

SQL PROJECT

SPOTIFY DATABASE MANAGEMENT REPOSITORY

NAME: NIKESH VAREKAR

COURSE: MASTER IN DATA SCIENCE AND ANALYTICS WITH AI

ABOUT DATASET:

The Spotify dataset can be understood as a structured collection of data that reflects the various elements and interactions within the Spotify music streaming service. Below is a detailed breakdown of what a typical Spotify dataset might contain. Users Table, Artists Table, Songs Table, Playlists Table, StreamingHistory Table, Album table, playlistsongs table.

AVAILABLE TABLES:

Users Table

To store information about users on the platform.

Artists Table

To store information about music artists.

Albums Table

To store information about music albums.

Songs Table

To store information about individual songs.

Playlists Table

To store information about user-created playlists.

PlaylistSongs Table

To represent the many-to-many relationship between playlists and songs.

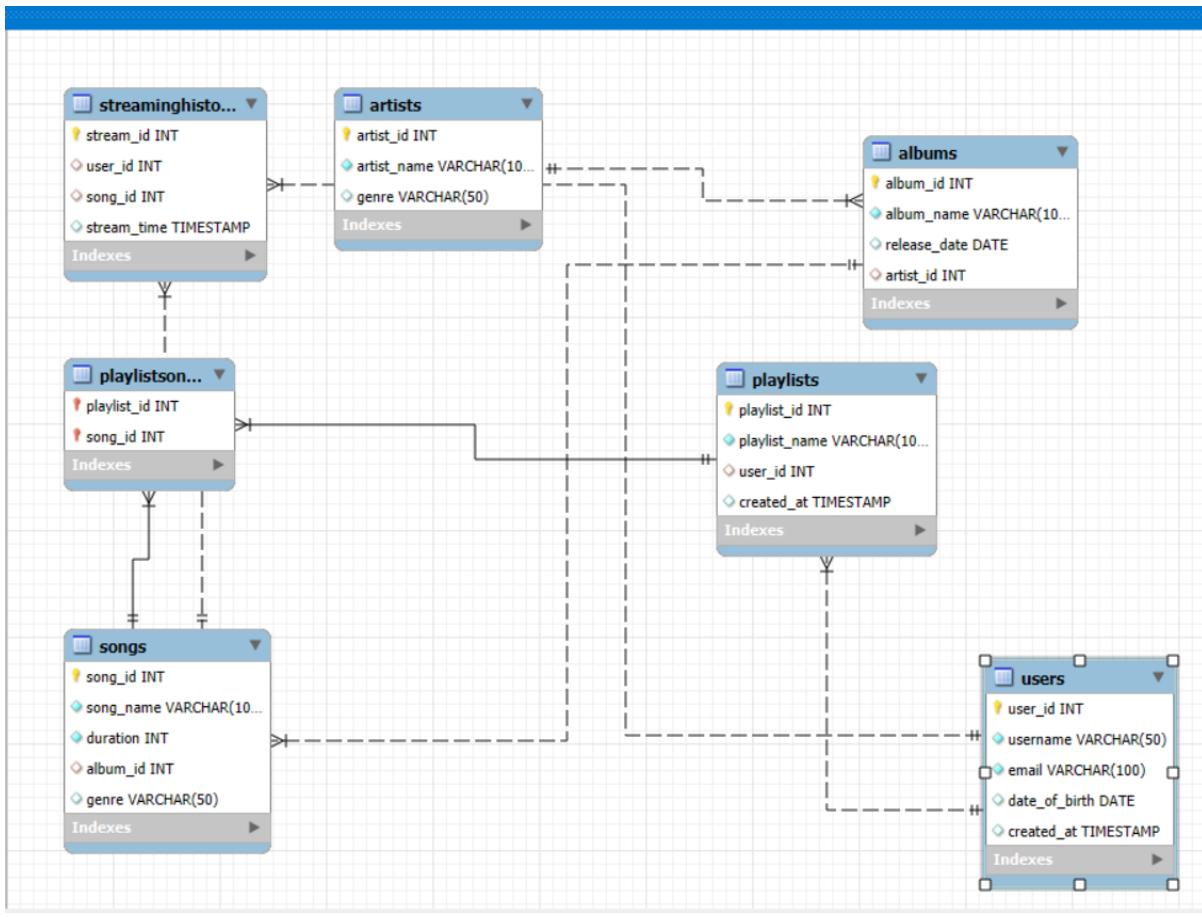
StreamingHistory Table

To log user streaming activities, such as when a user plays a song.

DATASET FROM: SAMPLE DATASET FROM CHATGPT



ER DIAGRAM:



QUERIES:

#1. List All Users

```
select *from users;
```

The screenshot shows the MySQL Workbench interface. The SQL Editor tab contains the following code:

```
280   (10, 5, 13, '2024-08-04 16:20:00'),
281   (11, 6, 10, '2024-08-05 12:20:00'),
282   (12, 6, 11, '2024-08-05 12:35:00'),
283   (13, 7, 7, '2024-08-06 08:35:00'),
284   (14, 7, 14, '2024-08-06 08:50:00'),
285   (15, 8, 12, '2024-08-07 18:15:00');
286 •   select*from StreamingHistory;
287
288 #1. List All Users
289 •   USE SPOTIFY;
290 •   select *from users;
??
1. List All Users
```

The Result Grid shows the following data:

user_id	username	email	date_of_birth	created_at
1	alice_jones	alice.jones@example.com	1985-07-24	2024-08-05 14:33:27
2	bob_smith	bob.smith@example.com	1995-07-24	2024-08-05 14:33:27
3	charlie_brown	charlie.brown@example.com	1990-11-05	2024-08-05 14:33:27
4	diana_prince	diana.prince@example.com	1993-02-17	2024-08-05 14:33:27
5	edward_cullen	edward.cullen@example.com	1993-06-20	2024-08-05 14:33:27
6	fiona_apple	fiona.apple@example.com	1985-12-09	2024-08-05 14:33:27
7	george_clark	george.clark@example.com	1997-04-21	2024-08-05 14:33:27
8	hannah.white	hannah.white@example.com	1991-09-15	2024-08-05 14:33:27

The Output pane shows the following log entries:

#	Time	Action	Message	Duration / Fetch
1	17:38:38	Action	Error Code: 1046. No database selected Select the default DB to be used by double-clicking its name in the SCH...	0.000 sec
2	17:38:55	select *from users LIMIT 0, 1000	0 row(s) affected	0.015 sec
3	17:38:58	USE SPOTIFY	0 row(s) affected	0.000 sec
4	17:39:04	select *from users LIMIT 0, 1000	15 row(s) returned	0.016 sec / 0.000 sec

#2. To get playlists created by user_id = 1

```
SELECT playlist_id, playlist_name, created_at
FROM Playlists
WHERE user_id = 1;
```

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Schemas: hms*, spotify*

```

286 • select * from StreamingHistory;
287
288 #1. List All Users
289 • USE SPOTIFY;
290 • select * from users;
291
292 #2. To get playlists created by user_id = 1
293
294 • SELECT playlist_id, playlist_name, created_at
295 FROM Playlists
296 WHERE user_id = 1;
297

```

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Contents: Result Grid Form Editor

playlist_id	playlist_name	created_at
1	Rock Classics	2024-08-01 10:00:00
2		
3		

Playlists 2 x

Action Output

#	Time	Action	Message	Duration / Fetch
2	17:38:55	USE SPOTIFY	0 row(s) affected	0.015 sec
3	17:38:58	USE SPOTIFY	0 row(s) affected	0.000 sec
4	17:39:04	select * from users LIMIT 0, 1000	15 row(s) returned	0.016 sec / 0.000 sec
5	17:41:21	SELECT playlist_id, playlist_name, created_at FROM Playlists WHERE user_id = 1 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

No object selected

Administration Schemas Information

Object Info Session

Query Completed

83°F Mostly cloudy

#3.To find users born on nov 11, 1990

```

SELECT *
FROM Users
WHERE date_of_birth = '1990-11-05';

```

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Schemas: hms*, spotify*

```

215 FROM Playlists
216 WHERE user_id = 1;

217 #3.To find users born on nov 11, 1990
218 • SELECT *
219 FROM Users
220 WHERE date_of_birth = '1990-11-05';

221 #4.To find users born on or after January 1, 1990
222 • SELECT *
223 FROM Users
224 WHERE date_of_birth >= '1990-01-01';

```

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Contents: Result Grid Form Editor

user_id	username	email	date_of_birth	created_at
3	charlie_brown	charlie.brown@example.com	1990-11-05	2024-08-25 14:33:27
4				

Users 3 x

Action Output

#	Time	Action	Message	Duration / Fetch
3	17:38:58	USE SPOTIFY	0 row(s) affected	0.000 sec
4	17:39:04	select * from users LIMIT 0, 1000	15 row(s) returned	0.016 sec / 0.000 sec
5	17:41:21	SELECT playlist_id, playlist_name, created_at FROM Playlists WHERE user_id = 1 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
6	17:42:37	SELECT * FROM Users WHERE date_of_birth >= '1990-01-01' LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

No object selected

Administration Schemas Information

Object Info Session

Query Completed

83°F Mostly cloudy

#4.To find users born on or after January 1, 1990

```
SELECT *  
FROM Users  
WHERE date_of_birth >= '1990-01-01';
```

The screenshot shows the MySQL Workbench interface. The SQL editor tab contains the following code:

```
218 #3.To find users born on nov 11, 1990  
219 • SELECT *  
220 FROM Users  
221 WHERE date_of_birth = '1990-11-05';  
222  
223 #4.To find users born on or after January 1, 1990  
224 • SELECT *  
225 FROM Users  
226 WHERE date_of_birth >= '1990-01-01';  
227  
228 #5.To find playlists create after August 10, 2024  
229
```

The Results Grid shows the following user data:

user_id	username	email	date_of_birth	created_at
2	bob_smith	bob.smith@example.com	1990-07-24	2024-08-25 14:33:27
3	charlie_brown	charlie.brown@example.com	1990-11-05	2024-08-25 14:33:27
4	diana_prince	diana.prince@example.com	1993-03-17	2024-08-25 14:33:27
5	edward_cullen	edward.cullen@example.com	1993-06-20	2024-08-25 14:33:27
7	george_dark	george.dark@example.com	1997-04-21	2024-08-25 14:33:27
8	hannah_white	hannah.white@example.com	1991-09-15	2024-08-25 14:33:27
10	jessica_green	jessica.green@example.com	1996-08-11	2024-08-25 14:33:27
11	kevin_krausen	kevin.krausen@example.com	1994-10-07	2024-08-25 14:33:27

The Output pane shows the execution log:

Action	Time	Message	Duration / Fetch
select * from users LIMIT 0, 1000	4 17:39:04	15 row(s) returned	0.016 sec / 0.000 sec
SELECT playlist_id, playlist_name, created_at FROM Playlists WHERE user_id = 1 LIMIT 0, 1000	5 17:41:21	1 row(s) returned	0.000 sec / 0.000 sec
SELECT * FROM Users WHERE date_of_birth >= '1990-11-05' LIMIT 0, 1000	6 17:42:37	1 row(s) returned	0.000 sec / 0.000 sec
SELECT * FROM Users WHERE date_of_birth >= '1990-01-01' LIMIT 0, 1000	7 17:43:41	11 row(s) returned	0.000 sec / 0.000 sec

#5.To find playlists create after August 10, 2024

```
SELECT *  
FROM Playlists  
WHERE created_at < '2024-08-10';
```

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: schemas

SCHEMAS: emp_info, hospital_management_system, mydatabase, nikesh, pizza_sales_analysis, spotify, sys

Tables: Views, Stored Procedures, Functions, spotify_database

Administration Schemas Information

No object selected

Object Info Session

Query Completed

Output

Action Output

```

224 • SELECT *
225   FROM Users
226   WHERE date_of_birth > '1990-01-01';
227
228   #5.To find playlists create after August 10, 2024
229
230 • SELECT *
231   FROM Playlists
232   WHERE created_at < '2024-08-10';
233
234
235   #6 To find songs in the genre 'Pop'
  
```

Result Grid

playlist_id	playlist_name	user_id	created_at
1	Rock Classics	1	2024-08-01 10:00:00
2	Top Hits 2024	2	2024-08-01 11:15:00
3	Chill Vibes	3	2024-08-02 14:30:00
4	World Music	4	2024-08-03 09:00:00
5	Indie Gems	5	2024-08-04 16:00:00
6	Pop Favorites	6	2024-08-05 12:15:00
7	Hip-Hop Essentials	7	2024-08-06 08:30:00
8	Tarz Naike	8	2024-08-07 18:00:00

Playlists 5 x

Result Grid

Form Editor

Apply Revert Context Help Snippets

Output

Action Output

```

# Time Action Message Duration / Fetch
5 17:41:21 SELECT * FROM Playlists WHERE user_id = 1 LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
6 17:42:37 SELECT * FROM Users WHERE date_of_birth >'1990-11-05' LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
7 17:43:41 SELECT * FROM Users WHERE date_of_birth >='1990-01-01' LIMIT 0, 1000 11 row(s) returned 0.000 sec / 0.000 sec
8 17:44:39 SELECT * FROM Playlists WHERE created_at < '2024-08-10' LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec
  
```

Object Info Session

Query Completed

Output

Action Output

#6.To find songs in the genre 'Pop'

SELECT *

FROM Songs

WHERE genre = 'Pop';

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: schemas

SCHEMAS: emp_info, hospital_management_system, mydatabase, nikesh, pizza_sales_analysis, spotify, sys

Tables: Views, Stored Procedures, Functions, spotify_database

Administration Schemas Information

No object selected

Object Info Session

Query Completed

Output

Action Output

```

230 • SELECT *
231   FROM Playlists
232   WHERE created_at < '2024-08-10';
233
234
235   #6.To find songs in the genre 'Pop'
236 • SELECT *
237   FROM Songs
238   WHERE genre = 'Pop';
239
240
241   #7 To find songs with a duration between 120 and 180 seconds
  
```

Result Grid

song_id	song_name	duration	album_id	genre
2	Hal	263	2	Pop
6	Shape of You	233	6	Pop
9	Bad Guy	194	9	Pop
11	7 Rings	179	11	Pop
11	HAL	HAL	HAL	HAL

Songs 6 x

Result Grid

Form Editor

Apply Revert Context Help Snippets

Output

Action Output

```

# Time Action Message Duration / Fetch
6 17:42:37 SELECT * FROM Users WHERE date_of_birth >'1990-11-05' LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
7 17:43:41 SELECT * FROM Users WHERE date_of_birth >='1990-01-01' LIMIT 0, 1000 11 row(s) returned 0.000 sec / 0.000 sec
8 17:44:39 SELECT * FROM Playlists WHERE created_at < '2024-08-10' LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec
9 17:45:50 SELECT * FROM Songs WHERE genre = 'Pop' LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec
  
```

Object Info Session

Query Completed

Output

Action Output

#7.To find songs with a duration between 250 and 300 seconds

```
SELECT *  
FROM Songs  
WHERE duration BETWEEN 250 AND 300;
```

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is 'spotify'.
- SQL Editor:** Contains the following SQL code:

```
236 • SELECT *  
237   FROM Songs  
238   WHERE genre = 'Pop'  
239  
240  
241   #7.To find songs with a duration between 250 and 300 seconds  
242 • SELECT *  
243   FROM Songs  
244   WHERE duration BETWEEN 250 AND 300;  
245  
246   #8.To find the average duration of all songs  
247 • SELECT AVG(duration) AS average_duration
```
- Result Grid:** Displays the results of the last query (line 244). The columns are song_id, song_name, duration, album_id, and genre. The data includes:

song_id	song_name	duration	album_id	genre
1	Come Together	259	1	Rock
2	Hello	263	2	Pop
4	Hello	258	4	Soul
13	Uptown Funk	269	13	Funk
...
- Action Output:** Shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
7	17:43:41	SELECT * FROM Users WHERE date_of_birth >='1990-01-01' LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec
8	17:44:39	SELECT * FROM Playlists WHERE created_at < '2024-08-10' LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
9	17:45:50	SELECT * FROM Songs WHERE genre = 'Pop' LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
10	17:46:59	SELECT * FROM Songs WHERE duration BETWEEN 250 AND 300 LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

#8.To find the average duration of all songs

```
SELECT AVG(duration) AS average_duration  
FROM Songs;
```

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is 'spotify'.
- SQL Editor:** Contains the following SQL code:

```
245  
246   #8.To find the average duration of all songs  
247 • SELECT AVG(duration) AS average_duration  
248   FROM Songs;  
249  
250   #9.To find the maximum duration of any song  
251 • SELECT MAX(duration) AS max_duration  
252   FROM Songs;  
253  
254   #10.To find the minimum duration of any song  
255 • SELECT MIN(duration) AS min_duration  
256   FROM Songs;
```
- Result Grid:** Displays the results of the last query (line 247). The column is average_duration. The data is:

average_duration
253.9333
- Action Output:** Shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
8	17:44:39	SELECT * FROM Playlists WHERE created_at < '2024-08-10' LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
9	17:45:50	SELECT * FROM Songs WHERE genre = 'Pop' LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
10	17:46:59	SELECT * FROM Songs WHERE duration BETWEEN 250 AND 300 LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
11	17:47:48	SELECT AVG(duration) AS average_duration FROM Songs LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

#9.To find the maximum duration of any song

```
SELECT MAX(duration) AS max_duration
```

```
FROM Songs
```

The screenshot shows the MySQL Workbench interface. The SQL editor tab contains the following code:

```
245
246  #8.To find the average duration of all songs
247 •   SELECT AVG(duration) AS average_duration
248   FROM Songs;
249
250  #9.To find the maximum duration of any song
251 •   SELECT MAX(duration) AS max_duration
252   FROM Songs;
253
254  #10.To find the minimum duration of any song
255 •   SELECT MIN(duration) AS min_duration
256   FROM Songs;
```

The result grid shows the output of the last query:

max_duration
341

The status bar at the bottom right indicates the date and time as 26-08-2024 17:50.

#10.To find the minimum duration of any song

```
SELECT MIN(duration) AS min_duration
```

```
FROM Songs;
```

The screenshot shows the MySQL Workbench interface. The SQL editor tab contains the following code:

```
248
249
250  #9.To find the maximum duration of any song
251 •   SELECT MAX(duration) AS max_duration
252   FROM Songs;
253
254  #10.To find the minimum duration of any song
255 •   SELECT MIN(duration) AS min_duration
256   FROM Songs;
257
258  #11.To get the total number of users
259 •   SELECT COUNT(*) AS total_users
260   FROM Songs;
```

The result grid shows the output of the last query:

min_duration
177

The status bar at the bottom right indicates the date and time as 26-08-2024 17:51.

#11.To get the total number of users

```
SELECT COUNT(*) AS total_users
```

```
FROM Users;
```

The screenshot shows the MySQL Workbench interface with a query editor window titled 'spotify'. The code entered is:

```
254 #10.To find the minimum duration of any song
255 • SELECT MIN(duration) AS min_duration
256 FROM Songs;
257
258 #11.To get the total number of users
259 • SELECT COUNT(*) AS total_users
260 FROM Users;
261
262 #12.to get the total number of songs in the database
263 • SELECT COUNT(*) AS total_songs
264 FROM Songs;
265
```

The result grid shows the output for the 'total_users' query:

total_users
15

The 'Result 11' tab shows the execution details for the queries:

Action	Time	Message	Duration / Fetch
11 17:47:48	SELECT AVG(duration) AS average_duration FROM Songs LIMIT 0, 1000	1row(s) returned	0.000 sec / 0.000 sec
12 17:50:35	SELECT MAX(duration) AS max_duration FROM Songs LIMIT 0, 1000	1row(s) returned	0.000 sec / 0.000 sec
13 17:51:38	SELECT MIN(duration) AS min_duration FROM Songs LIMIT 0, 1000	1row(s) returned	0.000 sec / 0.000 sec
14 17:52:46	SELECT COUNT(*) AS total_users FROM Users LIMIT 0, 1000	1row(s) returned	0.000 sec / 0.000 sec

#12.to get the total number of songs in the database

```
SELECT COUNT(*) AS total_songs
```

```
FROM Songs;
```

The screenshot shows the MySQL Workbench interface with a query editor window titled 'spotify'. The code entered is:

```
254 #10.To find the minimum duration of any song
255 • SELECT MIN(duration) AS min_duration
256 FROM Songs;
257
258 #11.To get the total number of users
259 • SELECT COUNT(*) AS total_users
260 FROM Users;
261
262 #12.to get the total number of songs in the database
263 • SELECT COUNT(*) AS total_songs
264 FROM Songs;
265
```

The result grid shows the output for the 'total_songs' query:

total_songs
15

The 'Result 12' tab shows the execution details for the queries:

Action	Time	Message	Duration / Fetch
12 17:50:35	SELECT MAX(duration) AS max_duration FROM Songs LIMIT 0, 1000	1row(s) returned	0.000 sec / 0.000 sec
13 17:51:38	SELECT MIN(duration) AS min_duration FROM Songs LIMIT 0, 1000	1row(s) returned	0.000 sec / 0.000 sec
14 17:52:46	SELECT COUNT(*) AS total_users FROM Users LIMIT 0, 1000	1row(s) returned	0.000 sec / 0.000 sec
15 17:53:24	SELECT COUNT(*) AS total_songs FROM Songs LIMIT 0, 1000	1row(s) returned	0.000 sec / 0.000 sec

#13.To get all users sorted alphabetically by their username

SELECT *

FROM Users

ORDER BY username ASC;

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema with the **spotify** database selected.
- SQL Editor:** Contains the following SQL code:

```
268
269 #13.To get all users sorted alphabetically by their username
270
271 SELECT *
272 FROM Users
273 ORDER BY username ASC;
```
- Result Grid:** Displays the results of the query, showing 13 rows of user data. The columns are: user_id, username, email, date_of_birth, and created_at. The data includes entries like alice_jones, bob_smith, charlie_brown, diana_prince, edward_dullen, flora_apple, george_clark, and kenneth_white.
- Action Output:** Shows the history of actions taken in the session, including the execution of various queries and their execution times.
- System Bar:** Includes the Windows taskbar at the bottom with icons for Start, Search, Task View, File Explorer, Edge, File Manager, and others.

#14>To find all songs sorted by duration in descending order

SELECT *

FROM Songs

ORDER BY duration DESC;

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema with the **spotify** database selected.
- SQL Editor:** Contains the following SQL code:

```
268 #13.To get all users sorted alphabetically by their username
269
270 SELECT *
271 FROM Users
272 ORDER BY username ASC;
```



```
273
274 #14.To find all songs sorted by duration in descending order
275
276 #15.To find songs in a specific album (e.g., album_id = 1) sorted alphabetically by song name
277
```
- Result Grid:** Displays the results of the query, showing 14 rows of song data. The columns are: song_id, song_name, duration, album_id, and genre. The data includes entries like Rocket Man, Paranoid Android, Love Yourself, Clocker, Hello, Uptown Funk, Halo, and Come Together.
- Action Output:** Shows the history of actions taken in the session, including the execution of various queries and their execution times.
- System Bar:** Includes the Windows taskbar at the bottom with icons for Start, Search, Task View, File Explorer, Edge, File Manager, and others.

#15. To find songs in a specific album (e.g., album_id = 1) sorted alphabetically by song name

```
SELECT song_id, song_name, duration
```

FROM Songs

WHERE album_id = 1

ORDER BY song_name ASC;

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar displays the Navigator and Schemas. The Navigator pane shows the current database is 'Local instance MySQL80 - W...', and the Schemas pane lists several databases like 'emp_info', 'hr', 'information_schema', 'mysql', 'nikeplus', 'pizza_sales_analysis', 'spothit', and 'sys'. The main workspace contains a query editor with two queries:

```
272 • SELECT *  
273   FROM Songs  
274   ORDER BY duration DESC;  
275  
#15.To find songs in a specific album (e.g., album_id = 1) sorted alphabetically by song name  
276  
277 • SELECT song_id, song_name, duration  
278   FROM Songs  
279   WHERE album_id = 1  
280   ORDER BY song_name ASC;  
281  
282 #16. To count the number of songs in each album
```

The results grid shows the output of the second query:

song_id	song_name	duration
1	Come Together	259
*		

The bottom section shows the SQL History and Action Output:

Songs 15 ×

Action Output

#	Time	Action	Message	Duration / Fetch
15	17:53:24	SELECT COUNT(*) AS total_songs FROM Songs LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
16	17:54:23	SELECT * FROM Users ORDER BY username ASC LIMIT 0, 1000	15 row(s) returned	0.000 sec / 0.000 sec
17	17:55:08	SELECT * FROM Songs ORDER BY duration DESC LIMIT 0, 1000	15 row(s) returned	0.000 sec / 0.000 sec
18	17:56:15	SELECT song_id, song_name, duration FROM Songs WHERE album_id = 1 ORDER BY song_name ASC LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

#16.To count the number of songs in each genre

```
SELECT genre, COUNT(*) AS num_songs
```

FROM Songs

GROUP BY genre

ORDER BY num_songs DESC;

The screenshot shows the MySQL Workbench interface. On the left, the Navigator pane displays the database schema with tables like emp_info, hospital_management_system, nilesh, and songs_sales_analysis. The ER Diagram pane shows the relationships between these tables. In the center, the SQL Editor pane contains a query to count songs by genre:

```
278 • SELECT song_id, song_name, duration
279   FROM Songs
280  WHERE album_id = 1
281  ORDER BY song_name ASC
282
283 --EE:To count the number of songs in each genre
284 • SELECT genre, COUNT(*) AS num_songs
285   FROM Songs
286  GROUP BY genre
287  ORDER BY num_songs DESC
288
289
```

The Results Grid pane shows the output of the second part of the query:

genre	num_songs
Pop	4
Hip-Hop	3
Alternative Rock	2
R&B	1
Soul	1
Country	1
Rap	1
Punk	1

The bottom status bar shows the current time as 17:54:23 and the date as 26-08-2024.

#17.To find the number of users born on each date

```
SELECT date_of_birth, COUNT(*) AS num_users
```

```
FROM Users
```

```
GROUP BY date_of_birth
```

```
ORDER BY date_of_birth;
```

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** spotty
- Queries:** A list of 294 queries, with the last few being:
 - 294. #17.To find the number of users born on each date;
 - 295. SELECT genre, COUNT(*) AS num_songs FROM Songs GROUP BY genre ORDER BY num_songs DESC;
 - 296. SELECT date_of_birth, COUNT(*) AS num_users FROM Users GROUP BY date_of_birth ORDER BY num_users DESC;
 - 297. SELECT genre, AVG(duration) AS average_duration FROM Songs GROUP BY genre ORDER BY average_duration DESC;
 - 298. #18.To find the total duration of songs for each genre
 - 299. SELECT genre, SUM(duration) AS total_duration FROM Songs GROUP BY genre
- Result Grid:** Shows the results for the last query (298).

genre	total_duration
Pop Rock	341.0000
Rap	326.0000
Alternative Rock	299.0000
Soul	295.0000
Funk	269.0000
Rock	230.0000
Country	231.0000
R'n'B	717.7400
- Action Output:** Shows the execution log for the last query (298).

#	Time	Action	Message	Duration / Fetch
17	17:55:08	SELECT * FROM Songs ORDER BY duration DESC LIMIT 0, 1000	15 rows(s) returned	0.000 sec / 0.000 sec
18	17:56:15	SELECT song_id, song_name, duration FROM Songs WHERE album_id = 1 ORDER BY song_name ASC LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
19	17:57:56	SELECT genre, COUNT(*) AS num_songs FROM Songs GROUP BY genre ORDER BY num_songs DESC LIMIT 0, 1000	9 rows(s) returned	0.000 sec / 0.000 sec
20	17:59:14	SELECT date_of_birth, COUNT(*) AS num_users FROM Users GROUP BY date_of_birth ORDER BY date_of_birth DESC LIMIT 0, 1000	15 rows(s) returned	0.000 sec / 0.000 sec
21	18:00:22	SELECT genre, AVG(duration) AS average_duration FROM Songs GROUP BY genre ORDER BY average_duration DESC LIMIT 0, 1000	9 rows(s) returned	0.000 sec / 0.000 sec

#18.To find the average duration of songs in each genre

```
SELECT genre, AVG(duration) AS average_duration
```

```
FROM Songs
```

```
GROUP BY genre
```

```
ORDER BY average_duration DESC;
```

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** spotty
- Queries:** A list of 303 queries, with the last few being:
 - 298. #17.To find the number of users born on each date;
 - 299. SELECT genre, COUNT(*) AS num_songs FROM Songs GROUP BY genre ORDER BY num_songs DESC LIMIT 0, 1000
 - 300. SELECT date_of_birth, COUNT(*) AS num_users FROM Users GROUP BY date_of_birth ORDER BY date_of_birth DESC LIMIT 0, 1000
 - 301. #18.To find the average duration of songs in each genre
 - 302. SELECT genre, AVG(duration) AS average_duration FROM Songs GROUP BY genre ORDER BY average_duration DESC
 - 303. #19.To find the total duration of songs for each genre
 - 304. SELECT genre, SUM(duration) AS total_duration FROM Songs GROUP BY genre
- Result Grid:** Shows the results for the last query (302).

genre	average_duration
Pop Rock	341.0000
Rap	326.0000
Alternative Rock	299.0000
Soul	295.0000
Funk	269.0000
Rock	230.0000
Country	231.0000
R'n'B	717.7400
- Action Output:** Shows the execution log for the last query (302).

#	Time	Action	Message	Duration / Fetch
18	17:56:15	SELECT song_id, song_name, duration FROM Songs WHERE album_id = 1 ORDER BY song_name ASC LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
19	17:57:56	SELECT genre, COUNT(*) AS num_songs FROM Songs GROUP BY genre ORDER BY num_songs DESC LIMIT 0, 1000	9 rows(s) returned	0.000 sec / 0.000 sec
20	17:59:14	SELECT date_of_birth, COUNT(*) AS num_users FROM Users GROUP BY date_of_birth ORDER BY date_of_birth DESC LIMIT 0, 1000	15 rows(s) returned	0.000 sec / 0.000 sec
21	18:00:22	SELECT genre, AVG(duration) AS average_duration FROM Songs GROUP BY genre ORDER BY average_duration DESC LIMIT 0, 1000	9 rows(s) returned	0.000 sec / 0.000 sec

#19.To find the total duration of songs for each genre

```
SELECT genre, SUM(duration) AS total_duration
```

```
FROM Songs
```

```
GROUP BY genre;
```

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is 'spotify'.
- Query Editor:** The code for query #19 is displayed:

```
299 GROUP BY genre
300 ORDER BY average_duration DESC;
301
302 #19.To find the total duration of songs for each genre
303 • SELECT genre, SUM(duration) AS total_duration
304 FROM Songs
305 GROUP BY genre
306
307 • use spotify;
308 #20. to extract the month and year from a date
309 • SELECT
310     EXTRACT(MONTH FROM release_date) AS month,
```
- Result Grid:** The results show the total duration for each genre:

genre	total_duration
Rock	297
Pop	869
Hip-Hop	582
Soul	295
Country	231
Alternative Rock	637
Rap	326
Indie	565
- Action Output:** Shows the history of executed queries with their times and durations.

#20. to extract the month and year from a date

```
SELECT
```

```
EXTRACT(MONTH FROM release_date) AS month,
```

```
EXTRACT(YEAR FROM release_date) AS year
```

```
FROM ALBUMS;
```

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is 'spotify'.
- Query Editor:** The code for query #20 is displayed:

```
303 • SELECT genre, SUM(duration) AS total_duration
304 FROM Songs
305 GROUP BY genre
306
307 • use spotify;
308 #20. to extract the month and year from a date
309 • SELECT
310     EXTRACT(MONTH FROM release_date) AS month,
311     EXTRACT(YEAR FROM release_date) AS year
312
313 FROM
314     albums
315
316 #21. To count the number of songs in each album
```
- Result Grid:** The results show the month and year for each album:

month	year
9	1969
4	2016
6	2018
11	2015
10	2014
3	2013
4	2017
4	1997
- Action Output:** Shows the history of executed queries with their times and durations.

#21.To count the number of songs in each album:

```
SELECT a.album_id, a.album_name, COUNT(s.song_id) AS num_songs  
FROM Albums a  
JOIN Songs s ON a.album_id = s.album_id  
GROUP BY a.album_id, a.album_name  
ORDER BY num_songs DESC;
```

The screenshot shows the MySQL Workbench interface. The SQL editor tab contains the following query:

```
309 • SELECT  
310   EXTRACT(MONTH FROM release_date) AS month,  
311   EXTRACT(YEAR FROM release_date) AS year  
312  FROM  
313    albums ;  
314  #21.To count the number of songs in each album:  
315  • SELECT a.album_id, a.album_name, COUNT(s.song_id) AS num_songs  
316  FROM Albums a  
317  JOIN Songs s ON a.album_id = s.album_id  
318  GROUP BY a.album_id, a.album_name  
319  ORDER BY num_songs DESC;
```

The results grid shows the following data:

album_id	album_name	num_songs
1	Abber Road	1
2	Lemonade	1
3	Skeleton	1
4	25	1
5	1989	1
6	Divide	1
7	DAMN.	1
8	Fever Dreamer	1

The output pane shows the execution log:

Action	Time	Message	Duration / Fetch
21	18:00:22	SELECT genre, AVG(duration) AS average_duration FROM Songs GROUP BY genre ORDER BY average_du...	0.000 sec / 0.000 sec
22	18:01:35	SELECT genre, SUM(duration) AS total_duration FROM Songs GROUP BY genre LIMIT 0, 1000	0.000 sec / 0.000 sec
23	18:02:23	SELECT EXTRACT(MONTH FROM release_date) AS month, EXTRACT(YEAR FROM release_date) AS...	0.000 sec / 0.000 sec
24	18:04:17	SELECT a.album_id, a.album_name, COUNT(s.song_id) AS num_songs FROM Albums a JOIN Songs s ON a...	0.000 sec / 0.000 sec

#22.To find the number of streams per song, grouped by date of streaming

```
SELECT s.song_id, s.song_name, DATE(sh.stream_time) AS stream_date,  
COUNT(sh.stream_id) AS daily_streams  
FROM Songs s  
JOIN StreamingHistory sh ON s.song_id = sh.song_id  
GROUP BY s.song_id, s.song_name, stream_date  
ORDER BY stream_date DESC;
```

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
    Filter objects
    emp_info
    hospital_management_system
    mydatabase
    nikes
    pizza_sales_analysis
    spotify
        Tables
        Views
        Stored Procedures
        Functions
        spottify_database
        sys
    Administration Schemas
Information
No object selected
Object Info Session
Query Completed
82°F Mostly cloudy
Result Grid | Filter Rows | Export: | Wrap Cell Content: |
song_id song_name stream_date daily_streams
12 Closer 2024-08-07 1
7 HUMBLE. 2024-08-06 1
14 Gold Dagger 2024-08-06 1
10 Lose Yourself 2024-08-05 1
11 7 Rings 2024-08-04 1
5 Blank Space 2024-08-04 1
13 Uptown Funk 2024-08-04 1
4 Hold 2024-08-04 1

```

Action Output

#	Time	Action	Message	Duration / Fetch
22	18:01:35	SELECT genre, SUM(duration) AS total_duration FROM Songs GROUP BY genre LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec
23	18:02:23	SELECT EXTRACT(MONTH FROM release_date) AS month, EXTRACT(YEAR FROM release_date) AS year FROM Albums	15 row(s) returned	0.000 sec / 0.000 sec
24	18:04:17	SELECT a.album_id, a.album_name, COUNT(s.song_id) AS num_songs FROM Albums a JOIN Songs s ON a.album_id = s.album_id GROUP BY a.album_id, a.album_name	15 row(s) returned	0.000 sec / 0.000 sec
25	18:05:25	SELECT s.song_id, s.song_name, DATE(sh.stream_time) AS stream_date, COUNT(sh.stream_id) AS daily_streams FROM Songs s INNER JOIN StreamingHistory sh ON s.song_id = sh.song_id GROUP BY s.song_id, s.song_name, stream_date ORDER BY stream_date DESC;	15 row(s) returned	0.000 sec / 0.000 sec

#22.To get a list of all songs along with their respective album names and artist names

`SELECT s.song_id, s.song_name, a.album_name, ar.artist_name`

`FROM Songs s`

`INNER JOIN Albums a ON s.album_id = a.album_id`

`INNER JOIN Artists ar ON a.artist_id = ar.artist_id;`

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
    Filter objects
    emp_info
    hospital_management_system
    mydatabase
    nikes
    pizza_sales_analysis
    spotify
        Tables
        Views
        Stored Procedures
        Functions
        spottify_database
        sys
    Administration Schemas
Information
No object selected
Object Info Session
Query Completed
82°F Mostly cloudy
Result Grid | Filter Rows | Export: | Wrap Cell Content: |
song_id song_name album_name artist_name
1 Come Together Abbey Road The Beatles
2 Halo Lemonade Beyoncé
3 God's Plan Scorpion Drake
4 Hello 25 Adele
5 Blank Space 1989 Taylor Swift
6 Shape of You Divide Ed Sheeran
7 HUMBLE. DAMN. Kendrick Lamar
8 Paranoid Android OK Computer Radiohead

```

Action Output

#	Time	Action	Message	Duration / Fetch
23	18:02:23	SELECT EXTRACT(MONTH FROM release_date) AS month, EXTRACT(YEAR FROM release_date) AS year FROM Albums	15 row(s) returned	0.000 sec / 0.000 sec
24	18:04:17	SELECT a.album_id, a.album_name, COUNT(s.song_id) AS num_songs FROM Albums a JOIN Songs s ON a.album_id = s.album_id GROUP BY a.album_id, a.album_name	15 row(s) returned	0.000 sec / 0.000 sec
25	18:05:25	SELECT s.song_id, s.song_name, DATE(sh.stream_time) AS stream_date, COUNT(sh.stream_id) AS daily_streams FROM Songs s INNER JOIN StreamingHistory sh ON s.song_id = sh.song_id GROUP BY s.song_id, s.song_name, stream_date ORDER BY stream_date DESC;	15 row(s) returned	0.000 sec / 0.000 sec
26	18:06:22	SELECT s.song_id, s.song_name, a.album_name, ar.artist_name FROM Songs s INNER JOIN Albums a ON s.album_id = a.album_id INNER JOIN Artists ar ON a.artist_id = ar.artist_id;	15 row(s) returned	0.000 sec / 0.000 sec

#23. To list all playlists along with the username of the user who created each playlist

```
SELECT p.playlist_id, p.playlist_name, u.username
```

FROM Playlists p

```
INNER JOIN Users u ON p.user_id = u.user_id;
```

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar has sections for Navigator, Schemas, Administration, and Information, with 'No object selected' currently highlighted. The main area displays an EER Diagram titled 'hms' with various tables like songs, albums, artists, playlists, and users. Below the diagram is a SQL editor window containing two queries labeled '#22.' and '#23.'. The results of query #23 are shown in a Result Grid, displaying columns for playlist_id, playlist_name, and username, with data rows for Alice Jones, Bob Smith, Charlie Brown, Diana Prince, Edward Cullen, Fiona Apple, George Clark, and Hannah White. To the right of the result grid is a sidebar with tabs for Result Grid, Form Editor, and Context Help. At the bottom, there's an Action Output section showing log entries for various MySQL statements, and a status bar at the very bottom.

#24. This query uses a LEFT JOIN to list all users, including those who haven't created any playlists

```
SELECT u.user_id, u.username, p.playlist_name
```

FROM Users u

```
LEFT JOIN Playlists p ON u.user_id = p.user_id;
```

The screenshot shows the MySQL Workbench interface. The top navigation bar includes 'File', 'Edit', 'View', 'Query', 'Database', 'Server', 'Tools', 'Scripting', and 'Help'. The left sidebar has sections for 'Navigator' (Schemas, Projects), 'Administration' (Information, Schemas), and 'Object Info' (No object selected). The main area displays an 'EER Diagram' titled 'spotify' with various tables like 'emp_info', 'hosted_database', 'nikest', 'pizza_sales_analysis', and 'songs'. Below the diagram is a SQL editor window containing several queries related to playlists and users. The results of the last query are shown in a grid format:

user_id	username	playlist_name
1	alice_jones	Rock Classics
2	bob_smith	Top Hits 2024
3	charlie_brown	Summer Vibes
4	david_graham	World Mix
5	edward_cullen	Indie Gems
6	fiona_apple	Pop Favorites
7	george_dark	Hip-Hop Essentials
8	hannah_white	Tech Nostalgia

The bottom status bar shows 'Query Completed' and the system tray includes icons for network, battery, and volume.

#25.This query lists all artists, including those who haven't released any albums

```
SELECT ar.artist_id, ar.artist_name, a.album_name
```

```
FROM Artists ar
```

```
LEFT JOIN Albums a ON ar.artist_id = a.artist_id;
```

The screenshot shows the MySQL Workbench interface with the SQL editor tab open. The code for query #25 is pasted into the editor. The results grid shows the output of the query, listing artists and their albums. The output pane at the bottom shows the execution log with four entries corresponding to the statements in the code.

artist_id	artist_name	album_name
1	The Beatles	Abbey Road
2	Beyoncé	Lemonade
3	Drake	Scorpion
4	Adele	25
5	Taylor Swift	1989
6	Ed Sheeran	Divide
7	Kendrick Lamar	DAMN.
8	Darkhoekid	OK Computer

Action Output

#	Time	Action	Message	Duration / Fetch
26	18:06:22	SELECT s.song_id, s.song_name, a.album_name, ar.artist_name FROM Songs s INNER JOIN Albums a ON s.album_id = a.album_id; 15 row(s) returned		0.000 sec / 0.000 sec
27	18:07:05	SELECT p.playlist_id, p.playlist_name, u.username FROM Playlists p INNER JOIN Users u ON p.user_id = u.user_id; 15 row(s) returned		0.000 sec / 0.000 sec
28	18:07:49	SELECT u.user_id, u.username, p.playlist_name FROM Users u LEFT JOIN Playlists p ON u.user_id = p.user_id; 15 row(s) returned		0.000 sec / 0.000 sec
29	18:08:54	SELECT ar.artist_id, ar.artist_name, a.album_name FROM Artists ar LEFT JOIN Albums a ON ar.artist_id = a.artist_id; 15 row(s) returned		0.000 sec / 0.000 sec

#26.This query lists all playlists, even if they are not associated with any user

```
SELECT p.playlist_id, p.playlist_name, u.username
```

```
FROM Users u
```

```
RIGHT JOIN Playlists p ON u.user_id = p.user_id;
```

The screenshot shows the MySQL Workbench interface with the SQL editor tab open. The code for query #26 is pasted into the editor. The results grid shows the output of the query, listing playlists and their users. The output pane at the bottom shows the execution log with five entries corresponding to the statements in the code.

playlist_id	playlist_name	username
1	Rock Classics	alex_jones
2	Pop 2024	bob_smith
3	Chill Vibes	charlie_brown
4	Workout Mix	diana_prince
5	Indie Gems	edward_cullen
6	Pop Favorites	fiona_apple
7	Hip-Hop Essentials	george_clark
8	Tamz Ninkhate	hannah.white

Action Output

#	Time	Action	Message	Duration / Fetch
27	18:07:05	SELECT p.playlist_id, p.playlist_name, u.username FROM Playlists p INNER JOIN Users u ON p.user_id = u.user_id; 15 row(s) returned		0.000 sec / 0.000 sec
28	18:07:49	SELECT u.user_id, u.username, p.playlist_name FROM Users u LEFT JOIN Playlists p ON u.user_id = p.user_id; 15 row(s) returned		0.000 sec / 0.000 sec
29	18:08:54	SELECT ar.artist_id, ar.artist_name, a.album_name FROM Artists ar LEFT JOIN Albums a ON ar.artist_id = a.artist_id; 15 row(s) returned		0.000 sec / 0.000 sec
30	18:09:41	SELECT p.playlist_id, p.playlist_name, u.username FROM Users u RIGHT JOIN Playlists p ON u.user_id = p.user_id; 15 row(s) returned		0.000 sec / 0.000 sec

#27.This query lists each artist with their most recent album release

```
SELECT artist_id, album_name, release_date  
FROM Albums a  
WHERE release_date = (  
    SELECT MAX(release_date)  
    FROM Albums  
    WHERE artist_id = a.artist_id  
);
```

The screenshot shows the MySQL Workbench interface with a query editor window containing the code for question #27. The code is as follows:

```
351 FROM Users u  
352     RIGHT JOIN Playlists p ON u.user_id = p.user_id;  
353  
354 #27.This query lists each artist with their most recent album release  
355 • SELECT artist_id, album_name, release_date  
356 FROM Albums a  
357 WHERE release_date = (  
358     SELECT MAX(release_date)  
359     FROM Albums  
360     WHERE artist_id = a.artist_id  
361 );
```

The result grid shows the output of the query:

artist_id	album_name	release_date
1	Alber Road	1989-09-26
2	Lemonade	2016-04-23
3	Scorpion	2018-06-29
4	25	2015-11-20
5	1989	2014-10-27
6	Divide	2017-03-03
7	DAMN.	2017-04-14
8	Mr. Crenshaw	1997-14-71

The output pane shows the execution details:

Action	Time	Action	Message	Duration / Fetch
28	18:07:49	SELECT u.user_id, u.username, p.playlist_name FROM Users u LEFT JOIN Playlists p ON u.user_id = p.user_id;	15 row(s) returned	0.000 sec / 0.000 sec
29	18:08:54	SELECT artist_id, artist_name, a.album_name FROM Artists a LEFT JOIN Albums b ON artist_id = a.artist_id;	15 row(s) returned	0.000 sec / 0.000 sec
30	18:09:41	SELECT p.playlist_id, p.playlist_name, u.username FROM Users u RIGHT JOIN Playlists p ON u.user_id = p.user_id;	15 row(s) returned	0.000 sec / 0.000 sec
31	18:10:34	SELECT artist_id, album_name, release_date FROM Albums a WHERE release_date > (SELECT MAX(rele...	15 row(s) returned	0.000 sec / 0.000 sec

#28.This query lists songs whose duration is longer than the average duration of all songs

```
SELECT song_name, duration  
FROM Songs  
WHERE duration > (SELECT AVG(duration) FROM Songs);
```

The screenshot shows the MySQL Workbench interface. In the top navigation bar, there are tabs for 'MySQL Workbench', 'Local instance MySQL80 - W...', 'MySQL Model', and 'EER Diagram'. The main area is a query editor titled 'hms' containing the following SQL code:

```

357 WHERE release_date = (
358     SELECT MAX(release_date)
359     FROM Albums
360     WHERE artist_id = a.artist_id
361 );
362
363 #28.This query lists songs whose duration is longer than the average duration of all songs.
364 • SELECT song_name, duration
365   FROM Songs
366   WHERE duration > (SELECT AVG(duration) FROM Songs)
367
368

```

Below the code, the 'Result Grid' shows the results of the query:

song_name	duration
Come Together	259
Halo	263
Hello	295
Pearl Jam	300
Lose Yourself	326
Clouds	307
Uptown Funk	269
Rhythm Nation	241

The 'Output' section shows the execution log:

#	Time	Action	Message	Duration / Fetch
29	18:08:54	SELECT artist_id, artist_name, album_name FROM Artists a LEFT JOIN Albums a ON artist_id = a.artist_id WHERE artist_id = 15 rows(s) returned		0.000 sec / 0.000 sec
30	18:09:41	SELECT p.playlist_id, p.playlist_name, u.username FROM Users u RIGHT JOIN Playlists p ON u.user_id = p.user_id WHERE user_id = 15 rows(s) returned		0.000 sec / 0.000 sec
31	18:10:34	SELECT artist_id, album_name, release_date FROM Albums a WHERE release_date > (SELECT MAX(release_date) FROM Songs) LIMIT 8 rows(s) returned		0.000 sec / 0.000 sec
32	18:11:23	SELECT song_name, duration FROM Songs WHERE duration > (SELECT AVG(duration) FROM Songs) LIMIT 8 rows(s) returned		0.000 sec / 0.000 sec

The status bar at the bottom shows the weather as '82°F Mostly cloudy' and the system date and time as '18:11 26-08-2024'.

CONCLUSION:

The Spotify Database Management System is a structured solution that efficiently organizes and manages vast data related to users, artists, albums, songs, playlists, and streaming activities on the platform. The Spotify Database Management System is a crucial component that enhances the platform's operations, user engagement, and business intelligence, ensuring a seamless and personalized experience for its global users.