MUSIC STORE DATA ANALYSIS PROJECT USING SQL

- **Project Overview:**
  - ✓ Designed and implemented a robust database structure to support a music-related business intelligence system. The project involved structuring data to enable effective analysis of customer behaviours, top-selling artists, and revenue trends across regions.
- **Project Steps:**
  - ✓ **Database Structure Planning**: Defined the schema with key entities like Customer, Invoice, Artist, Genre, and Track, ensuring each table accurately captured relationships and transactional data.
  - ✓ **Schema Design & Normalization**: Applied normalization principles to eliminate redundancy and maintain data integrity, resulting in efficient data storage and retrieval.
  - ✓ **Key Relationships & Indexing**: Established primary and foreign keys to define clear relationships (e.g., linking customers to invoices, artists to tracks). Added indexing on frequently queried columns for performance enhancement.
  - ✓ **Query Optimization & Testing**: Designed complex SQL queries for in-depth analysis, such as top customers by country, revenue by city, and most popular genres. Employed CTEs, window functions, and subqueries to maximize efficiency.
  - ✓ **Validation & Documentation**: Verified accuracy through testing and documented the schema, query processes, and insights for future scalability.
- **SQL Queries:**

BEGINNER LEVEL

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

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

   - ✓ This query retrieves the senior-most employee based on the **levels** column, assuming higher levels indicate more seniority. It orders the employees in **descending** order by levels and limits the result to 1 to get the most senior employee.

2. **Which countries have the most Invoices?**

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

- ✓ **Count** invoices per billing_country. **Grouping** by billing_country and **ordering** by total_count in descending order helps us see which countries have the most invoices.

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

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

- ✓ This query finds the **top 3** invoice totals by selecting and ordering the total field in **descending** order and limiting results to 3.

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;
```

- ✓ This query identifies the city that generated the most revenue from invoices. It **groups** by billing_city and calculates the total invoice amount per city, **ordering** the results by invoice_total in **descending** order.

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

- ✓ This query finds the customer who spent the most. It **joins** customer and invoice tables, **groups** by customer_id, and **sums** total to find each customer's total spending. Then, it **orders** in **descending** order, showing the highest spender.

INTERMEDIATE LEVEL

6. **Write a 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;
```

   ✓ This query lists customers who have purchased **Rock** music. It performs a **subquery** to find track_ids of Rock tracks, then uses that to filter invoices and retrieve **distinct** customer details, **ordered** alphabetically by email.

7. **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 ar. artist_id, ar.name, COUNT (ar. artist_id) AS number_of_songs
FROM track tr
JOIN album al ON al. album_id = tr. album_id
JOIN artist ar ON ar. artist_id = al. artist_id
JOIN genre g ON g. genre_id = tr. genre_id
WHERE g.name LIKE 'Rock'
GROUP BY ar. artist_id
ORDER BY number_of_songs DESC
LIMIT 10;
```

   ✓ This query finds the top **10 Rock** artists by **counting** the number of Rock tracks they've created. It joins track, album, and artist tables, **grouping** by artist_id and **ordering** by number_of_songs in **descending** order.

8. **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, milliseconds
FROM track
WHERE milliseconds > (
        SELECT AVG (milliseconds) AS avg_track_length
```

```
    FROM track)
ORDER BY milliseconds DESC;
```

- ✓ This query lists tracks longer than the average track length. It uses a **subquery** to calculate the **average** length, filters by this average, and **orders** the results by milliseconds in **descending** order.

ADVANCE LEVEL

9. **Find how much amount spent by each customer on artists? Write a query to return 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;
```

- ✓ This query calculates how much each customer has spent on a specific artist (the highest-grossing artist). The best_selling_artist **CTE** finds the top-selling artist. The main query then **sums** up the spending per customer for that artist, **ordering** results by amount_spent.

10. **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.**

```
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;
```

- ✓ This query finds the most popular music genre for each country by using the **ROW_NUMBER ()** function to rank genres within each country by purchases. The CTE popular_genre groups by country and genre, ordering by genre popularity. Only the top genre (RowNo = 1) per country is selected.

11. 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.

```
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 * FROM Customer_with_country WHERE RowNo <= 1;
```

- ✓ This query identifies the top spender in each country by using **ROW_NUMBER ()** to rank customers by total spending per country. The **CTE** Customer_with_country groups by billing_country and orders by spending, and we **select** only the top spender (RowNo = 1) per country.