

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | PROJECT REPORT | | | | |  | |
|  |  | | | | | | |  |
|  | | | |  |  | | | |
|  | | | | Nidhi Pujara |  | | | |
|  | | | | 22/08/2025—SQL Project—Professor Chintan Patel |  | | | |
|  | | |  | | |  | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  |  | | |  |
|  | INTRODUCTION | | | | | | |  |
|  |  | | |  |  | | |  |
|  |  | |  | | |  | |  |
|  |  |  | The purpose of this project is to design and implement a **Songs Database** using SQL. The database is built to store essential information about songs, artists, albums, genres, and other related details. It provides an efficient way to query and analyze the dataset using SQL queries.  The dataset contains details such as **song title, artist information, year of release, duration, and location**. SQL queries are applied to retrieve meaningful insights, perform aggregations, and demonstrate concepts like subqueries, joins, grouping, and ordering. | | |  |  |  |
| Decorative | | | | |
|  |  |  |  |
|  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Decorative | |  |  | | |  |  | |
|  | | OBJECTIVES | | | | |  | |
|  | |  |  | | |  |  | |
|  | The main objectives of this project are:   * To design and implement a structured database for storing song and artist details. * To practice SQL concepts such as **DDL, DML, aggregate functions, subqueries, joins, case statements, group by, having, and order by**. * To extract useful insights such as most popular artists, longest songs, and distribution of songs over years. * To showcase real-world applications of relational databases in music management systems. | | | | | | |  |
|  | SCOPE | | |  |  | | |  |
|  | **In Scope**:   * Storage of song details (title, duration, year). * Storage of artist details (name, location, coordinates). * SQL queries for analysis and retrieval of insights. | | |  | * **Out of Scope**:   + Actual audio storage or streaming.   + Integration with music recommendation systems. | | |  |

|  |  |
| --- | --- |
|  | SONGS DATABASE |

|  |  |  |  |
| --- | --- | --- | --- |
|  | DATABASE DESIGN **Entities and Attributes**   * **Song**: song\_id, title, duration, year, num\_songs, artist\_id * **Artist**: artist\_id, artist\_name, artist\_location, artist\_latitude, artist\_longitude   **Relationships**   * Each **song** is linked to one **artist** (via artist\_id). * One artist can have multiple songs. |  | |
|  |  | |  |
|  |  |  | |

|  |  |
| --- | --- |
|  | SONGS DATABASE |

|  |  |  |  |
| --- | --- | --- | --- |
|  | QUERIES AND OUTPUTS **PROBLEM STATEMENT 1:**  Retrieve the list of songs released after the year 2002, displaying their title, artist name, and release year, sorted in ascending order of year.  **QUERY 1:**  select title, artist\_name, year from songs where year > 2002 order by year asc;  **OUTPUT:** |  | |
|  |  | |  |
|  |  |  | |

**CONCLUSION:**

This query retrieves all songs released after 2002, along with their titles, artist names, and

release years. The results are arranged in ascending order of year, meaning the earliest

songs after 2002 appear first.

|  |  |
| --- | --- |
|  | SONGS DATABASE |

|  |
| --- |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | QUERIES AND OUTPUTS **PROBLEM STATEMENT 2:**  Find the average, maximum, and minimum duration of all songs in the database.  **QUERY 2:**  select avg(duration) as avg\_duration, max(duration) as max\_duration,  min(duration) as min\_duration from songs.  **OUTPUT:** |  | |
|  |  | |  |
|  |  |  | |

**CONCLUSION:**

This query calculates the overall statistics of song lengths. It shows the average

average duration of songs, the longest song (maximum duration), and the shortest

song (minimum duration). These results give insight into the typical length of songs in

the dataset and help identify extremes in song durations.

|  |  |
| --- | --- |
|  | SONGS DATABASE |

|  |  |  |
| --- | --- | --- |
|  | QUERIES AND OUTPUTS **PROBLEM STATEMENT 3:**  Determine the total number of unique songs by each artist and display the results in descending order of song count.  **QUERY 3:**  select artist\_name, count(distinct(song\_id)) as total\_songs  from songs group by artist\_name order by total\_songs desc.  **OUTPUT:** |  |
|  |  |  |
|  |  |  |

**CONCLUSION:**

This query groups the dataset by artist name and counts the number of distinct songs

for each artist. The results are sorted in descending order, meaning the artist with the

highest number of songs appear first. This helps identify the most prolific artists in

the database.

|  |  |
| --- | --- |
|  | SONGS DATABASE |

|  |
| --- |
|  |

|  |  |  |
| --- | --- | --- |
|  | QUERIES AND OUTPUTS **PROBLEM STATEMENT 4:**  Identify the artists who have contributed more than one unique song in the database.  **QUERY 4:**  select artist\_name, count(distinct(song\_id)) as total\_songs  from songs group by artist\_name having total\_songs>1:  **OUTPUT:** |  |
|  |  |  |
|  |  |  |

**CONCLUSION:**

This query groups the data by artists and counts their distinct songs. The **having**

clause filters the results to only those artists with more than one song. The outcome

highlights the artists with multiple contributions, excluding those who appear only

once in the dataset.

|  |  |
| --- | --- |
|  | SONGS DATABASE |

|  |  |  |
| --- | --- | --- |
|  | QUERIES AND OUTPUTS **PROBLEM STATEMENT 5:**  Classify songs into categories (short’, ‘medium’, and ‘long’) based on their duration and display them in descending order of length.  **QUERY 5:**  select title, artist\_name, duration, case  when duration<120 then "short"  when duration between 120 and 240 then "medium"  else "long"  end as length  from songs order by duration desc;  **OUTPUT:**  **A screenshot of a computer  AI-generated content may be incorrect.** |  |
|  |  |  |
|  |  |  |

**CONCLUSION:**

The results are ordered by duration in descending order, meaning the longest songs

appear first. This classification provides an easy way to analyze the distribution of

songs by their list.

|  |  |
| --- | --- |
|  | SONGS DATABASE |

|  |  |  |
| --- | --- | --- |
|  | QUERIES AND OUTPUTS **PROBLEM STATEMENT 6:**  Retrieve the titles and artist names of songs belonging to artists who have contributed more than one song to the database.  **QUERY 6:**  select title, artist\_name from songs where artist\_id in (  select artist\_id from songs group by artist\_id having count(song\_id)>1);  **OUTPUT:** |  |
|  |  |  |
|  |  |  |

**CONCLUSION:**

This query uses a subquery to first identify all artists with more than one song. Then,

The outer query retrieves the titles and artist names of their songs. The result highlights

all songs from multi-song artists, excluding those from artists with only one entry.

|  |  |  |
| --- | --- | --- |
|  | QUERIES AND OUTPUTS **PROBLEM STATEMENT 7:**  Find the top 5 longest songs in the database, displaying their title, artist name, and duration.  **QUERY 7:**  select title, artist\_name, duration from songs order by duration desc limit 5;  **OUTPUT:** |  |
|  |  |  |
|  |  |  |

**CONCLUSION:**

This query sorts all songs by duration in descending order and selects the first 5 results.

It effectively identifies the five longest songs in the dataset, giving insight into which

tracks have the maximum playtime.

|  |  |
| --- | --- |
|  | SONGS DATABASE |

|  |  |  |
| --- | --- | --- |
|  | QUERIES AND OUTPUTS **PROBLEM STATEMENT 8:**  Retrieve the titles, artist name, and release years of songs by the artist who has the highest number of songs in the database.  **QUERY 8:**  select title, artist\_name, year from songs  where artist\_name = (  select artist\_name  from songs  group by artist\_name  order by count(song\_id) desc  limit 1 );  **OUTPUT:** |  |
|  |  |  |
|  |  |  |

**CONCLUSION:**

This query uses a subquery to first determine the artist with the maximum number

of songs. The outer query then retrieves all songs by that artist, along with their titles

and release years. The result highlights the most prolific artist in the dataset and their

contributions.

|  |  |
| --- | --- |
|  | SONGS DATABASE |

|  |  |  |
| --- | --- | --- |
|  | QUERIES AND OUTPUTS **PROBLEM STATEMENT 9:**  List all distinct artists along with their locations, excluding entries where the artist location is missing or blank.  **QUERY 9:**  select distinct(artist\_name), artist\_location from songs  where artist\_location is not null and trim(artist\_location) <> ‘ ‘;  **OUTPUT:** |  |
|  |  |  |
|  |  |  |

**CONCLUSION:**

This query filters out null and empty location values, ensuring only valid artist locations

are considered. It then retrieves each artist’s name with their location, eliminating

duplicates using **distinct** . The result provides a clean list of artists with known

geographic information.

|  |  |  |
| --- | --- | --- |
|  | SONGS DATABASE |  |

|  |  |  |
| --- | --- | --- |
|  | QUERIES AND OUTPUTS **PROBLEM STATEMENT 10:**  Calculate the average duration of songs for each year (excluding year 0) and display the results in ascending order of year.  **QUERY 10:**  select year, round(avg(duration), 2) as avg\_duration from songs  where year > 0 group by year order by year asc;  **OUTPUT:** |  |
|  |  |  |
|  |  |  |

**CONCLUSION:**

This query groups songs by their release year and computes the average duration for

each year, rounded to two decimal places. The results are sorted in ascending order,

allowing us to observe how average song lengths vary across different years.

|  |  |
| --- | --- |
|  | SONGS DATABASE |

|  |  |  |
| --- | --- | --- |
|  | CONCLUSION The project successfully demonstrated how SQL can be applied to manage and analyze a songs database. Using various queries, we extracted meaningful insights such as identifying the most prolific artist, classifying songs by duration, and analyzing trends across years. FUTURE SCOPE How the project can be extended:   * Add genres, albums, or user playlists. * Build a front-end music management app. * Use machine learning for recommendations. * Expand dataset with streaming stats. |  |
|  |  |  |
|  |  |  |

|  |  |
| --- | --- |
|  | SONGS DATABASE |

