
Q.2 We want to find out the most popular music genre for each country. We determine the most popular genre as the genre with the highest amount of purchases. Write a query that returns each country along with the top genre. For countries where the maximum number of purchases is same for two or more genres return all genres.



```
WITH popular_genre AS(
    SELECT COUNT(invoice_line.quantity) as purchashes, customer.country, genre.name, genre.genre_id,
    ROW_NUMBER() OVER(PARTITION BY customer.country ORDER BY COUNT(invoice_line.quantity) desc) as RowNo
    FROM invoice_line
    JOIN invoice ON invoice_line.invoice_id = invoice.invoice_id
    JOIN customer ON invoice.customer_id = customer.customer_id
    JOIN track ON invoice_line.track_id = track.track_id
    JOIN genre ON track.genre_id = genre.genre_id
    GROUP BY 2,3,4
    ORDER BY 2 asc, 1 desc
)

SELECT * FROM popular_genre WHERE RowNo <= 1
```

Q.3 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 the top customer and how much they spent. For countries where the top amount spent is same, provide all customers who spent this amount.



```
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 invoice.customer_id = customer.customer_id
    GROUP BY 1,2,3,4
    ORDER BY 4 asc, 5 desc
)

SELECT * FROM customer_with_country WHERE RowNo <= 1
```