Music Store Analysis using SQL

Task 1: Who is the senior most employee based on job title?

Query:

```
SELECT * FROM EMPLOYEE

ORDER BY LEVELS DESC

LIMIT 1;
```

Task 2: Which countries have the most Invoices?

Query:

```
SELECT BILLING_COUNTRY, COUNT(*) AS MOST_INVOICE FROM INVOICE

GROUP BY BILLING_COUNTRY

ORDER BY MOST_INVOICE DESC

LIMIT 1;
```

Task 3: What are top 3 values of total invoice?

Query:

```
SELECT * FROM INVOICE

ORDER BY TOTAL DESC

LIMIT 3;
```

Task 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.

Query:

```
SELECT BILLING_CITY, SUM(TOTAL) AS INVOICE_TOTAL FROM INVOICE

GROUP BY BILLING_CITY

ORDER BY INVOICE_TOTAL DESC;
```

Task 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.

Query:

```
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;
```

Task 6: Write query to return the email, first name, last name, & Genre of all Rock Music listeners. Return your list ordered alphabetically by email.

Query:

```
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

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;
```

Task 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.

Query:

```
SELECT ARTIST.NAME, COUNT(*) AS TRACK_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

WHERE GENRE.NAME='Rock'

GROUP BY ARTIST.NAME

ORDER BY TRACK_COUNT DESC LIMIT 5;
```

Task 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.

Query:

```
SELECT NAME, MILLISECONDS FROM TRACK

WHERE MILLISECONDS >

(SELECT AVG(MILLISECONDS) AS AVG_TRACK_LENGTH FROM TRACK)

ORDER BY MILLISECONDS DESC;
```