SQL PROJECT - MUSIC STORE DATA ANALYSIS

Question Set 1 - Easy

1. Identify the most senior employee based on their job title.

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

2. Determine the countries with the highest number of invoices and provide a count for each.

```
SELECT COUNT(*) AS invoice_count, billing_country
FROM invoice
GROUP BY billing_country
ORDER BY invoice_count DESC;
```

3. Retrieve the top three values of total invoices.

```
SELECT total
FROM invoice
ORDER BY total DESC
LIMIT 3;
```

4. Identify the city with the highest sum of invoice totals, as we plan a promotional Music Festival.

```
SELECT billing_city, SUM(total) AS total_invoice_sum
FROM invoice
GROUP BY billing_city
ORDER BY total_invoice_sum DESC
LIMIT 1;
```

5. Find the customer who has spent the most money.

```
SELECT customer.customer_id, customer.first_name, customer.last_name, SUM(invoice.total)
AS total_spent
FROM customer
JOIN invoice ON customer.customer_id = invoice.customer_id
GROUP BY customer.customer_id
ORDER BY total_spent DESC
LIMIT 1;
```

6. Determine the average total for all invoices.

```
SELECT AVG(total) AS average_invoice_total
FROM invoice;
```

7. Which genre has the highest number of tracks in the database?

```
SELECT genre.name, COUNT(track.track_id) AS track_count
FROM genre
JOIN track ON genre.genre_id = track.genre_id
GROUP BY genre.name
ORDER BY track_count DESC
LIMIT 1;
```

Question Set 2 – Moderate

1. Retrieve the email, first name, and last name of all Rock Music listeners, ordered alphabetically by email.

```
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 LIKE 'Rock'
)
ORDER BY email;
```

2. Identify the top 10 rock bands based on the total number of tracks they have written.

```
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;
```

3. Retrieve track names and milliseconds for tracks longer than the average song length.

```
 SELECT name, milliseconds
 FROM track
 WHERE milliseconds > (
    SELECT AVG(milliseconds) AS avg_track_length
    FROM track
 )
 ORDER BY milliseconds DESC;
```

4. List all artists with albums containing more than 20 tracks.

```
 SELECT artist.name, COUNT(track.track_id) AS track_count
 FROM artist
JOIN album ON artist.artist_id = album.artist_id
JOIN track ON album.album_id = track.album_id
GROUP BY artist.name
HAVING COUNT(track.track_id) > 20
ORDER BY track_count DESC;
```

5. Identify customers who have made purchases in every genre available. Return customer details such as customer_id, first_name, and last_name.

```
WITH GenreCount AS (
   SELECT
      c.customer_id,
      COUNT(DISTINCT g.genre_id) AS total_genres
   FROM
      customer c
   CROSS JOIN
      genre g
   LEFT JOIN
      track t ON g.genre_id = t.genre_id
   LEFT JOIN
      invoice_line il ON t.track_id = il.track_id
   GROUP BY
      c.customer_id
)

SELECT
   c.customer_id,
   c.first_name,
   c.last_name
FROM
   customer c
JOIN
   GenreCount gc ON c.customer_id = gc.customer_id
WHERE  gc.total_genres = (SELECT COUNT(*) FROM genre);
```

Question Set 3 – Advanced

1. Determine the amount spent by each customer on artists, returning customer name, artist name, and total 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 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;
```

2. Determine the most popular music Genre for each country, considering the genre with the highest number of purchases.

-- Method 1: Using CTE

```
  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;
```

-- Method 2: Using Recursive

```
  WITH RECURSIVE sales_per_country AS(
     SELECT
        COUNT(*) AS purchases_per_genre,
        customer.country,
        genre.name,
        genre.genre_id
     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
),
max_genre_per_country AS (
    SELECT
        MAX(purchases_per_genre) AS max_genre_number,
        country
    FROM
        sales_per_country
    GROUP BY
        2
    ORDER BY
        2
)
```

3. Identify the top-spending customer for each country and their total expenditure.

-- Method 1: Using CTE

```
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 customer.customer_id = invoice.customer_id
    GROUP BY
        1, 2, 3, 4
    ORDER BY
        4 ASC, 5 DESC
)
SELECT
    customer_id,
    first_name,
    last_name,
    billing_country,
    total_spending
FROM
    Customer_with_country
WHERE
    RowNo <= 1;
```

-- Method 2: Using Recursive

```
WITH RECURSIVE customter_with_country AS (
    SELECT
        customer.customer_id,
        first_name,
        last_name,
        billing_country,
        SUM(total) AS total_spending
    FROM
        invoice
    JOIN
        customer ON customer.customer_id = invoice.customer_id
    GROUP BY
        1, 2, 3, 4
    ORDER BY
        2, 3 DESC
),
country_max_spending AS (
    SELECT
        billing_country,
        MAX(total_spending) AS max_spending
    FROM
        customter_with_country
    GROUP BY
        billing_country
)
SELECT
    cc.billing_country,
    cc.total_spending,
    cc.first_name,
    cc.last_name,
    cc.customer_id
FROM
    customter_with_country cc
JOIN
    country_max_spending ms ON cc.billing_country = ms.billing_country
WHERE
    cc.total_spending = ms.max_spending
ORDER BY
    1;
```

4. List the top 5 customers who have made the highest total purchases.

```
SELECT
    customer.customer_id,
    customer.first_name,
    customer.last_name,
    SUM(invoice.total) AS total_purchases
FROM
    customer
JOIN
    invoice ON customer.customer_id = invoice.customer_id
GROUP BY
    customer.customer_id,
    customer.first_name,
    customer.last_name
ORDER BY
    total_purchases DESC
LIMIT 5;
```

5. Identify the top 3 countries that have the highest average invoice total. Return the country name and the average invoice total.

```
SELECT
    billing_country,
    AVG(total) AS average_invoice_total
FROM
    invoice
GROUP BY
    billing_country
ORDER BY
    average_invoice_total DESC
LIMIT 3;
```

6. Find the artists who have tracks in multiple genres. Return the artist name and the count of distinct genres.

```
SELECT
    artist.name AS artist_name,
    COUNT(DISTINCT genre.genre_id) AS distinct_genre_count
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
GROUP BY
    artist_name
HAVING
    COUNT(DISTINCT genre.genre_id) > 1
ORDER BY
    distinct_genre_count DESC;
```