(CS 425-02) DATABASE ORGANISATION

Project Deliverable - 3

Group Members Details

Name	CWID	EMAIL
DEVESH PATEL	A20548274	dpatel219@hawk.iit.edu
PURVIT ASHESH PATEL	A20551053	ppatel180@hawk.iit.edu
VISHWASHREE CHANNAA REDDY	A20556543	vishwashreech@hawk.iit.edu

Equal Contribution

1) Shows how many songs each artist has.

Explaination: This query counts and lists the total number of songs attributed to each artist.

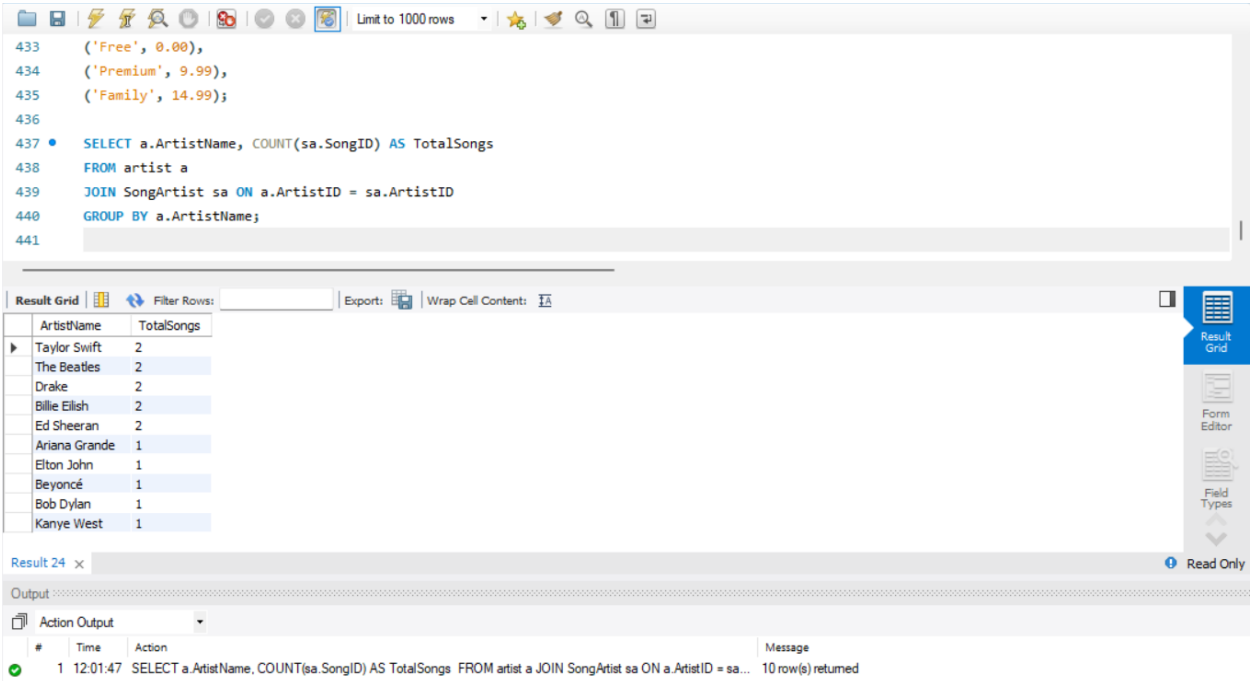
Query:

SELECT a.ArtistName, COUNT(sa.SongID) AS TotalSongs

FROM artist a

JOIN SongArtist sa ON a.ArtistID = sa.ArtistID

GROUP BY a.ArtistName;

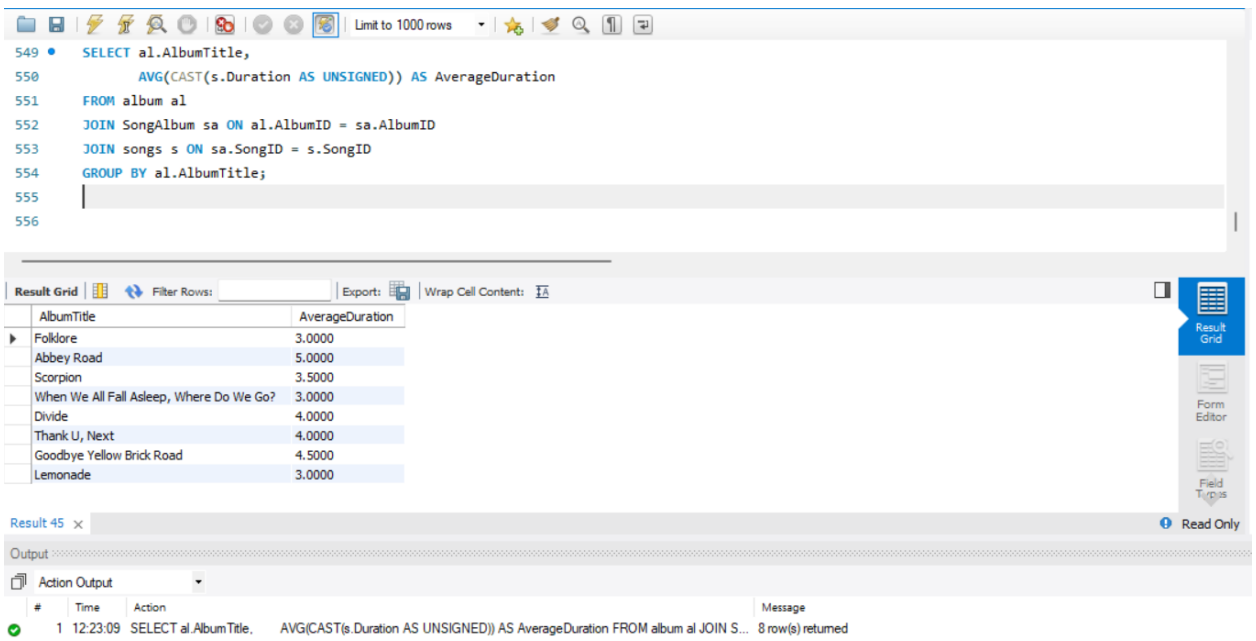


2) Average Duration of Albums

Explanation: This query results the calculation and displays the mean duration of songs across each album in a music database

Query:

```
SELECT al.AlbumTitle,
    AVG(CAST(s.Duration AS UNSIGNED)) AS AverageDuration
FROM album al
JOIN SongAlbum sa ON al.AlbumID = sa.AlbumID
JOIN songs s ON sa.SongID = s.SongID
GROUP BY al.AlbumTitle;
```



3) List all genres with the number of albums associated with each genre.

Explanation: The objective of identifying each music genre available in a database and counting how many albums are categorized under each genre.

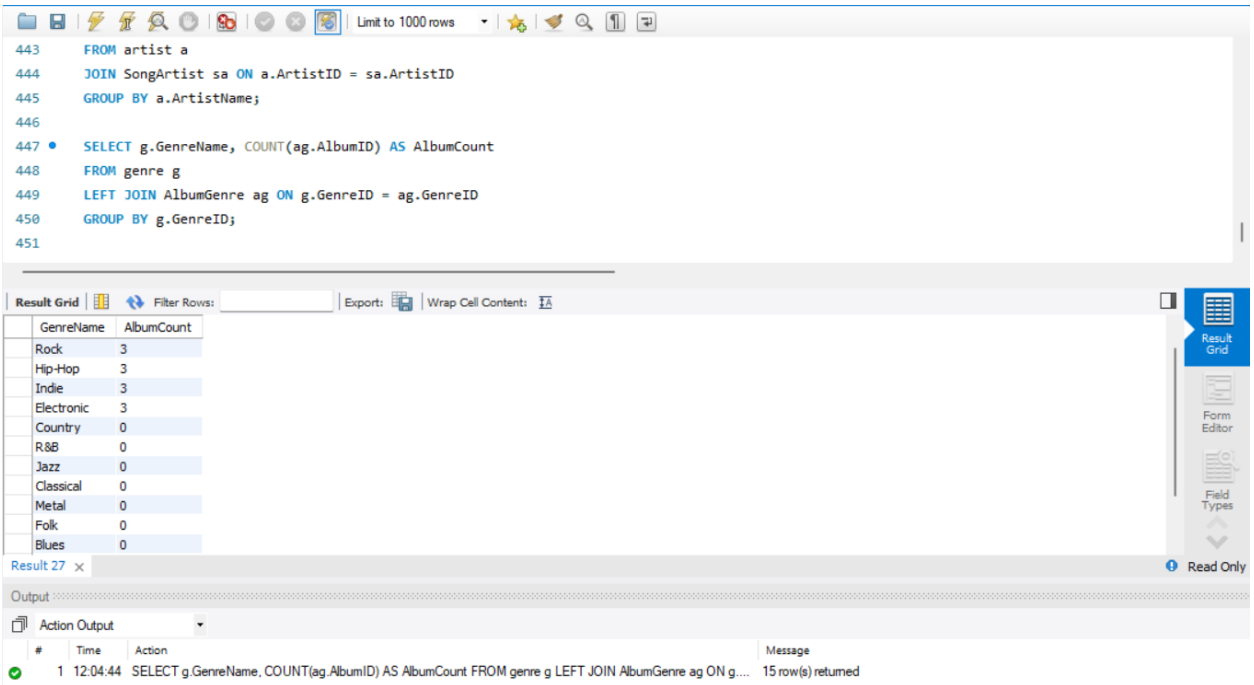
Query:

SELECT g.GenreName, COUNT(ag.AlbumID) AS AlbumCount

FROM genre g

LEFT JOIN AlbumGenre ag ON g.GenreID = ag.GenreID

GROUP BY g.GenreID;



4) List all artists who released albums in the last 5 years.

Explanation: This query uses for identifying and listing artists who have released albums within the past five years.

Query:

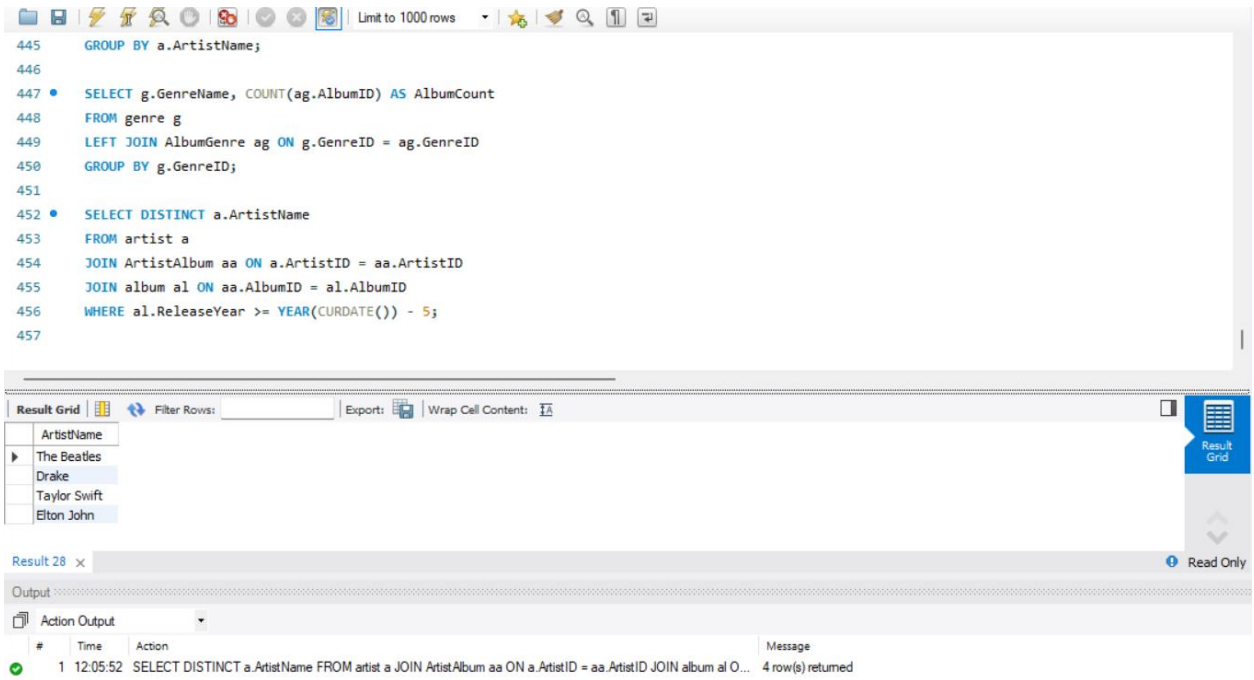
SELECT DISTINCT a.ArtistName

FROM artist a

JOIN ArtistAlbum aa ON a.ArtistID = aa.ArtistID

JOIN album al ON aa.AlbumID = al.AlbumID

WHERE al.ReleaseYear >= YEAR(CURDATE()) - 5;



5) Calculate the average song duration by genre.

Explanation: This query describes the process of calculating the mean duration of songs within each genre in a music database.

Query:

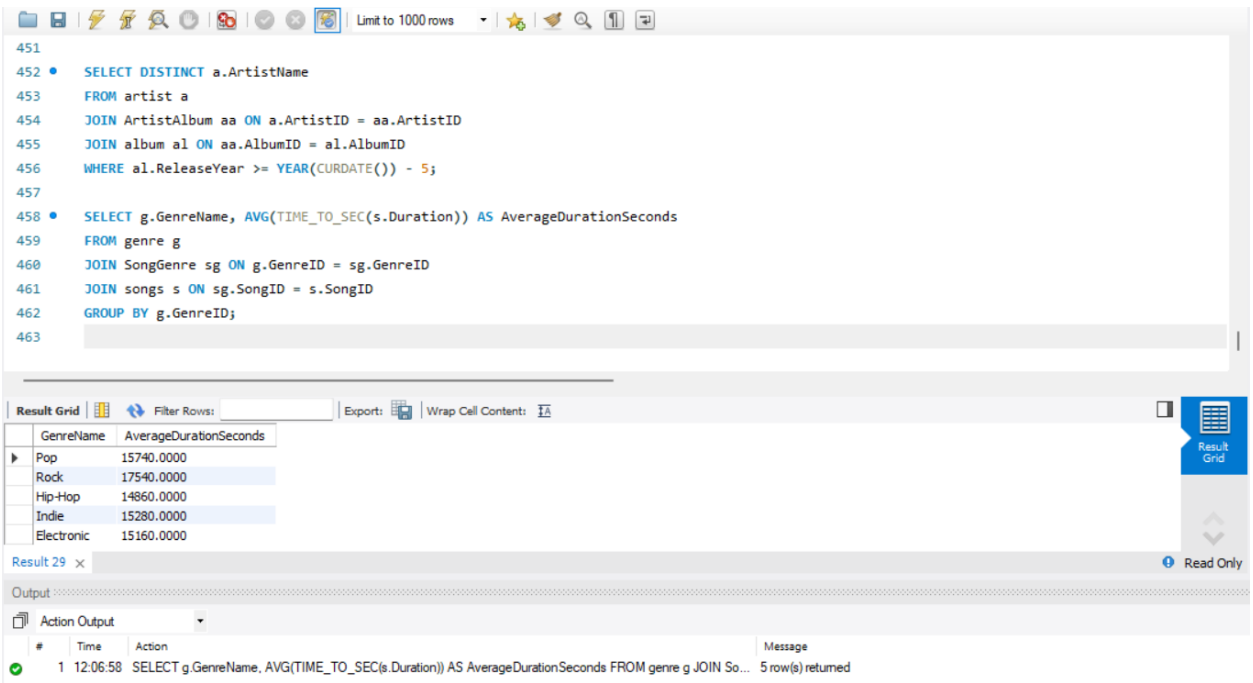
SELECT g.GenreName, AVG(TIME_TO_SEC(s.Duration)) AS AverageDurationSeconds

FROM genre g

JOIN SongGenre sg ON g.GenreID = sg.GenreID

JOIN songs s ON sg.SongID = s.SongID

GROUP BY g.GenreID;



6) Find the user with the most playlists.

Explanation: The query individual user who has created the highest number of playlists in a database, indicating the most active or engaged user in terms of playlist creation.

Query:

SELECT u.Username, COUNT(p.PlaylistID) AS PlaylistCount

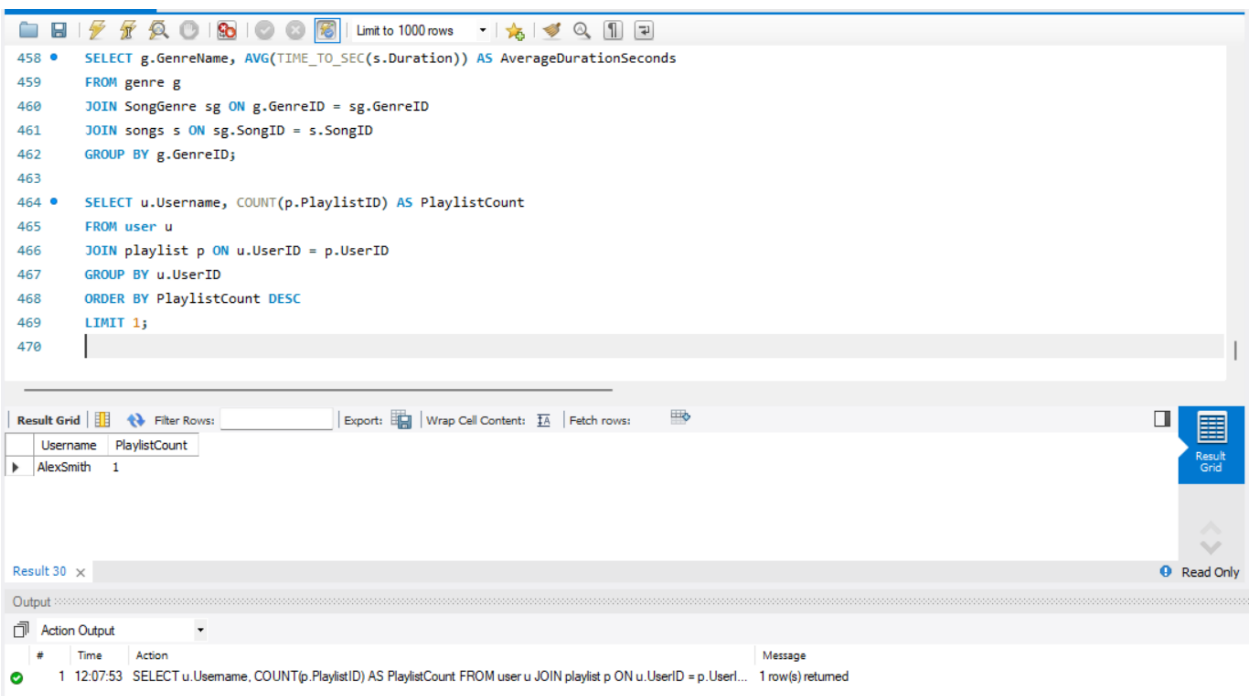
FROM user u

JOIN playlist p ON u.UserID = p.UserID

GROUP BY u.UserID

ORDER BY PlaylistCount DESC

LIMIT 1;



7) Rank albums by release year within each genre.

Explanation: This query involves ordering albums based on their release years within each distinct genre, providing a way to see how albums are temporally distributed across various musical styles.

Query:

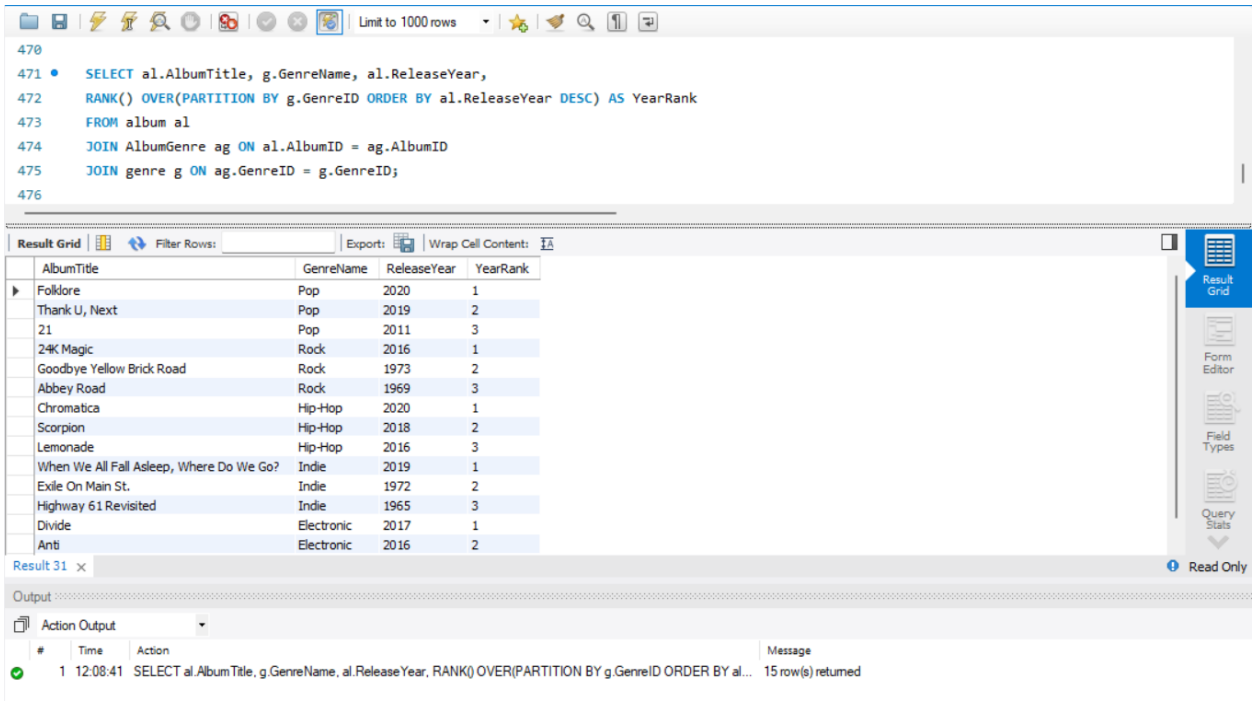
SELECT al.AlbumTitle, g.GenreName, al.ReleaseYear,

RANK() OVER(PARTITION BY g.GenreID ORDER BY al.ReleaseYear DESC) AS YearRank

FROM album al

JOIN AlbumGenre ag ON al.AlbumID = ag.AlbumID

JOIN genre g ON ag.GenreID = g.GenreID;



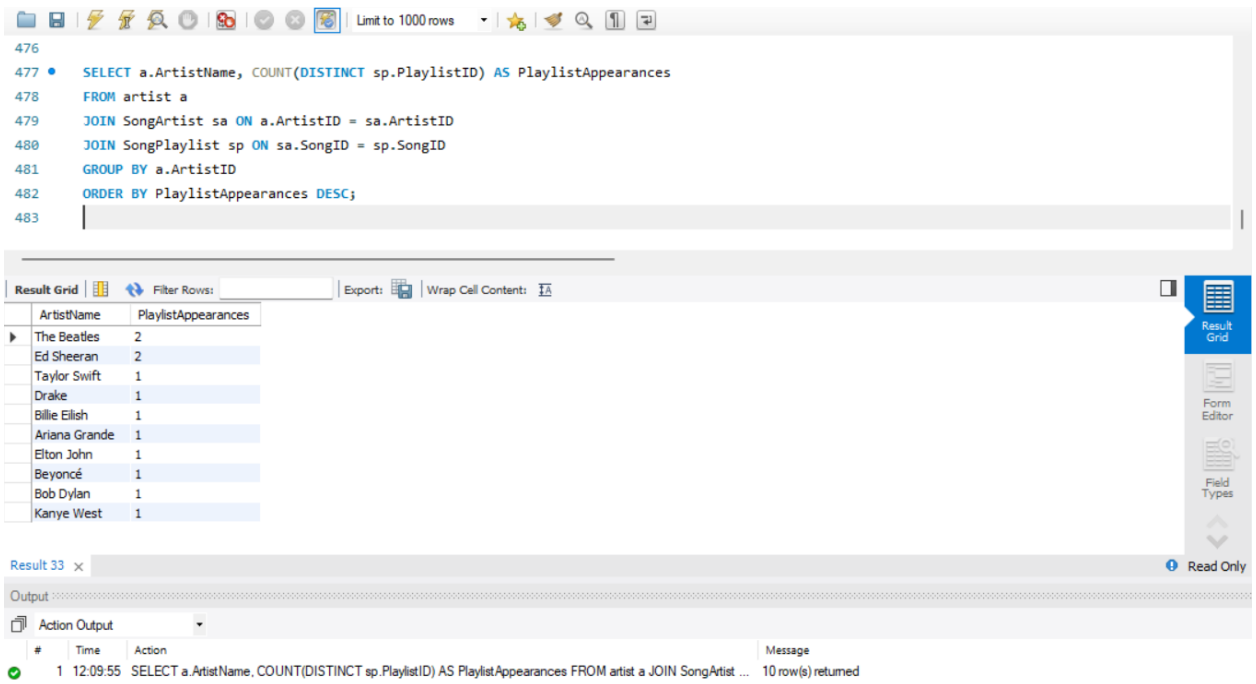
AlbumTitle	GenreName	ReleaseYear	YearRank
Folklore	Pop	2020	1
Thank U, Next	Pop	2019	2
21	Pop	2011	3
24K Magic	Rock	2016	1
Goodbye Yellow Brick Road	Rock	1973	2
Abbey Road	Rock	1969	3
Chromatica	Hip-Hop	2020	1
Scorpion	Hip-Hop	2018	2
Lemonade	Hip-Hop	2016	3
When We All Fall Asleep, Where Do We Go?	Indie	2019	1
Exile On Main St.	Indie	1972	2
Highway 61 Revisited	Indie	1965	3
Divide	Electronic	2017	1
Anti	Electronic	2016	2

8) Rank artists by the number of playlists their songs appear in.

Explanation: This query involves determining the popularity or visibility of artists based on the frequency of their songs' appearances in playlists. By counting how many playlists include at least one song by an artist and then ranking these artists accordingly.

Query:

```
SELECT a.ArtistName, COUNT(DISTINCT sp.PlaylistID) AS PlaylistAppearances

FROM artist a

JOIN SongArtist sa ON a.ArtistID = sa.ArtistID

JOIN SongPlaylist sp ON sa.SongID = sp.SongID

GROUP BY a.ArtistID

ORDER BY PlaylistAppearances DESC;
```

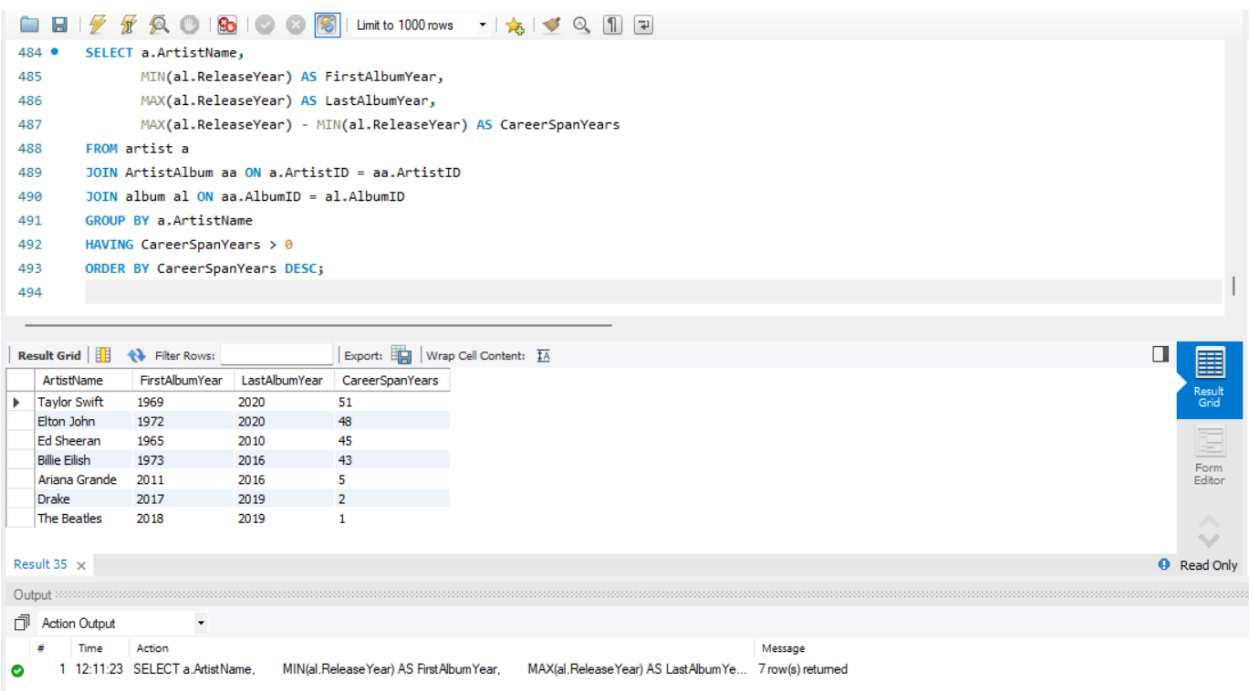


9) Artist's First and Last Album Release Difference

Explanation: This query calculates the time span between the first and last albums released by each artist, offering insights into the length of their active recording careers.

Query:

```
SELECT a.ArtistName,
    MIN(al.ReleaseYear) AS FirstAlbumYear,
    MAX(al.ReleaseYear) AS LastAlbumYear,
    MAX(al.ReleaseYear) - MIN(al.ReleaseYear) AS CareerSpanYears
FROM artist a
JOIN ArtistAlbum aa ON a.ArtistID = aa.ArtistID
JOIN album al ON aa.AlbumID = al.AlbumID
GROUP BY a.ArtistName
HAVING CareerSpanYears > 0
ORDER BY CareerSpanYears DESC;
```



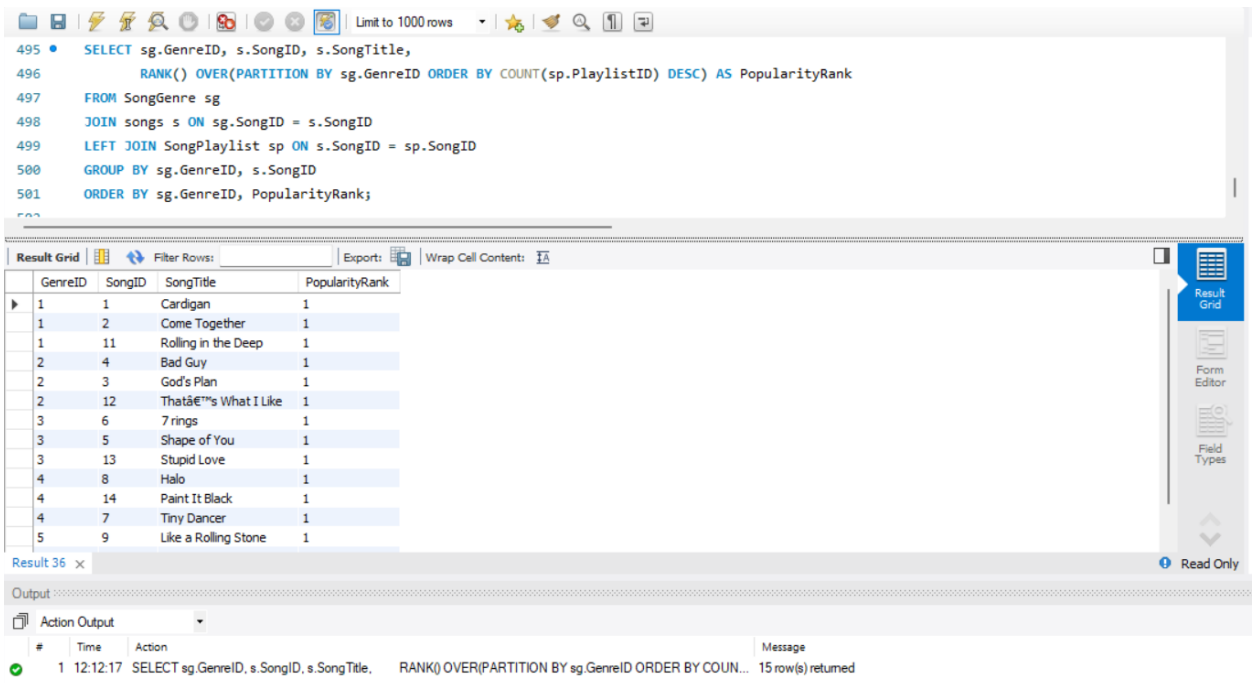
ArtistName	FirstAlbumYear	LastAlbumYear	CareerSpanYears
Taylor Swift	1969	2020	51
Elton John	1972	2020	48
Ed Sheeran	1965	2010	45
Billie Eilish	1973	2016	43
Ariana Grande	2011	2016	5
Drake	2017	2019	2
The Beatles	2018	2019	1

10) Ranking Songs by Popularity Within Each Genre

Explanation: This process involves evaluating and ranking songs based on their popularity within their respective genres. Popularity can be determined by various metrics such as the number of times a song is played, featured in playlists, or downloaded.

Query:

```
SELECT sg.GenreID, s.SongID, s.SongTitle,
      RANK() OVER(PARTITION BY sg.GenreID ORDER BY COUNT(sp.PlaylistID) DESC) AS PopularityRank
FROM SongGenre sg
JOIN songs s ON sg.SongID = s.SongID
LEFT JOIN SongPlaylist sp ON s.SongID = sp.SongID
GROUP BY sg.GenreID, s.SongID
ORDER BY sg.GenreID, PopularityRank;
```



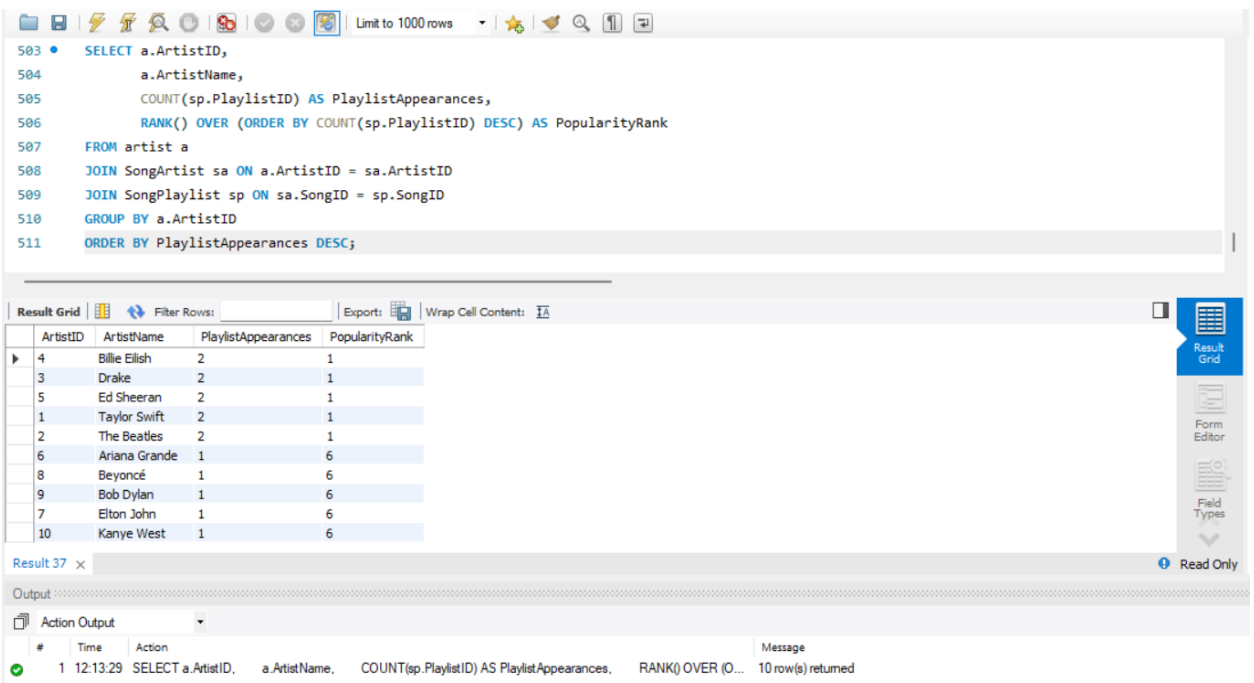
GenreID	SongID	SongTitle	PopularityRank
1	1	Cardigan	1
1	2	Come Together	1
1	11	Rolling in the Deep	1
2	4	Bad Guy	1
2	3	God's Plan	1
2	12	Thatâ€™s What I Like	1
3	6	7 rings	1
3	5	Shape of You	1
3	13	Stupid Love	1
4	8	Halo	1
4	14	Paint It Black	1
4	7	Tiny Dancer	1
5	9	Like a Rolling Stone	1

11) Artist Ranking by Number of Songs in Playlists

Explanation: This query approach ranks artists based on how frequently their songs are included in playlists, serving as a measure of their popularity and listener engagement within a music streaming platform. By counting the total number of playlist appearances for each artist's songs and then ranking the artists from most to least appearances.

Query:

```
SELECT a.ArtistID,
    a.ArtistName,
    COUNT(sp.PlaylistID) AS PlaylistAppearances,
    RANK() OVER (ORDER BY COUNT(sp.PlaylistID) DESC) AS PopularityRank
FROM artist a
JOIN SongArtist sa ON a.ArtistID = sa.ArtistID
JOIN SongPlaylist sp ON sa.SongID = sp.SongID
GROUP BY a.ArtistID
ORDER BY PlaylistAppearances DESC;
```



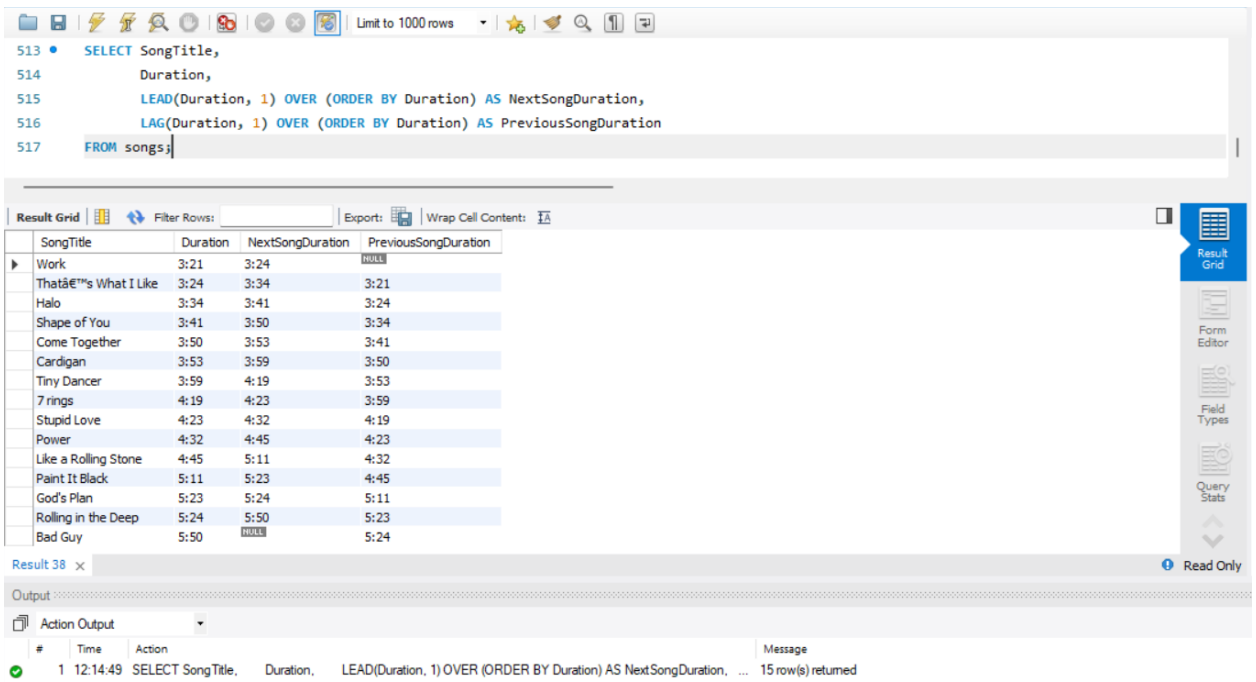
ArtistID	ArtistName	PlaylistAppearances	PopularityRank
4	Billie Eilish	2	1
3	Drake	2	1
5	Ed Sheeran	2	1
1	Taylor Swift	2	1
2	The Beatles	2	1
6	Ariana Grande	1	6
8	Beyoncé	1	6
9	Bob Dylan	1	6
7	Elton John	1	6
10	Kanye West	1	6

12) Next and Previous Song Duration for Each Song

Explanation: This query analysis involves comparing the duration of each song in a database with the durations of the song that comes immediately before (previous) and after (next) it when sorted by a specified criterion, such as release date or alphabetical order. By applying window functions like LEAD and LAG, one can retrieve the duration of adjacent songs, providing context for understanding a song's length relative to others in the collection.

Query:

```
SELECT SongTitle,
    Duration,
    LEAD(Duration, 1) OVER (ORDER BY Duration) AS NextSongDuration,
    LAG(Duration, 1) OVER (ORDER BY Duration) AS PreviousSongDuration
FROM songs;
```



SongTitle	Duration	NextSongDuration	PreviousSongDuration
Work	3:21	3:24	NULL
Thatâ€™s What I Like	3:24	3:34	3:21
Halo	3:34	3:41	3:24
Shape of You	3:41	3:50	3:34
Come Together	3:50	3:53	3:41
Cardigan	3:53	3:59	3:50
Tiny Dancer	3:59	4:19	3:53
7 rings	4:19	4:23	3:59
Stupid Love	4:23	4:32	4:19
Power	4:32	4:45	4:23
Like a Rolling Stone	4:45	5:11	4:32
Paint It Black	5:11	5:23	4:45
God's Plan	5:23	5:24	5:11
Rolling in the Deep	5:24	5:50	5:23
Bad Guy	5:50	NULL	5:24

Result 38 Read Only

Output

Action Output

#	Time	Action	Message
1	12:14:49	SELECT SongTitle, Duration, LEAD(Duration, 1) OVER (ORDER BY Duration) AS NextSongDuration, ...	15 row(s) returned

13) First and Last Album Released Each Year

Explanation: This query identifies the first and last albums released in each year within a music database. It involves sorting albums by their release dates for every year and then selecting the earliest and latest release as representatives of the year's musical bookends.

Query:

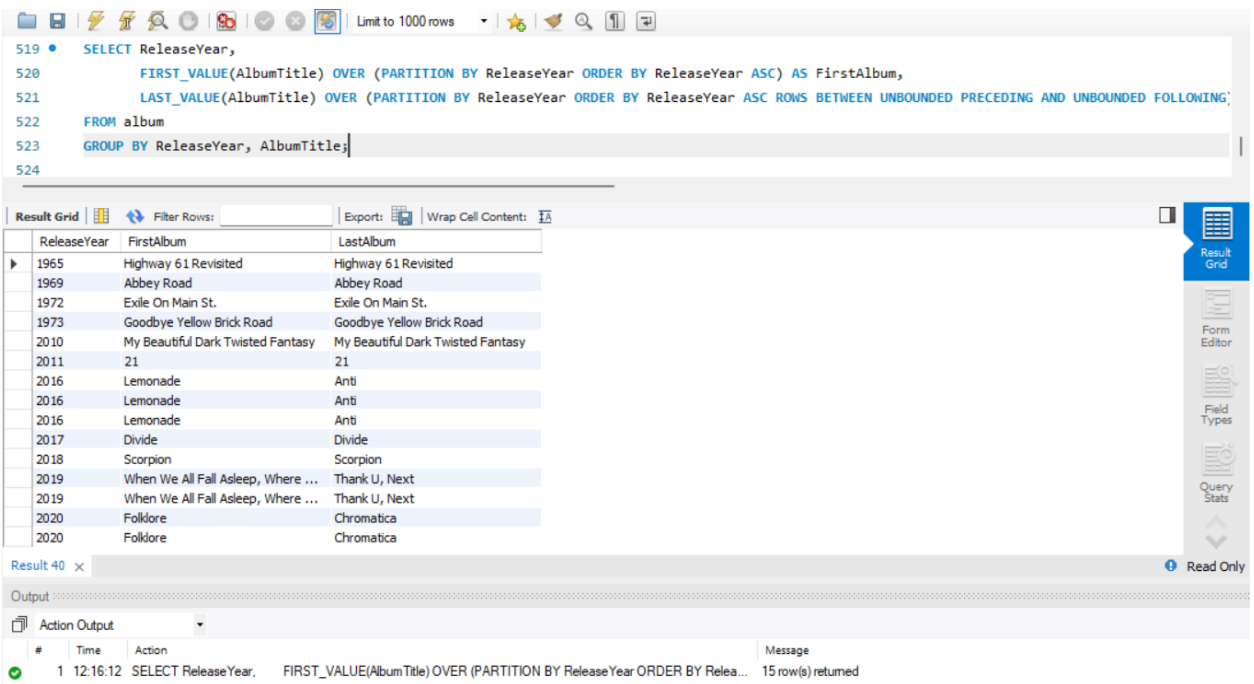
SELECT ReleaseYear,

FIRST_VALUE(AlbumTitle) OVER (PARTITION BY ReleaseYear ORDER BY ReleaseYear ASC) AS FirstAlbum,

LAST_VALUE(AlbumTitle) OVER (PARTITION BY ReleaseYear ORDER BY ReleaseYear ASC ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS LastAlbum

FROM album

GROUP BY ReleaseYear, AlbumTitle;



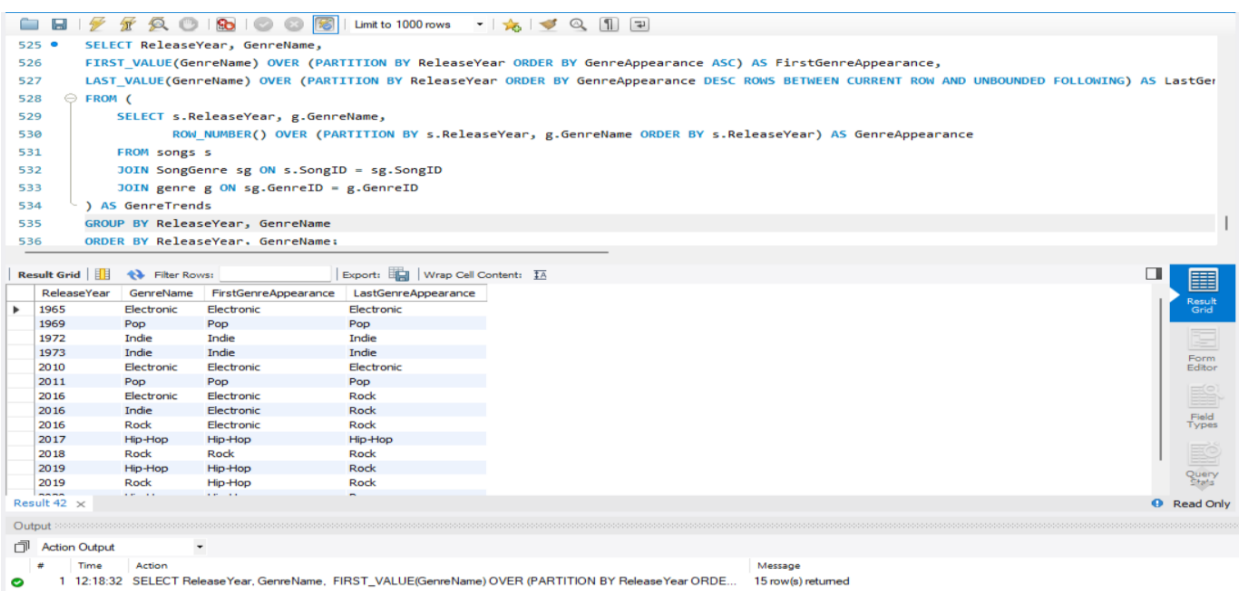
ReleaseYear	FirstAlbum	LastAlbum
1965	Highway 61 Revisited	Highway 61 Revisited
1969	Abbey Road	Abbey Road
1972	Exile On Main St.	Exile On Main St.
1973	Goodbye Yellow Brick Road	Goodbye Yellow Brick Road
2010	My Beautiful Dark Twisted Fantasy	My Beautiful Dark Twisted Fantasy
2011	21	21
2016	Lemonade	Anti
2016	Lemonade	Anti
2016	Lemonade	Anti
2017	Divide	Divide
2018	Scorpion	Scorpion
2019	When We All Fall Asleep, Where ...	Thank U, Next
2019	When We All Fall Asleep, Where ...	Thank U, Next
2020	Folklore	Chromatica
2020	Folklore	Chromatica

14) Analysis of Genre Trends Over Time

Explanation: This query involves examining how the popularity of different music genres has changed over time within a music database or industry at large. By tracking the number of albums or songs released in each genre annually, one can identify trends, such as the rise or decline of specific genres, shifts in musical tastes, and the emergence of new genres.

Query:

```
SELECT ReleaseYear, GenreName,

FIRST_VALUE(GenreName) OVER (PARTITION BY ReleaseYear ORDER BY GenreAppearance ASC) AS FirstGenreAppearance,

LAST_VALUE(GenreName) OVER (PARTITION BY ReleaseYear ORDER BY GenreAppearance DESC ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) AS LastGenreAppearance

FROM (

   SELECT s.ReleaseYear, g.GenreName,

          ROW_NUMBER() OVER (PARTITION BY s.ReleaseYear, g.GenreName ORDER BY s.ReleaseYear) AS GenreAppearance

   FROM songs s

   JOIN SongGenre sg ON s.SongID = sg.SongID

   JOIN genre g ON sg.GenreID = g.GenreID

) AS GenreTrends

GROUP BY ReleaseYear, GenreName

ORDER BY ReleaseYear, GenreName;
```

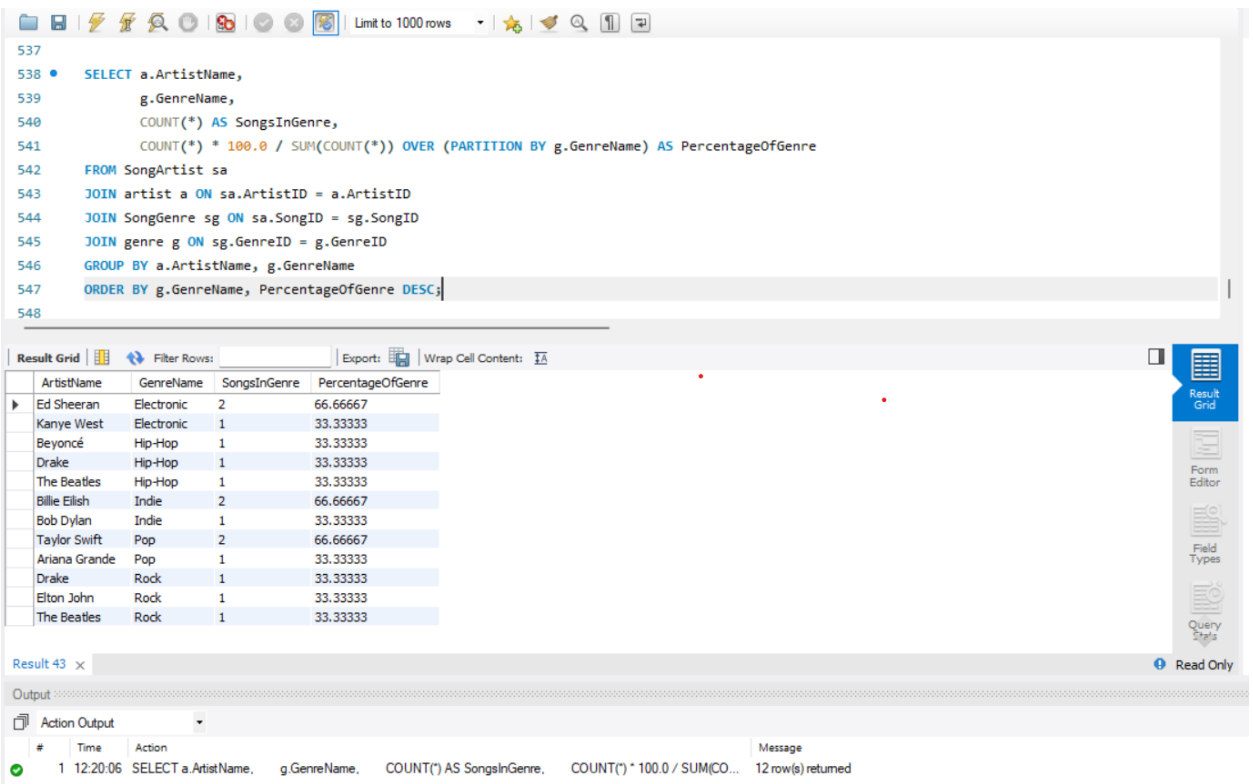


15) Artist Contribution to Genres

Explanation: This query analysis quantifies and highlights the contribution of individual artists to specific music genres. It involves counting the number of songs or albums an artist has released within each genre, thereby measuring their influence or dominance in those categories.

Query:

```
SELECT a.ArtistName,
    g.GenreName,
    COUNT(*) AS SongsInGenre,
    COUNT(*) * 100.0 / SUM(COUNT(*)) OVER (PARTITION BY g.GenreName) AS PercentageOfGenre
FROM SongArtist sa
JOIN artist a ON sa.ArtistID = a.ArtistID
JOIN SongGenre sg ON sa.SongID = sg.SongID
JOIN genre g ON sg.GenreID = g.GenreID
GROUP BY a.ArtistName, g.GenreName
ORDER BY g.GenreName, PercentageOfGenre DESC;
```



ArtistName	GenreName	SongsInGenre	PercentageOfGenre
Ed Sheeran	Electronic	2	66.66667
Kanye West	Electronic	1	33.33333
Beyoncé	Hip-Hop	1	33.33333
Drake	Hip-Hop	1	33.33333
The Beatles	Hip-Hop	1	33.33333
Billie Eilish	Indie	2	66.66667
Bob Dylan	Indie	1	33.33333
Taylor Swift	Pop	2	66.66667
Ariana Grande	Pop	1	33.33333
Drake	Rock	1	33.33333
Elton John	Rock	1	33.33333
The Beatles	Rock	1	33.33333