SQL Queries

**Introduction:**

The power of databases shines through SQL, a pivotal tool for structured data interactions. This document showcases various SQL exercises on the Chinook sample database, a simulated dataset rich with music-related insights. From elementary to complex queries, we dive deep into SQL's capabilities. For clarity, while our SQL commands are present here, the full results are documented separately.

# Problem 1: Querying a Single Table

## 1a: Write a Query that returns the following data from the Customers table: (1)

1. FirstName and Last name concatenated into a single column,
2. The Company Name
3. Their city
4. Their state

Order the resulting data by city.

## ```SELECT

## c.FirstName || c.LastName as "first\_and\_last\_name",

## Company as "customer\_company\_name",

## City as "customer\_city",

## State as "customer\_state"

## FROM customers as c

## ORDER BY 3;```

## 1b: Modify your query to only return customers from Canada or the United States (1)

## ```SELECT

## c.FirstName || c.LastName as "first\_and\_last\_name",

## Company as "customer\_company\_name",

## City as "customer\_city",

## State as "customer\_state"

## FROM customers as c

## WHERE c.Country IN ("Canada", "USA")

## ORDER BY 3;```

## 1c: Modify your query to only return customers in Canada or the United States whose last name starts with the letter M. (1)

## ```SELECT

## c.FirstName || c.LastName as "first\_and\_last\_name",

## Company as "customer\_company\_name",

## City as "customer\_city",

## State as "customer\_state"

## FROM customers as c

## WHERE c.Country IN ("Canada", "USA")

## AND c.LastName LIKE "M%"

## ORDER BY 3;```

# Problem 2: Joining tables

## 2a: Write a query which returns the following information from the Artist, Albums and Tracks tables: (2)

1. Artist Name
2. Album Title
3. Track Names

Order the results by Artists

## ```SELECT

## ar.name as "artist\_name",

## al.Title as "album\_title",

## tr.name as "track\_name"

## from artists ar

## LEFT JOIN albums al ON ar.ArtistId = al.artistid

## LEFT JOIN tracks tr ON al.albumid = tr.albumid

## ORDER BY 1;```

## 2b: Modify the previous query so it only returns tracks that have the word “dancing” somewhere in the track name. (1)

## ```SELECT

## ar.name as "artist\_name",

## al.Title as "album\_title",

## tr.name as "track\_name"

## from artists ar

## LEFT JOIN albums al ON ar.ArtistId = al.artistid

## LEFT JOIN tracks tr ON al.albumid = tr.albumid

## WHERE tr.name LIKE "%dancing%"

## ORDER BY 1;```

## 2c: You have been asked to create a org chart for the company. Create a query that returns two columns one with the employee’s first and last name and the second with their managers first and last name. (2)

Example:

Employee Manager

Nancy Edwards | Andrew Adams

## ```SELECT

## e.firstname || " " || e.lastname as "employee\_name",

## m.firstname || " " || m.lastname as "manager\_name"

## FROM employees e

## INNER JOIN employees m on e.reportsto = m.employeeid;```

## 2d: When you review the list you notice that the General Manager, Andrew Adams, is not included on the list. Modify your query from 2c: to include Andrew so the record looks as follows: “Andrew Adams reports to himself” (1)

## ```SELECT

## e.firstname || " " || e.lastname as "employee\_name",

## COALESCE(m.firstname || " " || m.lastname,

## e.firstname || " " || e.lastname || " reports to himself") as "manager\_name"

## FROM employees e

## LEFT join employees m on e.reportsto = m.employeeid;```

# Problem 3: Aggregation

## 3a: Create a report that that Joins the Albums to Tracks table and returns the title of each album and the number of tracks it contains. (2)

## ```SELECT

## a.albumid as "album\_id",

## a.title as "album\_title",

## COUNT(t.TrackId) as "track\_count"

## FROM albums a

## INNER JOIN tracks t on a.AlbumId = t.albumid

## GROUP BY 1,2```

## 3b: Modify your query from 3a so that the report only shows albums that have more than ten tracks. (1)

## ```SELECT

## a.albumid as "album\_id",

## a.title as "album\_title",

## COUNT(t.TrackId) as "track\_count"

## FROM albums a

## INNER JOIN tracks t ON a.AlbumId = t.albumid

## GROUP BY 1,2

## HAVING "track\_count" > 10```

**Conclusion:**

Our exploration with the Chinook database reaffirms SQL's versatility in data querying and analysis. This exercise, coupled with the separate results, paints a comprehensive picture of data's transformative role in our digital era.