***Analysis with SQL***

**Project Overview**

This project focuses on creating and managing a database for a music record company, with the goal of performing various analyses on bands, albums, and songs. Using MySQL and SQL queries, we create tables, populate data, and perform data analysis to extract insights such as identifying the oldest album, longest song, and band-related statistics.

**Steps for the Project**

**1. Database and Schema Creation**

* A new database named record\_company was created to store information about bands, albums, and songs.
* The schema includes the following tables:
  + **Bands**: Stores the details of music bands.
  + **Albums**: Contains information about albums, with a foreign key relationship to the Bands table.
  + **Songs**: Contains songs data, linked to albums via a foreign key.

Example for table creation:

CREATE TABLE Songs (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255),

length INT,

album\_id INT,

FOREIGN KEY (album\_id) REFERENCES Albums(id)

);

**2. Data Population**

* Data was inserted using predefined SQL scripts:
  + schema.sql: Contains the commands to create the necessary tables.
  + data.sql: Populates the tables with initial data for bands, albums, and songs.
* After creating the tables, the data was loaded into the database using:

sql

SOURCE /path/to/data.sql;

**3. SQL Queries for Data Analysis**

Various SQL queries were executed to extract insights from the data:

* **Select all band names**:

sql

SELECT DISTINCT Bands.name

FROM Bands

JOIN Albums ON Bands.id = Albums.band\_id

JOIN Songs ON Albums.id = Songs.album\_id;

* **Find the oldest album**:

sql

SELECT name FROM Albums ORDER BY release\_year ASC LIMIT 1;

* **Get bands with and without albums**:
  + Bands **with albums**:

sql

SELECT DISTINCT Bands.name

FROM Bands

JOIN Albums ON Bands.id = Albums.band\_id;

* + Bands **without albums**:

sql

SELECT name

FROM Bands

WHERE id NOT IN (SELECT band\_id FROM Albums);

* **Find the longest album (sum of song lengths)**:

sql

SELECT Albums.name

FROM Albums

JOIN Songs ON Albums.id = Songs.album\_id

GROUP BY Albums.id

ORDER BY SUM(Songs.length) DESC

LIMIT 1;

* **Insert a new record** for a favorite band and one of their albums:

sql

INSERT INTO Bands (name) VALUES ('My Favorite Band');

INSERT INTO Albums (name, release\_year, band\_id)

VALUES ('My Favorite Album', 2024, LAST\_INSERT\_ID());

* **Calculate the average length of all songs**:

sql

SELECT AVG(length) AS avg\_song\_length FROM Songs;

* **Find the longest song of each album**:

sql

SELECT Albums.name AS album\_name, Songs.name AS song\_name, MAX(Songs.length) AS longest\_song

FROM Albums

JOIN Songs ON Albums.id = Songs.album\_id

GROUP BY Albums.id;

* **Count the number of songs for each band**:

sql

SELECT Bands.name, COUNT(Songs.id) AS song\_count

FROM Bands

JOIN Albums ON Bands.id = Albums.band\_id

JOIN Songs ON Albums.id = Songs.album\_id

GROUP BY Bands.id;

**4. Database Management**

* Foreign key relationships were carefully maintained to ensure data integrity.
* Songs were linked to albums, and albums were linked to bands, allowing for meaningful data analysis.
* Records were inserted, updated, and deleted as needed to complete the assignment requirements.

**How to Run This Project**

1. **Create the Database**:
   * Run the schema.sql file in MySQL to create the database and tables.
2. **Insert Data**:
   * Run the data.sql file to populate the tables with data.
3. **Run Queries**:
   * Execute the SQL queries provided in this README or the analysis.sql file to perform data analysis.