**QUERY WITH RESULTS**

**QUESTION SET 2 – EASY**

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

SELECT title, last\_name, first\_name FROM employee ORDER BY levels DESC LIMIT 1;

**Q2: Which countries have the most Invoices?**

SELECT COUNT(\*) AS c, billing\_country FROM invoice

GROUP BY billing\_country ORDER BY c DESC;

**Q3: What are top 3 values of total invoice?**

SELECT total FROM invoice ORDER BY total DESC;

**Q4: 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 billing\_city, SUM(total) AS InvoiceTotal FROM invoice

GROUP BY billing\_city ORDER BY InvoiceTotal DESC LIMIT 1;

**Q5: 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 customer with customer\_id, customer name who has spent the most money.**

SELECT c.customer\_id, concat(c.first\_name,' ',c.last\_name) as customer\_name, SUM(i.total) AS total\_spending FROM customer c JOIN invoice i ON c.customer\_id = i.customer\_id

GROUP BY c.customer\_id,customer\_name

ORDER BY total\_spending DESC LIMIT 1;

**QUESTION SET 2 – MODERATE**

**Q1: 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.**

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

WHERE track\_id IN(

SELECT track\_id FROM track t

JOIN genre g ON t.genre\_id = g.genre\_id WHERE g.name LIKE 'Rock')

ORDER BY email;

**Q2: 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 album.album\_id = track.album\_id

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

JOIN genre ON genre.genre\_id = track.genre\_id WHERE genre.name LIKE 'Rock'

GROUP BY artist.artist\_id ORDER BY number\_of\_songs DESC LIMIT 10;

**Q3: 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;

**QUESTION SET 3 – ADVANCE**

**Q1: Find how much amount spent by each customer on artists? Write a query to return customer name, artist name and total spent.**

**Steps to Solve: First, find which artist has earned the most according to the InvoiceLines. Now use this artist to find which customer spent the most on this artist. For this query, you will need to use the Invoice, InvoiceLine, Track, Customer, album, and Artist tables. Note, this one is tricky because the Total spent in the Invoice table might not be on a single product,**

**so you need to use the InvoiceLine table to find out how many of each product was purchased, and then multiply this by the price**

**for each artist.**

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 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;

**Q2: 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.**

**Steps to Solve: There are two parts in question- first most popular music genre and second need data at country level.**

WITH popular\_genre AS (

SELECT COUNT(invoice\_line.quantity) AS purchases, 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.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 2,3,4

ORDER BY 2 ASC, 1 DESC )

SELECT \* FROM popular\_genre WHERE RowNo <= 1 ;

**Q3: 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 shared, provide all customers who spent this amount. Steps to Solve: Similar to the above question. There are two parts in question-**

**first find the most spent on music for each country and second filter the data for respective customers.**

WITH Customter\_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 customer.customer\_id = invoice.customer\_id GROUP BY 1,2,3,4

ORDER BY 4 ASC,5 DESC)

SELECT \* FROM Customter\_with\_country WHERE RowNo <= 1;