IFT 530: Initial Proposal

Music Library Database Management System

PabbathiReddy Pavan

Sai Poornima lingala

Ruchitha Juturu

Komal Bandi

Dr. Robert Rucker

30 April,2023

Table of Contents

Synopsis

The objective of Music Library Database Management System is to design and develop a music library platform that provides all listeners, regardless of ability level, with a wide range of comprehensive features. There is lots of potential to meet all types of diverse user wants because the music industry dominates the entertainment industry and has a global user base. Users of our site have access to features including recommendations for suitable music depending on their location, preferred language, favorite genres, and current mood. Moreover, users can locate songs based on the very minimum of information they can recall, like: For instance: publishing date, artist, or text fragment.Many entities, including tracks, albums, artists, bands, users, device players, playlists, and more are stored in the database.

We think there is always potential for new features to be offered to the platform given the thriving music industry and the evergreen and rising consumer base. By just humming the music or singing the words, users may now employ new machine learning capabilities to find certain songs. We create a flexible, adjustable music library database management system that finally includes a broad range of potent capabilities.

Area And Topic of Interest

The music industry is the one in which we are employed. Since its founding 200 years ago, the music industry has experienced exponential growth. There are many different players in the music industry, from the real musicians who play the tracks to the massive production corporations that support and enable them, the technical personnel that assist them, and the fans who enjoy them. That is In the past ten years, the music business has seen significant upheaval. Record sales decreased as music listening habits drastically changed. Today's musicians mainly rely on social media and streaming services to market their music, and many are succeeding in this brand-new landscape.

You may listen to any of your chosen content anytime you want thanks to digital music streaming services, which provide you immediate access to their vast online music collection. Using it is easy. There are millions of music accessible from numerous genres and artists. There are many different types of music, such as pop, jazz, hip-hop, rock, and classical. There is music on computers, smartphones, tablets, TVs, automobiles, watches, and more.

Importance / Interest to us in this Area

By definition, music is a type of sound art that effectively conveys ideas and emotions through rhythm, melody, harmony, and color. As music is an enduring beauty, I think everyone can enjoy it. Because we have always loved music, it comes naturally to us. We adore how diverse music can be, just like humans. Like to music, not everyone has the same personalities, tastes, or preferences. Some people like rock; others don't. A person's musical preferences can frequently be used to describe them.

Music is created and sold for profit by businesses and individuals. Band members perform live shows and go on tour to get extra cash. Recorded music is produced and promoted by record labels, publishers, producers, engineers, and other businesses. Music is joining the digital realm, improving its viability and accessibility.

Digital music, which combines technology and our love of the music business, is where our interests truly converge. Before deciding that Music Library Management System would be our passion project, all we needed to do was have a brief conversation and iron out the details.

Entities and Attributes

Structuring the data requirements begins by identifying the relevant data and the entities contained within. The project's data is made up of artists, bands, songs, and their labels. An initial set of database entities has been outlined below as a foundation for later modeling.

ENTITY NAME	ATTRIBUTES
Track	Track Id
	Title
	Duration
	Genre
	Language
Album	Album Id
	Title
	Production Company
	Region
	Number of Songs
Artist	Artist Id

	Name
	Band
User	User Id
	Username
	Password
	Location
	Preferred Language
	Preferred Genre
Playlist	Name
	Number of Songs
	Created On
Device Player	Device Id
	Type
	isShuffle
	Volume
	Registered On

Questions

1. What is the length of the track?

2. How many albums were made available in 2021?

3. What genre belongs to this song?

4. Include each song's artist name, band name, and album name.

5. Locate every track name that begins with the letter "A"

6. How frequently was this song played on the type of mobile device?

7. Determine the song's title that appeared the most frequently.

8. What is this song's album region?

9. Locate the track with the least duration

10. List every musician that has ever recorded a song in this genre.

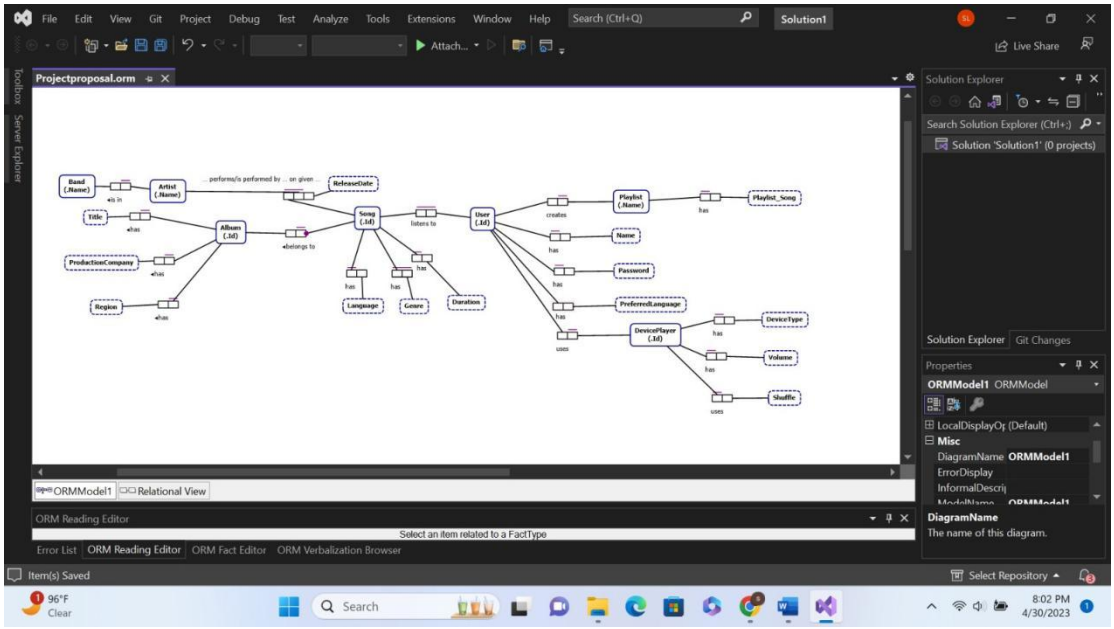
11.What information is retrieved from the "Album" bucket where the ID is "album1"?

12.What are the playlists created after January 1st, 2022?

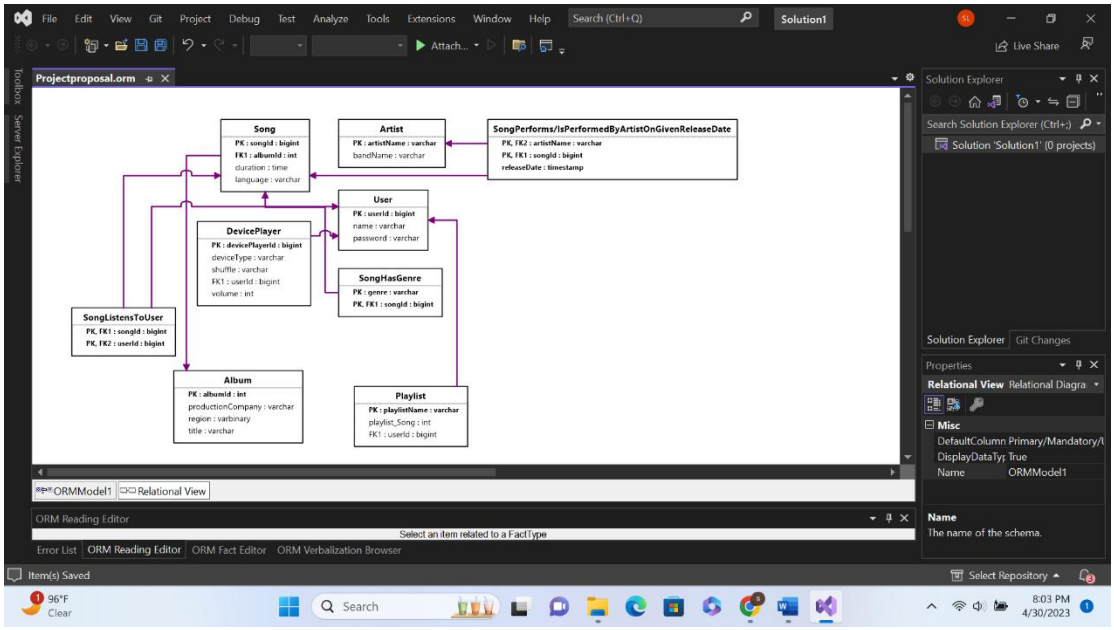
13.What is the minimum number of songs required for a row to be selected?

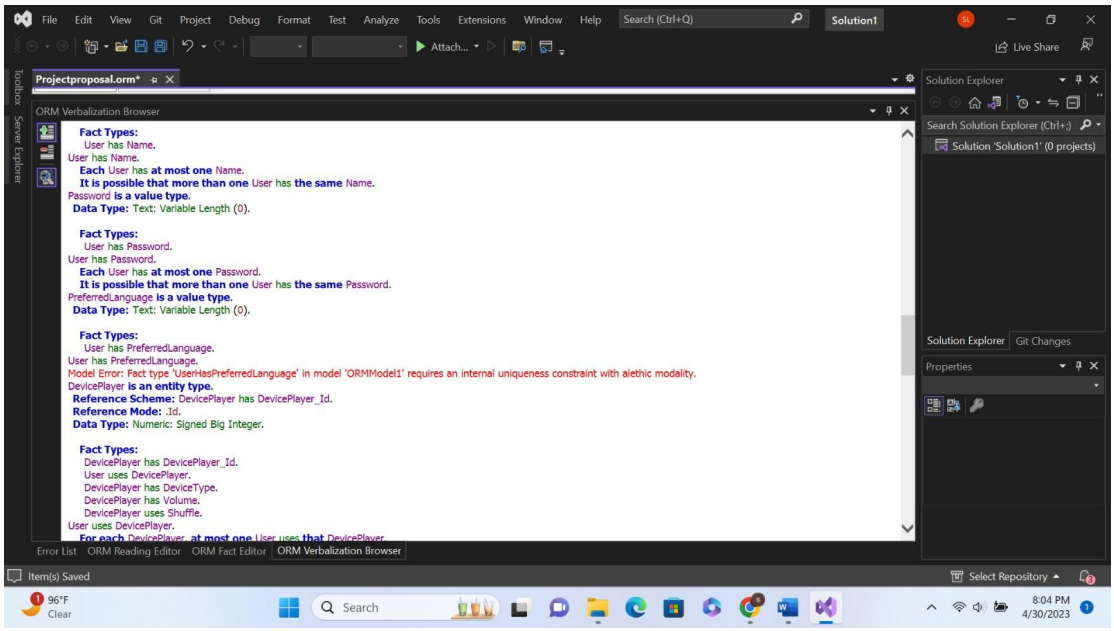
14.What is the total duration of all songs in the albums that satisfy the condition "numberOfSongs > 9 AND productionCompany = 'Golden valley'"?

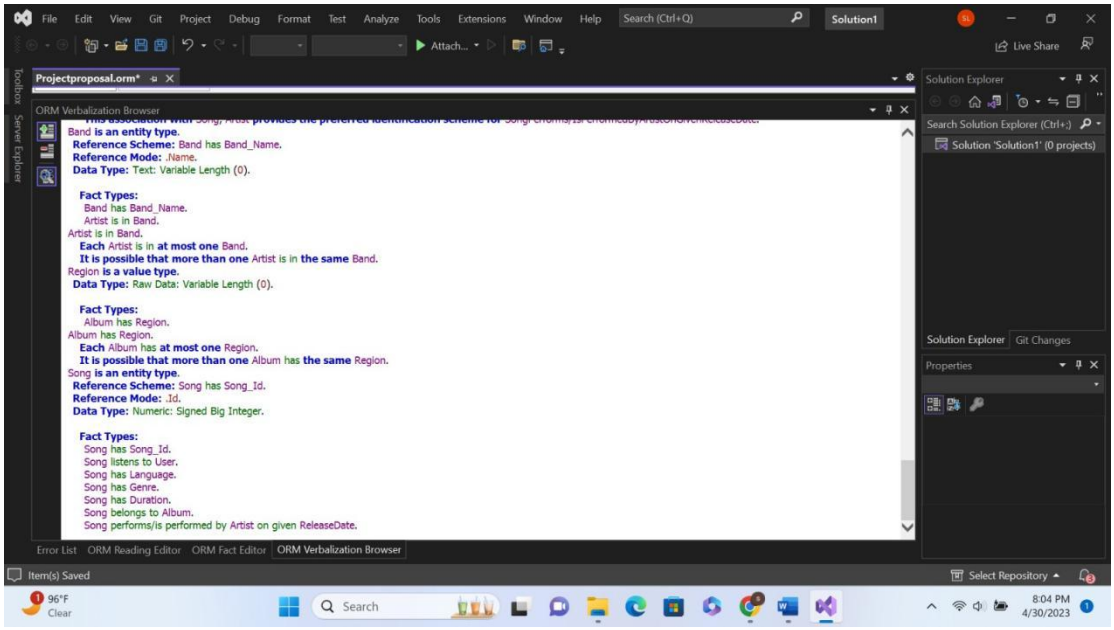
ORM Diagram

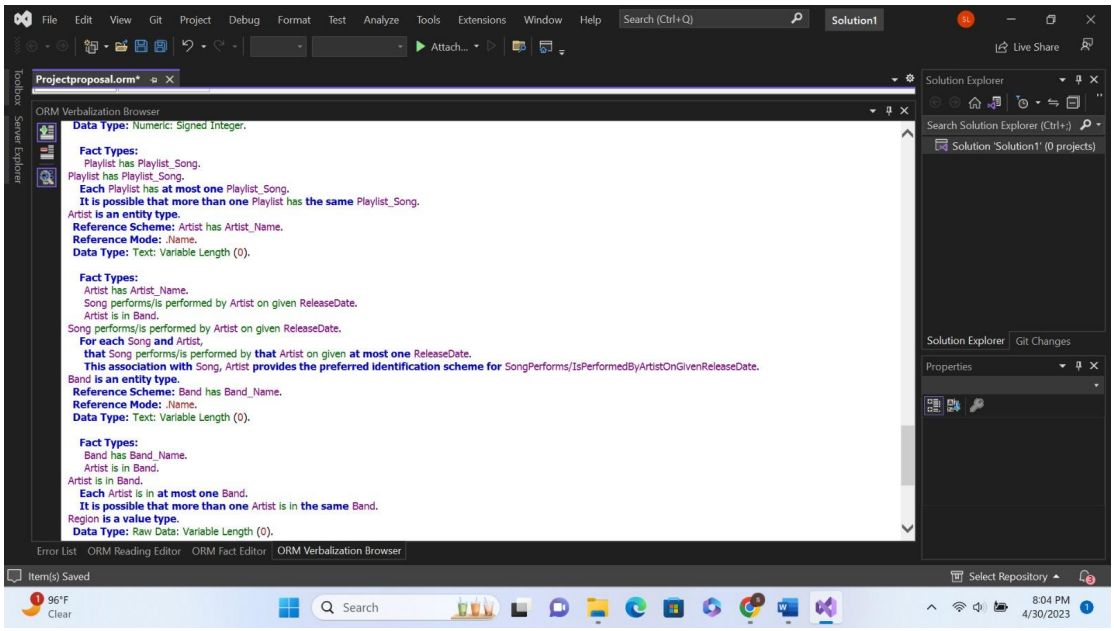


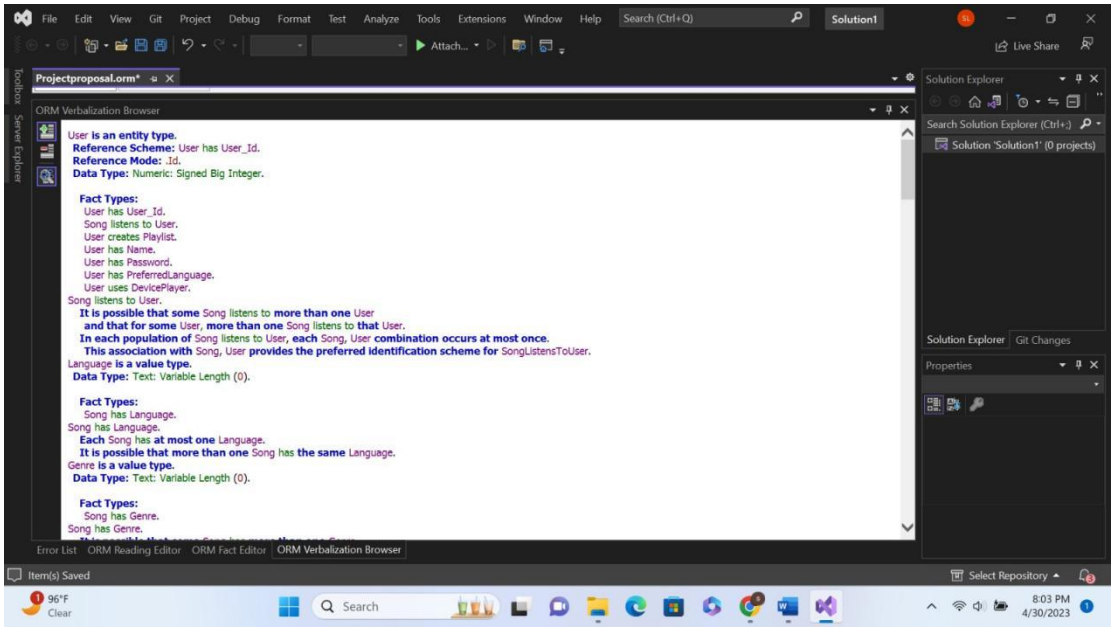
Relational View

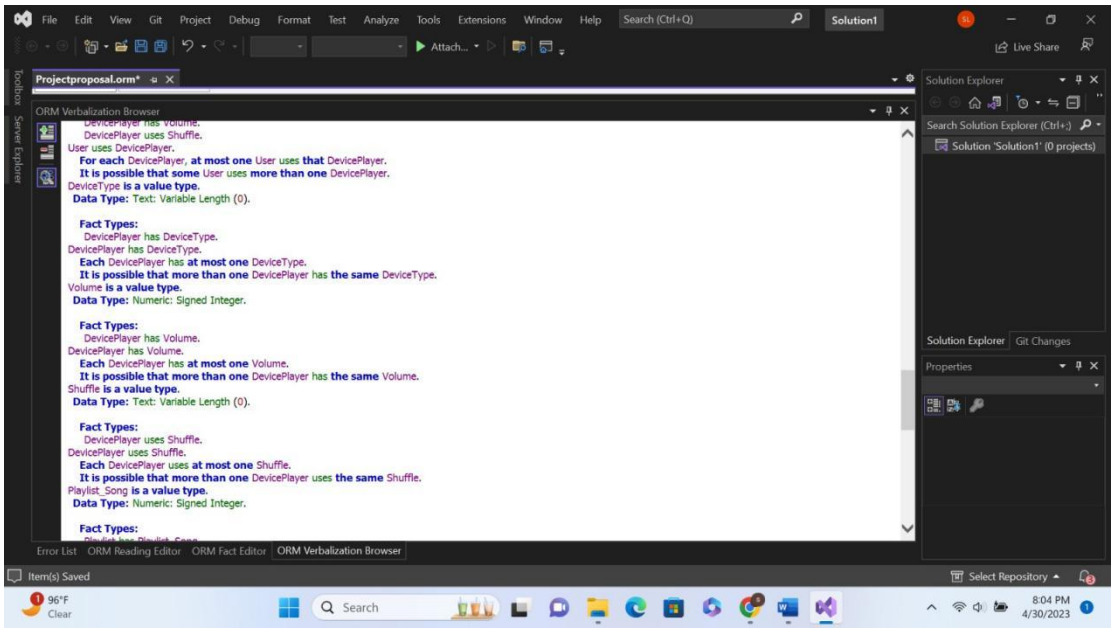


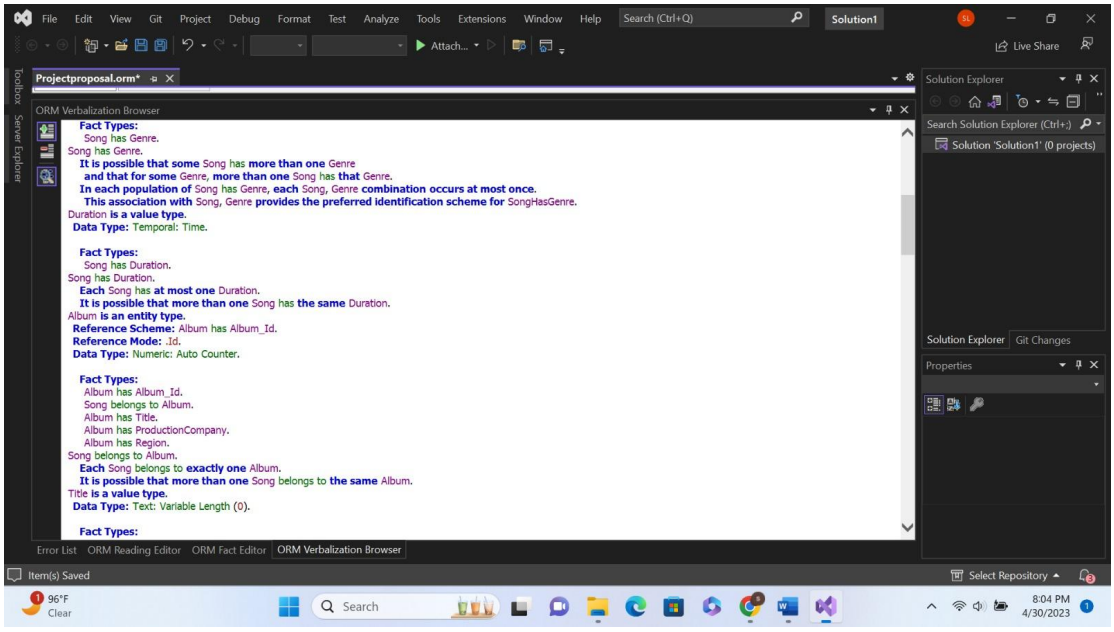

File Edit View Git Project Debug Format Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Solution1
Live Share
Attach...
Projectproposal.orm*
ORM Verbalization Browser
Fact Types:
User has Name.
User has Name.
Each User has at most one Name.
It is possible that more than one User has the same Name.
Password is a value type.
Data Type: Text: Variable Length (0).
Fact Types:
User has Password.
User has Password.
Each User has at most one Password.
It is possible that more than one User has the same Password.
PreferredLanguage is a value type.
Data Type: Text: Variable Length (0).
Fact Types:
User has PreferredLanguage.
User has PreferredLanguage.
Model Error: Fact type 'UserHasPreferredLanguage' in model 'ORMModel1' requires an internal uniqueness constraint with alethic modality.
DevicePlayer is an entity type.
Reference Scheme: DevicePlayer has DevicePlayer_Id.
Reference Mode: .Id.
Data Type: Numeric: Signed Big Integer.
Fact Types:
DevicePlayer has DevicePlayer_Id.
User uses DevicePlayer.
DevicePlayer has DeviceType.
DevicePlayer has Volume.
DevicePlayer uses Shuffle.
User uses DevicePlayer.
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Solution1' (0 projects)
Solution Explorer Git Changes
Properties
Error List ORM Reading Editor ORM Fact Editor ORM Verbalization Browser
Item(s) Saved
Select Repository
96°F Clear
Search
8:04 PM 4/30/2023

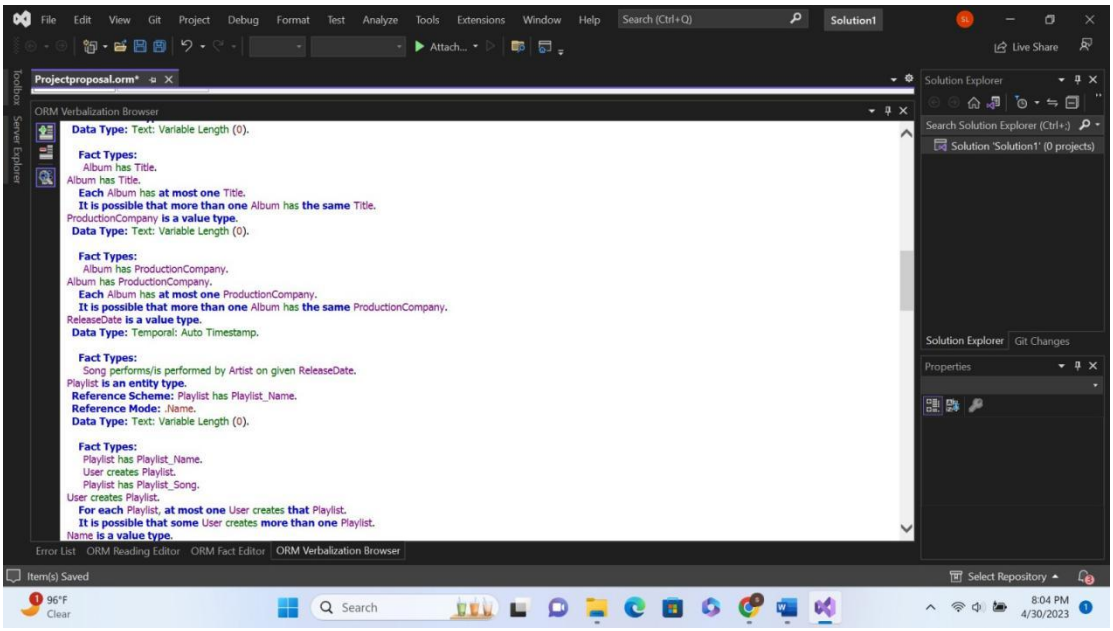

File Edit View Git Project Debug Format Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Solution1
Live Share
Attach...
Projectproposal.orm*
ORM Verbalization Browser
Band is an entity type.
Reference Scheme: Band has Band_Name.
Reference Mode: .Name.
Data Type: Text: Variable Length (0).
Fact Types:
Band has Band_Name.
Artist is in Band.
Artist is in Band.
Each Artist is in at most one Band.
It is possible that more than one Artist is in the same Band.
Region is a value type.
Data Type: Raw Data: Variable Length (0).
Fact Types:
Album has Region.
Album has Region.
Each Album has at most one Region.
It is possible that more than one Album has the same Region.
Song is an entity type.
Reference Scheme: Song has Song_Id.
Reference Mode: .Id.
Data Type: Numeric: Signed Big Integer.
Fact Types:
Song has Song_Id.
Song listens to User.
Song has Language.
Song has Genre.
Song has Duration.
Song belongs to Album.
Song performs/is performed by Artist on given ReleaseDate.
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Solution1' (0 projects)
Solution Explorer Git Changes
Properties
Error List ORM Reading Editor ORM Fact Editor ORM Verbalization Browser
Item(s) Saved
Select Repository
96°F Clear
Search
8:04 PM 4/30/2023


Projectproposal.orm*
ORM Verbalization Browser
Data Type: Numeric: Signed Integer.
Fact Types:
Playlist has Playlist_Song.
Playlist has Playlist_Song.
Each Playlist has at most one Playlist_Song.
It is possible that more than one Playlist has the same Playlist_Song.
Artist is an entity type.
Reference Scheme: Artist has Artist_Name.
Reference Mode: .Name.
Data Type: Text: Variable Length (0).
Fact Types:
Artist has Artist_Name.
Song performs/is performed by Artist on given ReleaseDate.
Artist is in Band.
Song performs/is performed by Artist on given ReleaseDate.
For each Song and Artist,
that Song performs/is performed by that Artist on given at most one ReleaseDate.
This association with Song, Artist provides the preferred identification scheme for SongPerforms/IsPerformedByArtistOnGivenReleaseDate.
Band is an entity type.
Reference Scheme: Band has Band_Name.
Reference Mode: .Name.
Data Type: Text: Variable Length (0).
Fact Types:
Band has Band_Name.
Artist is in Band.
Artist is in Band.
Each Artist is in at most one Band.
It is possible that more than one Artist is in the same Band.
Region is a value type.
Data Type: Raw Data: Variable Length (0).
Error List
ORM Reading Editor
ORM Fact Editor
ORM Verbalization Browser
Item(s) Saved
Select Repository
Solution Explorer
Git Changes
Properties
8:04 PM
4/30/2023


Projectproposal.orm*
ORM Verbalization Browser
User is an entity type.
Reference Scheme: User has User_Id.
Reference Mode: .Id.
Data Type: Numeric: Signed Big Integer.
Fact Types:
User has User_Id.
Song listens to User.
User creates Playlist.
User has Name.
User has Password.
User has PreferredLanguage.
User uses DevicePlayer.
Song listens to User.
It is possible that some Song listens to more than one User
and that for some User, more than one Song listens to that User.
In each population of Song listens to User, each Song, User combination occurs at most once.
This association with Song, User provides the preferred identification scheme for SongListensToUser.
Language is a value type.
Data Type: Text: Variable Length (0).
Fact Types:
Song has Language.
Song has Language.
Each Song has at most one Language.
It is possible that more than one Song has the same Language.
Genre is a value type.
Data Type: Text: Variable Length (0).
Fact Types:
Song has Genre.
Song has Genre.
Error List
ORM Reading Editor
ORM Fact Editor
ORM Verbalization Browser
Item(s) Saved
Select Repository
Solution Explorer
Git Changes
Properties
8:03 PM
4/30/2023


Projectproposal.orm*
ORM Verbalization Browser
DevicePlayer has Volume.
DevicePlayer uses Shuffle.
User uses DevicePlayer.
For each DevicePlayer, at most one User uses that DevicePlayer.
It is possible that some User uses more than one DevicePlayer.
DeviceType is a value type.
Data Type: Text: Variable Length (0).
Fact Types:
DevicePlayer has DeviceType.
DevicePlayer has DeviceType.
Each DevicePlayer has at most one DeviceType.
It is possible that more than one DevicePlayer has the same DeviceType.
Volume is a value type.
Data Type: Numeric: Signed Integer.
Fact Types:
DevicePlayer has Volume.
DevicePlayer has Volume.
Each DevicePlayer has at most one Volume.
It is possible that more than one DevicePlayer has the same Volume.
Shuffle is a value type.
Data Type: Text: Variable Length (0).
Fact Types:
DevicePlayer uses Shuffle.
DevicePlayer uses Shuffle.
Each DevicePlayer uses at most one Shuffle.
It is possible that more than one DevicePlayer uses the same Shuffle.
Playlist_Song is a value type.
Data Type: Numeric: Signed Integer.
Fact Types:
Error List
ORM Reading Editor
ORM Fact Editor
ORM Verbalization Browser
Item(s) Saved
Select Repository
Solution Explorer
Git Changes
Properties
8:04 PM
4/30/2023


Projectproposal.orm*
ORM Verbalization Browser
Fact Types:
Song has Genre.
Song has Genre.
It is possible that some Song has more than one Genre
and that for some Genre, more than one Song has that Genre.
In each population of Song has Genre, each Song, Genre combination occurs at most once.
This association with Song, Genre provides the preferred identification scheme for SongHasGenre.
Duration is a value type.
Data Type: Temporal: Time.
Fact Types:
Song has Duration.
Song has Duration.
Each Song has at most one Duration.
It is possible that more than one Song has the same Duration.
Album is an entity type.
Reference Scheme: Album has Album_Id.
Reference Mode: .Id.
Data Type: Numeric: Auto Counter.
Fact Types:
Album has Album_Id.
Song belongs to Album.
Album has Title.
Album has ProductionCompany.
Album has Region.
Song belongs to Album.
Each Song belongs to exactly one Album.
It is possible that more than one Song belongs to the same Album.
Title is a value type.
Data Type: Text: Variable Length (0).
Fact Types:
Error List
ORM Reading Editor
ORM Fact Editor
ORM Verbalization Browser
Item(s) Saved
Select Repository
Solution Explorer
Git Changes
Properties
8:04 PM
4/30/2023

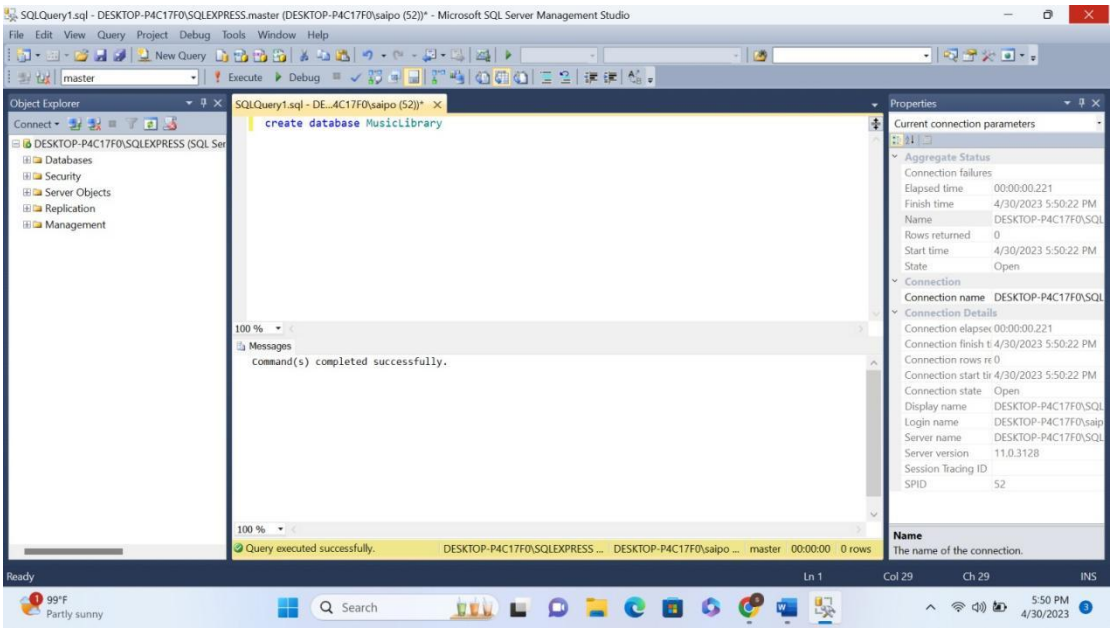


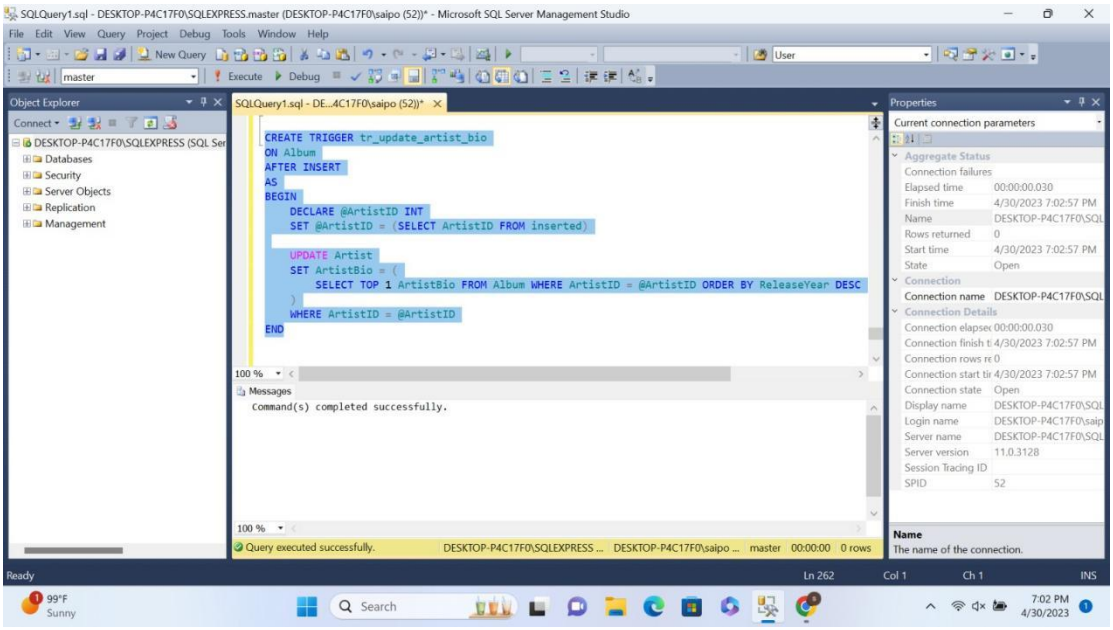
Tables:

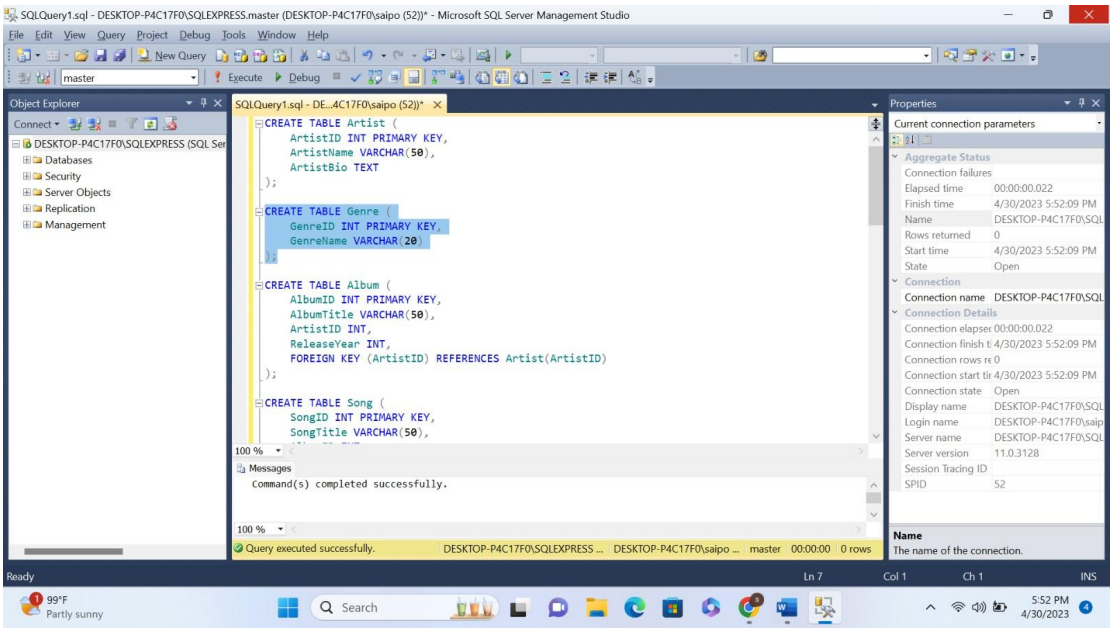
```
CREATE TABLE Artist (
    ArtistID INT PRIMARY KEY,
    ArtistName VARCHAR(50),
    ArtistBio TEXT
);

CREATE TABLE Genre (
    GenreID INT PRIMARY KEY,
    GenreName VARCHAR(20)
);

CREATE TABLE Album (
    AlbumID INT PRIMARY KEY,
```

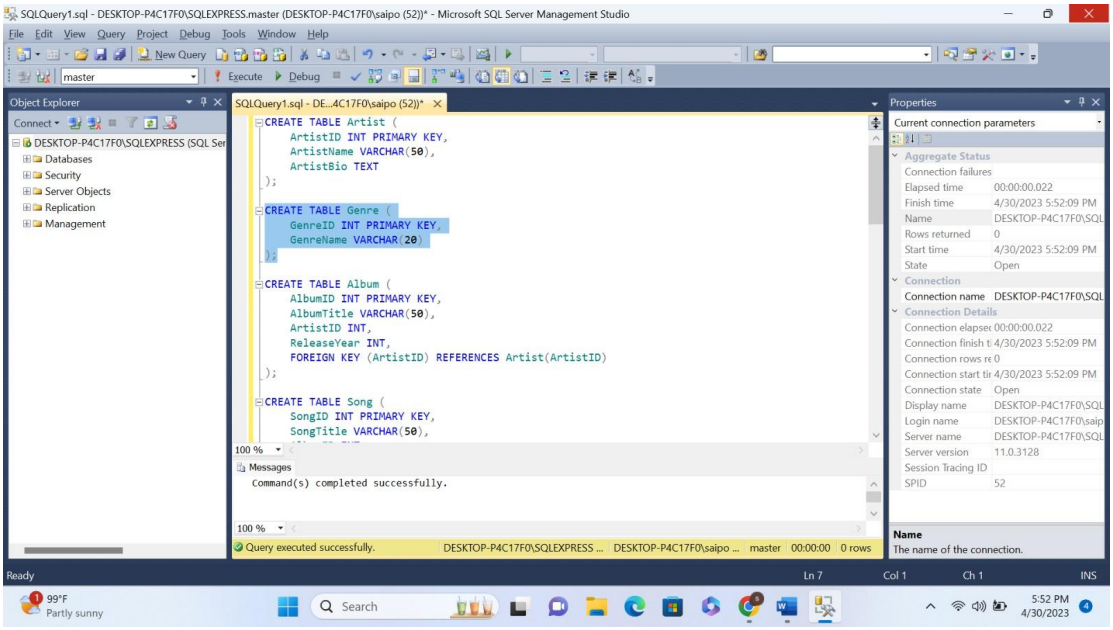
```
    AlbumTitle VARCHAR(50),
    ArtistID INT,
    ReleaseYear INT,
    FOREIGN KEY (ArtistID) REFERENCES Artist(ArtistID)
);

CREATE TABLE Song (
    SongID INT PRIMARY KEY,
    SongTitle VARCHAR(50),
    AlbumID INT,
    GenreID INT,
    FOREIGN KEY (AlbumID) REFERENCES Album(AlbumID),
    FOREIGN KEY (GenreID) REFERENCES Genre(GenreID)
);

CREATE TABLE Playlist (
    PlaylistID INT PRIMARY KEY,
    PlaylistTitle VARCHAR(50),
    UserID INT,
    FOREIGN KEY (UserID) REFERENCES EndUser(UserID)
);

CREATE TABLE Playlist_Song (
    PlaylistID INT,
    SongID INT,
```

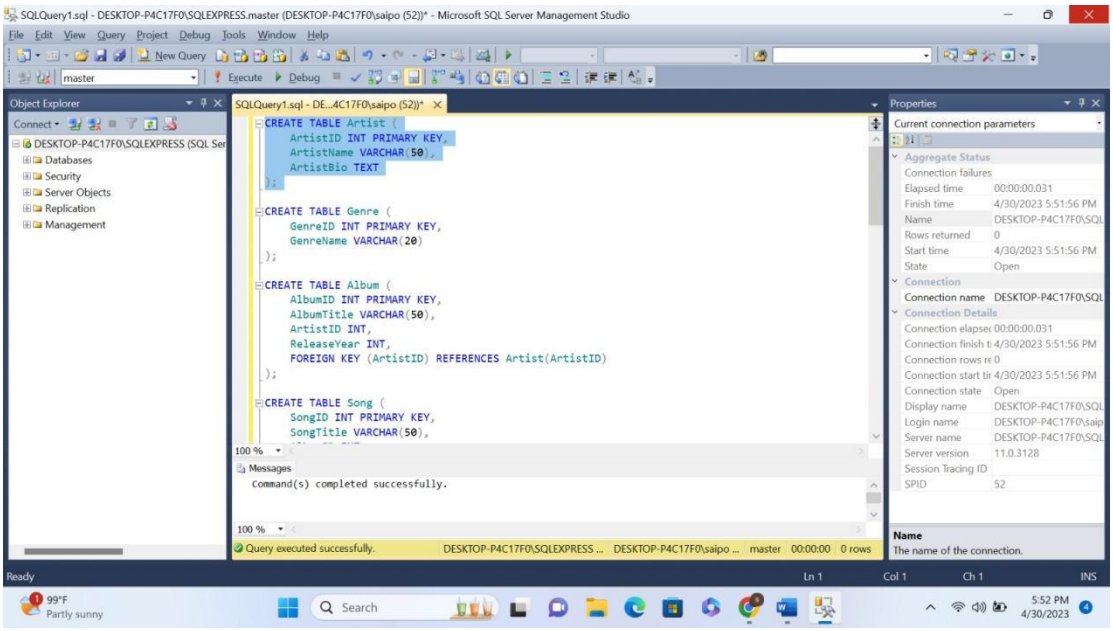
```
    PRIMARY KEY (PlaylistID, SongID),
    FOREIGN KEY (PlaylistID) REFERENCES Playlist(PlaylistID),
    FOREIGN KEY (SongID) REFERENCES Song(SongID)
);

CREATE TABLE EndUser (
    UserID INT PRIMARY KEY,
    UserName VARCHAR(50),
    Email VARCHAR(50),
    Password VARCHAR(50)
);

CREATE TABLE User_Playlist (
    UserID INT,
    PlaylistID INT,
    PRIMARY KEY (UserID, PlaylistID),
    FOREIGN KEY (UserID) REFERENCES EndUser(UserID),
    FOREIGN KEY (PlaylistID) REFERENCES Playlist(PlaylistID)
```

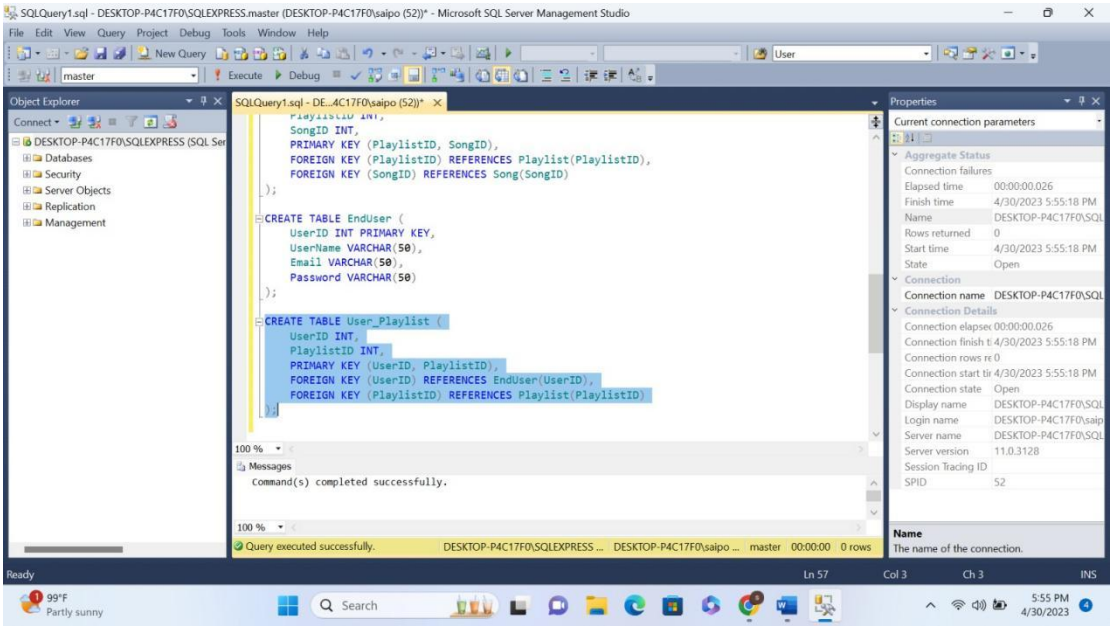

SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query
Execute Debug
master
Object Explorer
Connect
DESKTOP-P4C17F0\SQLEXPRESS (SQL Ser
Databases
Security
Server Objects
Replication
Management
SQLQuery1.sql - DE...4C17F0\saipo (52))*
create database MusicLibrary
100 %
Messages
Command(s) completed successfully.
Query executed successfully.
DESKTOP-P4C17F0\SQLEXPRESS ...
DESKTOP-P4C17F0\saipo ...
master
00:00:00
0 rows
Properties
Current connection parameters
Aggregate Status
Connection failures
Elapsed time 00:00:00.221
Finish time 4/30/2023 5:50:22 PM
Rows returned 0
Start time 4/30/2023 5:50:22 PM
State Open
Connection
Connection name
Connection Details
Server name
Server version 11.0.3128
Session Tracing ID
SPID 52
Name
The name of the connection.
Ready
Ln 1
Col 29
Ch 29
INS
99°F
Partly sunny
Search
5:50 PM
4/30/2023

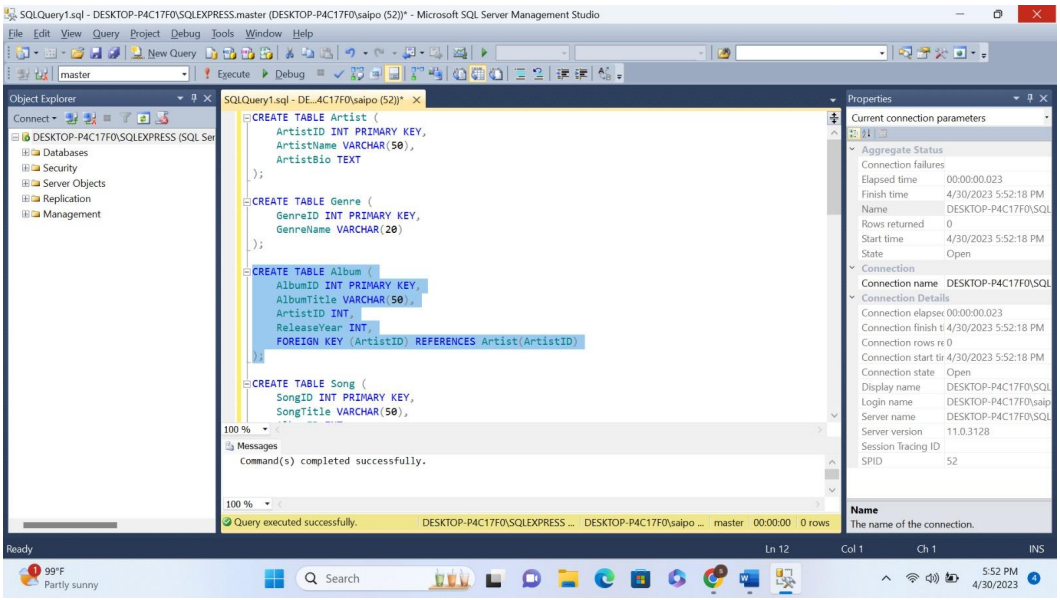

SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query
User
Execute Debug
master
Object Explorer
Connect
DESKTOP-P4C17F0\SQLEXPRESS (SQL Ser
Databases
Security
Server Objects
Replication
Management
SQLQuery1.sql - DE...4C17F0\saipo (52))*
CREATE TRIGGER tr_update_artist_bio
ON Album
AFTER INSERT
AS
BEGIN
DECLARE @ArtistID INT
SET @ArtistID = (SELECT ArtistID FROM inserted)
UPDATE Artist
SET ArtistBio = (
SELECT TOP 1 ArtistBio FROM Album WHERE ArtistID = @ArtistID ORDER BY ReleaseYear DESC
)
WHERE ArtistID = @ArtistID
END
100 %
Messages
Command(s) completed successfully.
Query executed successfully.
DESKTOP-P4C17F0\SQLEXPRESS ...
DESKTOP-P4C17F0\saipo ...
master
00:00:00
0 rows
Properties
Current connection parameters
Aggregate Status
Connection failures
Elapsed time 00:00:00.030
Finish time 4/30/2023 7:02:57 PM
Rows returned 0
Start time 4/30/2023 7:02:57 PM
State Open
Connection
Connection name
Connection Details
Server name
Server version 11.0.3128
Session Tracing ID
SPID 52
Name
The name of the connection.
Ready
Ln 262
Col 1
Ch 1
INS
99°F
Sunny
Search
7:02 PM
4/30/2023

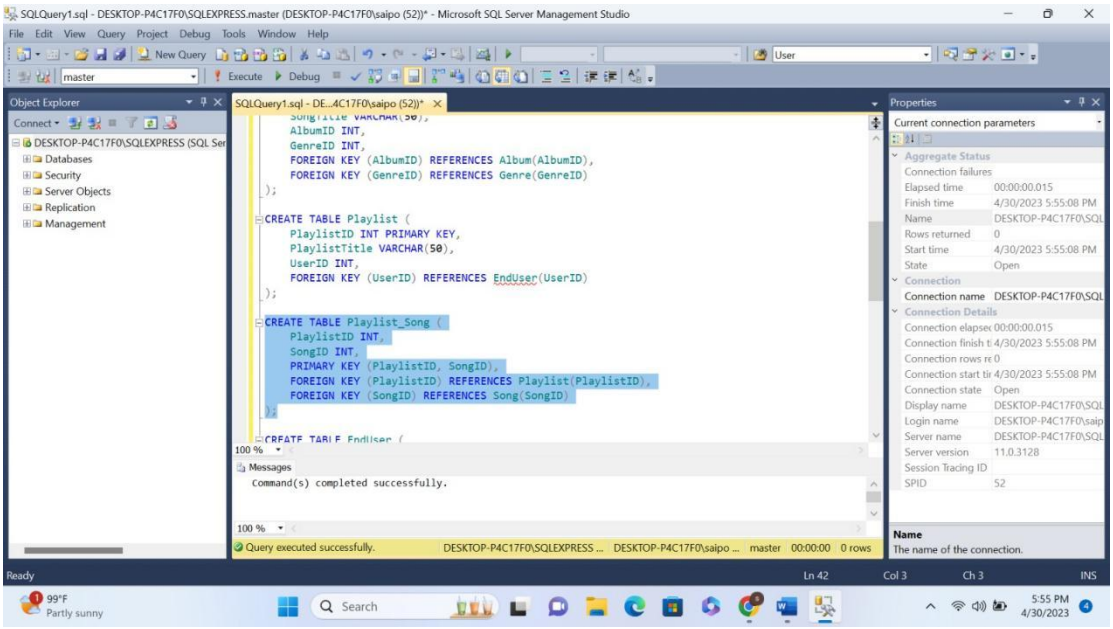

SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query
Execute Debug
master
Object Explorer
Connect
DESKTOP-P4C17F0\SQLEXPRESS (SQL Ser
Databases
Security
Server Objects
Replication
Management
SQLQuery1.sql - DE...4C17F0\saipo (52))*
CREATE TABLE Artist (
ArtistID INT PRIMARY KEY,
ArtistName VARCHAR(50),
ArtistBio TEXT
);
CREATE TABLE Genre (
GenreID INT PRIMARY KEY,
GenreName VARCHAR(20)
);
CREATE TABLE Album (
AlbumID INT PRIMARY KEY,
AlbumTitle VARCHAR(50),
ArtistID INT,
ReleaseYear INT,
FOREIGN KEY (ArtistID) REFERENCES Artist(ArtistID)
);
CREATE TABLE Song (
SongID INT PRIMARY KEY,
SongTitle VARCHAR(50),
100 %
Messages
Command(s) completed successfully.
100 %
Query executed successfully.
DESKTOP-P4C17F0\SQLEXPRESS ...
DESKTOP-P4C17F0\saipo ...
master
00:00:00
0 rows
Properties
Current connection parameters
Aggregate Status
Connection failures
Elapsed time 00:00:00.022
Finish time 4/30/2023 5:52:09 PM
Name DESKTOP-P4C17F0\SQL
Rows returned 0
Start time 4/30/2023 5:52:09 PM
State Open
Connection
Connection name DESKTOP-P4C17F0\SQL
Connection Details
Connection elapse 00:00:00.022
Connection finish t 4/30/2023 5:52:09 PM
Connection rows re 0
Connection start ti 4/30/2023 5:52:09 PM
Connection state Open
Display name DESKTOP-P4C17F0\SQL
Login name DESKTOP-P4C17F0\saip
Server name DESKTOP-P4C17F0\SQL
Server version 11.0.3128
Session Tracing ID
SPID 52
Name
The name of the connection.
Ready
Ln 7
Col 1
Ch 1
INS
99°F
Partly sunny
Search
5:52 PM
4/30/2023

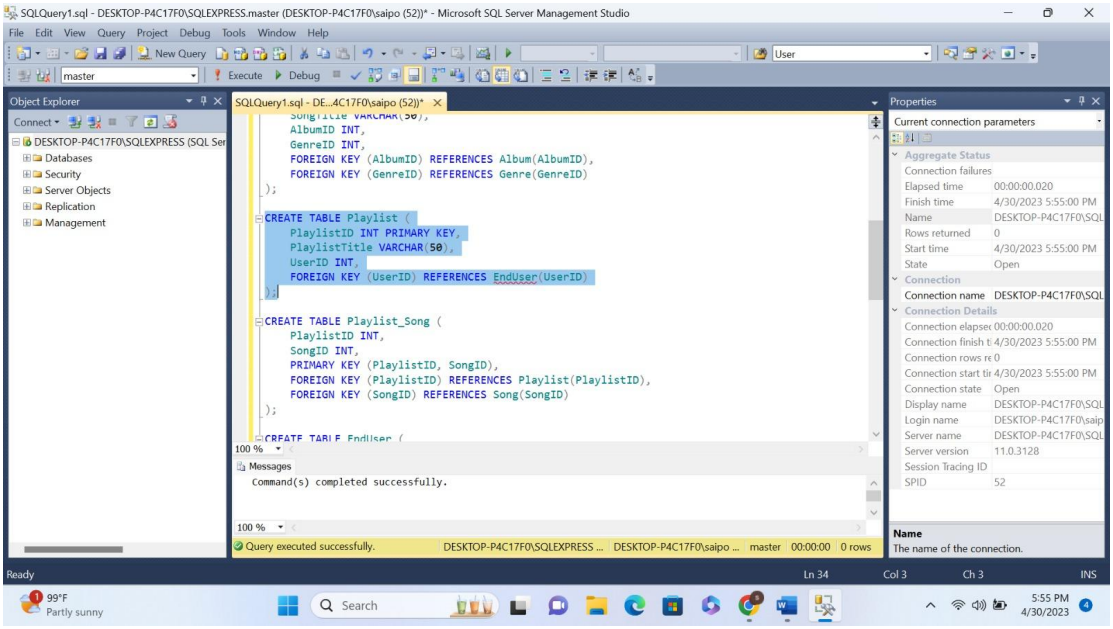

SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query
Execute Debug
master
Object Explorer
Connect
DESKTOP-P4C17F0\SQLEXPRESS (SQL Ser
Databases
Security
Server Objects
Replication
Management
SQLQuery1.sql - DE...4C17F0\saipo (52))*
CREATE TABLE Artist (
ArtistID INT PRIMARY KEY,
ArtistName VARCHAR(50),
ArtistBio TEXT
);
CREATE TABLE Genre (
GenreID INT PRIMARY KEY,
GenreName VARCHAR(20)
);
CREATE TABLE Album (
AlbumID INT PRIMARY KEY,
AlbumTitle VARCHAR(50),
ArtistID INT,
ReleaseYear INT,
FOREIGN KEY (ArtistID) REFERENCES Artist(ArtistID)
);
CREATE TABLE Song (
SongID INT PRIMARY KEY,
SongTitle VARCHAR(50),
100 %
Messages
Command(s) completed successfully.
100 %
Query executed successfully.
DESKTOP-P4C17F0\SQLEXPRESS ...
DESKTOP-P4C17F0\saipo ...
master
00:00:00
0 rows
Properties
Current connection parameters
Aggregate Status
Connection failures
Elapsed time 00:00:00.022
Finish time 4/30/2023 5:52:09 PM
Name DESKTOP-P4C17F0\SQL
Rows returned 0
Start time 4/30/2023 5:52:09 PM
State Open
Connection
Connection name DESKTOP-P4C17F0\SQL
Connection Details
Connection elapse 00:00:00.022
Connection finish t 4/30/2023 5:52:09 PM
Connection rows re 0
Connection start ti 4/30/2023 5:52:09 PM
Connection state Open
Display name DESKTOP-P4C17F0\SQL
Login name DESKTOP-P4C17F0\saip
Server name DESKTOP-P4C17F0\SQL
Server version 11.0.3128
Session Tracing ID
SPID 52
Name
The name of the connection.
Ready
Ln 7
Col 1
Ch 1
INS
99°F
Partly sunny
Search
5:52 PM
4/30/2023

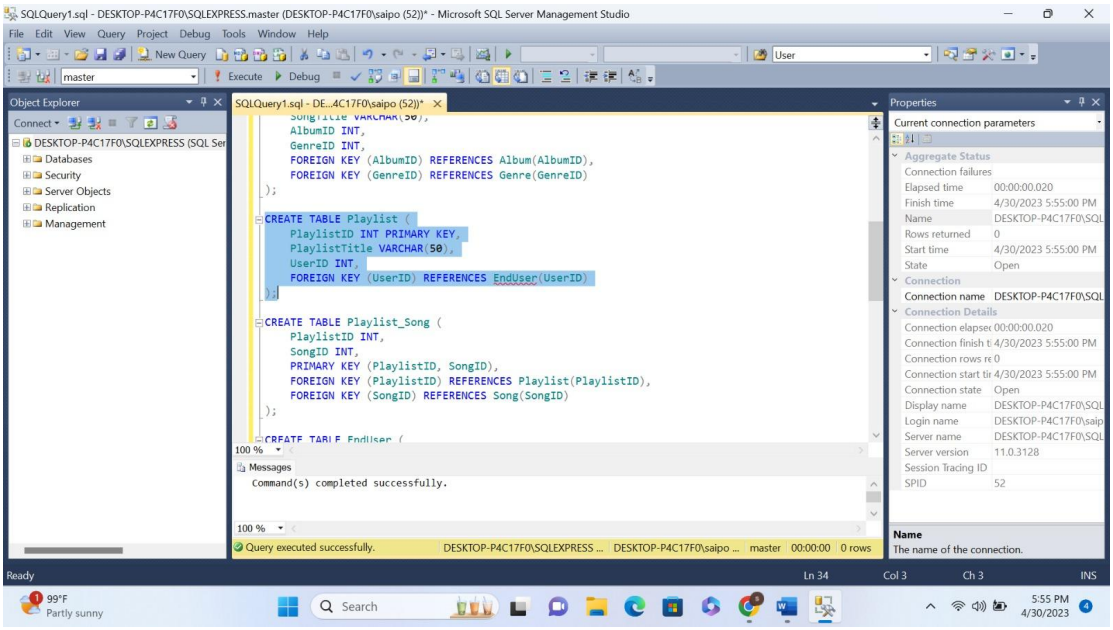

SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
CREATE TABLE Artist (
ArtistID INT PRIMARY KEY,
ArtistName VARCHAR(50),
ArtistBio TEXT
);
CREATE TABLE Genre (
GenreID INT PRIMARY KEY,
GenreName VARCHAR(20)
);
CREATE TABLE Album (
AlbumID INT PRIMARY KEY,
AlbumTitle VARCHAR(50),
ArtistID INT,
ReleaseYear INT,
FOREIGN KEY (ArtistID) REFERENCES Artist(ArtistID)
);
CREATE TABLE Song (
SongID INT PRIMARY KEY,
SongTitle VARCHAR(50),
Command(s) completed successfully.
Query executed successfully.

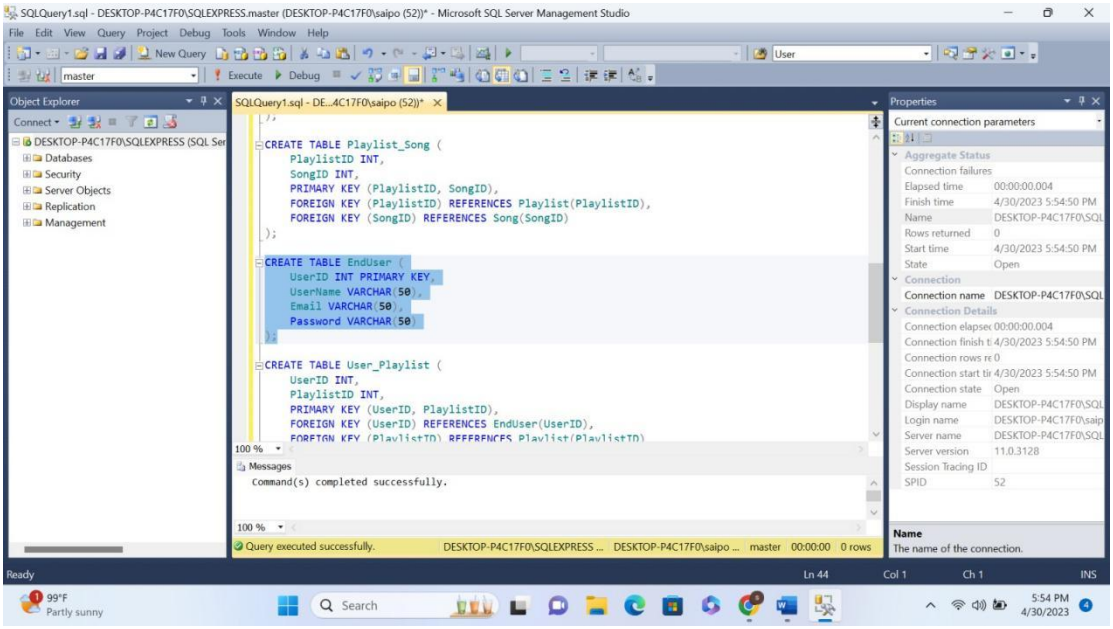

SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
SongID INT,
PRIMARY KEY (PlaylistID, SongID),
FOREIGN KEY (PlaylistID) REFERENCES Playlist(PlaylistID),
FOREIGN KEY (SongID) REFERENCES Song(SongID)
);
CREATE TABLE EndUser (
UserID INT PRIMARY KEY,
UserName VARCHAR(50),
Email VARCHAR(50),
Password VARCHAR(50)
);
CREATE TABLE User_Playlist (
UserID INT,
PlaylistID INT,
PRIMARY KEY (UserID, PlaylistID),
FOREIGN KEY (UserID) REFERENCES EndUser(UserID),
FOREIGN KEY (PlaylistID) REFERENCES Playlist(PlaylistID)
);
Command(s) completed successfully.
Query executed successfully.

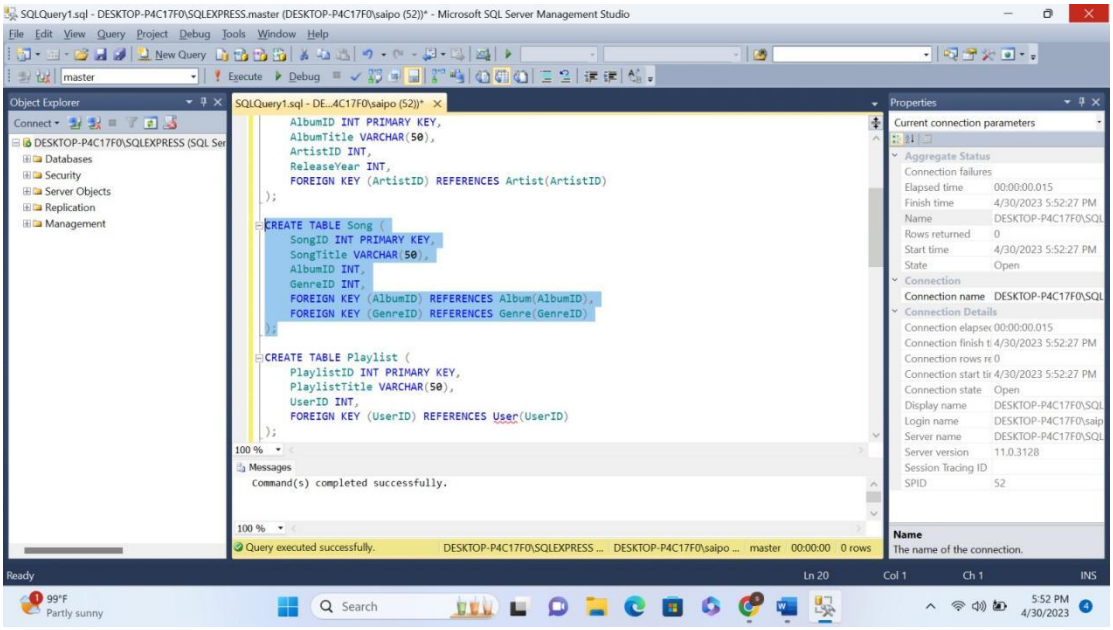

SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
CREATE TABLE Artist (
ArtistID INT PRIMARY KEY,
ArtistName VARCHAR(50),
ArtistBio TEXT
);
CREATE TABLE Genre (
GenreID INT PRIMARY KEY,
GenreName VARCHAR(20)
);
CREATE TABLE Album (
AlbumID INT PRIMARY KEY,
AlbumTitle VARCHAR(50),
ArtistID INT,
ReleaseYear INT,
FOREIGN KEY (ArtistID) REFERENCES Artist(ArtistID)
);
CREATE TABLE Song (
SongID INT PRIMARY KEY,
SongTitle VARCHAR(50),
Command(s) completed successfully.
Query executed successfully.


SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
AlbumID INT,
GenreID INT,
FOREIGN KEY (AlbumID) REFERENCES Album(AlbumID),
FOREIGN KEY (GenreID) REFERENCES Genre(GenreID)
);
CREATE TABLE Playlist (
PlaylistID INT PRIMARY KEY,
PlaylistTitle VARCHAR(50),
UserID INT,
FOREIGN KEY (UserID) REFERENCES EndUser(UserID)
);
CREATE TABLE Playlist_Song (
PlaylistID INT,
SongID INT,
PRIMARY KEY (PlaylistID, SongID),
FOREIGN KEY (PlaylistID) REFERENCES Playlist(PlaylistID),
FOREIGN KEY (SongID) REFERENCES Song(SongID)
);
Command(s) completed successfully.
Query executed successfully.


SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query
Execute Debug
master
User
Object Explorer
Connect
DESKTOP-P4C17F0\SQLEXPRESS (SQL Ser
Databases
Security
Server Objects
Replication
Management
SQLQuery1.sql - DE...4C17F0\saipo (52))*
AlbumID INT,
GenreID INT,
FOREIGN KEY (AlbumID) REFERENCES Album(AlbumID),
FOREIGN KEY (GenreID) REFERENCES Genre(GenreID)
);
CREATE TABLE Playlist (
PlaylistID INT PRIMARY KEY,
PlaylistTitle VARCHAR(50),
UserID INT,
FOREIGN KEY (UserID) REFERENCES EndUser(UserID)
);
CREATE TABLE Playlist_Song (
PlaylistID INT,
SongID INT,
PRIMARY KEY (PlaylistID, SongID),
FOREIGN KEY (PlaylistID) REFERENCES Playlist(PlaylistID),
FOREIGN KEY (SongID) REFERENCES Song(SongID)
);
100 %
Messages
Command(s) completed successfully.
Query executed successfully.
DESKTOP-P4C17F0\SQLEXPRESS ...
DESKTOP-P4C17F0\saipo ...
master
00:00:00
0 rows
Properties
Current connection parameters
Aggregate Status
Connection failures
Elapsed time 00:00:00.020
Finish time 4/30/2023 5:55:00 PM
Name DESKTOP-P4C17F0\SQL
Rows returned 0
Start time 4/30/2023 5:55:00 PM
State Open
Connection
Connection name DESKTOP-P4C17F0\SQL
Connection Details
Connection elapse 00:00:00.020
Connection finish t 4/30/2023 5:55:00 PM
Connection rows re 0
Connection start ti 4/30/2023 5:55:00 PM
Connection state Open
Display name DESKTOP-P4C17F0\SQL
Login name DESKTOP-P4C17F0\saip
Server name DESKTOP-P4C17F0\SQL
Server version 11.0.3128
Session Tracing ID
SPID 52
Name
The name of the connection.
Ready
Ln 34
Col 3
Ch 3
INS
99°F
Partly sunny
Search
5:55 PM
4/30/2023


SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query
Execute Debug
master
User
Object Explorer
Connect
DESKTOP-P4C17F0\SQLEXPRESS (SQL Ser
Databases
Security
Server Objects
Replication
Management
SQLQuery1.sql - DE...4C17F0\saipo (52))*
AlbumID INT,
GenreID INT,
FOREIGN KEY (AlbumID) REFERENCES Album(AlbumID),
FOREIGN KEY (GenreID) REFERENCES Genre(GenreID)
);
CREATE TABLE Playlist (
PlaylistID INT PRIMARY KEY,
PlaylistTitle VARCHAR(50),
UserID INT,
FOREIGN KEY (UserID) REFERENCES EndUser(UserID)
);
CREATE TABLE Playlist_Song (
PlaylistID INT,
SongID INT,
PRIMARY KEY (PlaylistID, SongID),
FOREIGN KEY (PlaylistID) REFERENCES Playlist(PlaylistID),
FOREIGN KEY (SongID) REFERENCES Song(SongID)
);
100 %
Messages
Command(s) completed successfully.
Query executed successfully.
DESKTOP-P4C17F0\SQLEXPRESS ...
DESKTOP-P4C17F0\saipo ...
master
00:00:00
0 rows
Properties
Current connection parameters
Aggregate Status
Connection failures
Elapsed time 00:00:00.020
Finish time 4/30/2023 5:55:00 PM
Name DESKTOP-P4C17F0\SQL
Rows returned 0
Start time 4/30/2023 5:55:00 PM
State Open
Connection
Connection name DESKTOP-P4C17F0\SQL
Connection Details
Connection elapse 00:00:00.020
Connection finish t 4/30/2023 5:55:00 PM
Connection rows re 0
Connection start ti 4/30/2023 5:55:00 PM
Connection state Open
Display name DESKTOP-P4C17F0\SQL
Login name DESKTOP-P4C17F0\saip
Server name DESKTOP-P4C17F0\SQL
Server version 11.0.3128
Session Tracing ID
SPID 52
Name
The name of the connection.
Ready
Ln 34
Col 3
Ch 3
INS
99°F
Partly sunny
Search
5:55 PM
4/30/2023


SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
CREATE TABLE Playlist_Song (
PlaylistID INT,
SongID INT,
PRIMARY KEY (PlaylistID, SongID),
FOREIGN KEY (PlaylistID) REFERENCES Playlist(PlaylistID),
FOREIGN KEY (SongID) REFERENCES Song(SongID)
);
CREATE TABLE EndUser (
UserID INT PRIMARY KEY,
UserName VARCHAR(50),
Email VARCHAR(50),
Password VARCHAR(50)
);
CREATE TABLE User_Playlist (
UserID INT,
PlaylistID INT,
PRIMARY KEY (UserID, PlaylistID),
FOREIGN KEY (UserID) REFERENCES EndUser(UserID),
Command(s) completed successfully.
Query executed successfully.
Ready


SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
AlbumID INT PRIMARY KEY,
AlbumTitle VARCHAR(50),
ArtistID INT,
ReleaseYear INT,
FOREIGN KEY (ArtistID) REFERENCES Artist(ArtistID)
);
CREATE TABLE Song (
SongID INT PRIMARY KEY,
SongTitle VARCHAR(50),
AlbumID INT,
GenreID INT,
FOREIGN KEY (AlbumID) REFERENCES Album(AlbumID),
FOREIGN KEY (GenreID) REFERENCES Genre(GenreID)
);
CREATE TABLE Playlist (
PlaylistID INT PRIMARY KEY,
PlaylistTitle VARCHAR(50),
UserID INT,
FOREIGN KEY (UserID) REFERENCES User(UserID)
);
Command(s) completed successfully.
Query executed successfully.
Ready

INSERT INTO Artist (ArtistID, ArtistName, ArtistBio) VALUES

(1, 'The Beatles', 'The Beatles were an English rock band formed in Liverpool in 1960.'),

(2, 'Bob Dylan', 'Bob Dylan is an American singer-songwriter, author, and visual artist.')

(3, 'Michael Jackson', 'Michael Jackson was an American singer, songwriter, and dancer.'),

(4, 'Prince', 'Prince was an American singer-songwriter, musician, record producer, and filmmaker.'),

(5, 'Beyoncé', 'Beyoncé is an American singer, songwriter, actress, and record producer.'),

(6, 'Jay-Z', 'Jay-Z is an American rapper, songwriter, record executive, and businessman.'),

(7, 'Taylor Swift', 'Taylor Swift is an American singer-songwriter.'),

(8, 'Adele', 'Adele is an English singer-songwriter.'),

(9, 'Ed Sheeran', 'Ed Sheeran is an English singer-songwriter and musician.'),

(10, 'Lady Gaga', 'Lady Gaga is an American singer, songwriter, and actress.'),

(11, 'Rihanna', 'Rihanna is a Barbadian singer, actress, and businesswoman.'),

(12, 'Justin Bieber', 'Justin Bieber is a Canadian singer, songwriter, and actor.'),

(13, 'Drake', 'Drake is a Canadian rapper, singer, songwriter, and actor.'),

(14, 'Kanye West', 'Kanye West is an American rapper, singer, songwriter, and record producer.'),

(15, 'Eminem', 'Eminem is an American rapper, songwriter, and record producer.'),

(16, 'Lana Del Rey', 'Lana Del Rey is an American singer-songwriter.'),

(17, 'Billie Eilish', 'Billie Eilish is an American singer-songwriter.'),

(18, 'Harry Styles', 'Harry Styles is an English singer, songwriter, and actor.'),

(19, 'Dua Lipa', 'Dua Lipa is an English singer, songwriter, and model.'),

(20, 'Bruno Mars', 'Bruno Mars is an American singer, songwriter, and record producer.');

```
INSERT INTO Genre (GenreID, GenreName) VALUES

(1, 'Rock'),

(2, 'Pop'),

(3, 'Hip-Hop'),

(4, 'Country'),

(5, 'Jazz'),

(6, 'Classical'),

(7, 'Electronic'),

(8, 'Reggae'),

(9, 'R&B');

INSERT INTO Album (AlbumID, AlbumTitle, ArtistID, ReleaseYear) VALUES

(1, 'Abbey Road', 1, 1969),

(2, 'The Game', 2, 1980),

(3, 'Thriller', 3, 1982),

(4, 'Lemonade', 4, 2016),
```

(5, '÷', 5, 2017),

(6, '21', 6, 2011),

(7, 'Unorthodox Jukebox', 7, 2012),

(8, 'FutureSex/LoveSounds', 8, 2006),

(9, 'The Fame', 9, 2008),

(10, 'Teenage Dream', 10, 2010);

INSERT INTO Song (SongID, SongTitle, AlbumID, GenreID) VALUES

(1, 'Come Together', 1, 1),

(2, 'Bohemian Rhapsody', 2, 1),

(3, 'Thriller', 3, 3),

(4, 'Formation', 4, 2),

(5, 'Shape of You', 5, 2),

(6, 'Rolling in the Deep', 6, 2),

(7, 'Locked Out of Heaven', 7, 2),

(8, 'SexyBack', 8, 3),

```
(9, 'Poker Face', 9, 2),

(10, 'Firework', 10, 2);

INSERT INTO Playlist (PlaylistID, PlaylistTitle, UserID) VALUES

(1, 'Top Hits', 1),

(2, 'Classic Rock', 2),

(3, 'Pop Mix', 3),

(4, '80s Jams', 4),

(5, 'Hip-Hop Hits', 5);

INSERT INTO EndUser (UserID, UserName, Email, Password)

VALUES

   (1, 'John Doe', 'johndoe@example.com', 'password1'),

   (2, 'Jane Smith', 'janesmith@example.com', 'password2'),

   (3, 'Robert Johnson', 'robertjohnson@example.com', 'password3'),

   (4, 'Sarah Lee', 'sarahlee@example.com', 'password4'),

   (5, 'David Kim', 'davidkim@example.com', 'password5'),
```

(6, 'Emily Brown', 'emilybrown@example.com', 'password6'),

(7, 'Michael Wang', 'michaelwang@example.com', 'password7'),

(8, 'Jessica Garcia', 'jessicagarcia@example.com', 'password8'),

(9, 'Daniel Nguyen', 'danielnguyen@example.com', 'password9'),

(10, 'Megan Taylor', 'megantaylor@example.com', 'password10'),

(11, 'Christopher Jones', 'christopherjones@example.com', 'password11'),

(12, 'Elizabeth Kim', 'elizabethkim@example.com', 'password12'),

(13, 'Jonathan Lee', 'jonathanlee@example.com', 'password13'),

(14, 'Olivia Johnson', 'oliviajohnson@example.com', 'password14'),

(15, 'William Chen', 'williamchen@example.com', 'password15'),

(16, 'Avery Wilson', 'averywilson@example.com', 'password16'),

(17, 'Ethan Davis', 'ethandavis@example.com', 'password17'),

(18, 'Natalie Lopez', 'natalielopez@example.com', 'password18'),

(19, 'Ryan Hernandez', 'ryanhernandez@example.com', 'password19'),

(20, 'Sophia Martinez', 'sophiamartinez@example.com', 'password20');

```
INSERT INTO Playlist_Song (SongID, PlaylistID)

VALUES

    (1, 1),

    (2, 1),

    (3, 1),

    (4, 1),

    (5, 1),

    (6, 2),

    (7, 2),

    (8, 2),

    (9, 3),

    (10, 3),

    (1, 4),

    (2, 4),

    (3, 4),
```

(4, 5),

(5, 5),

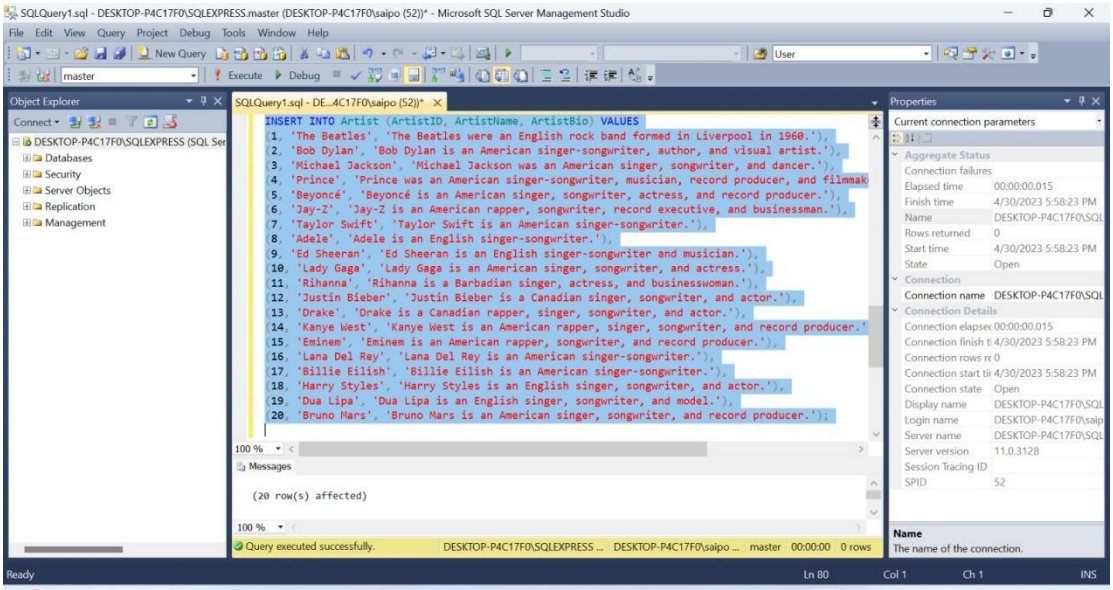
(6, 5),

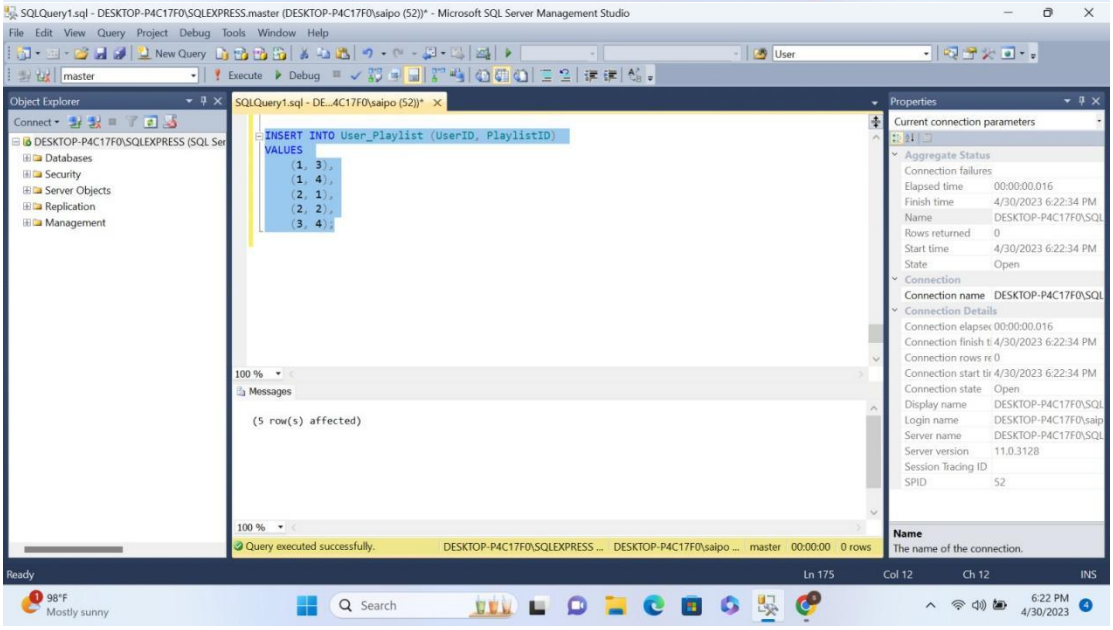
(7, 5),

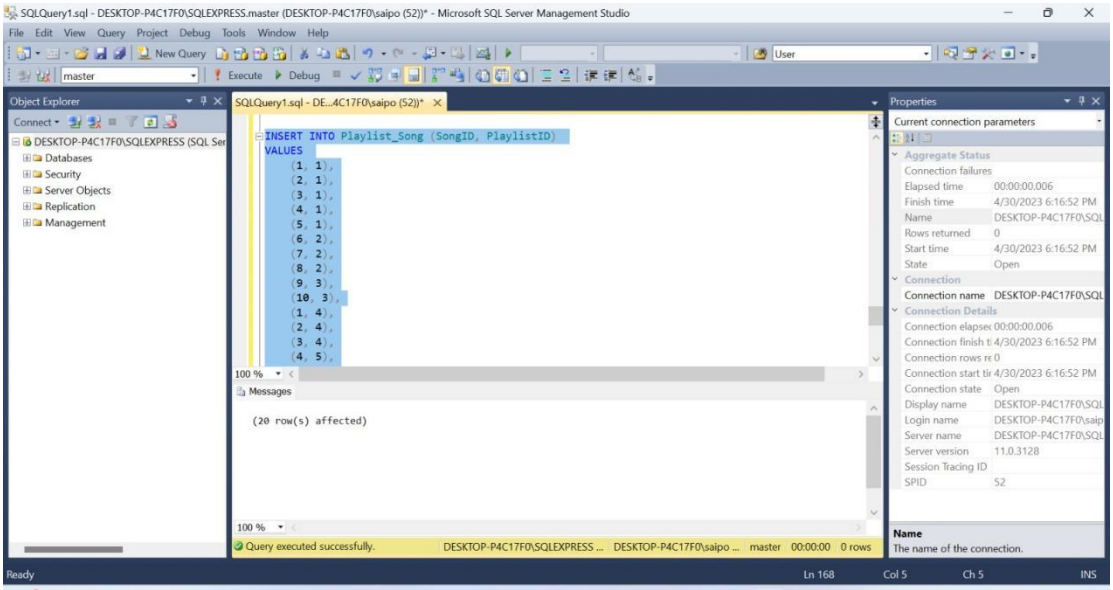
(8, 5),

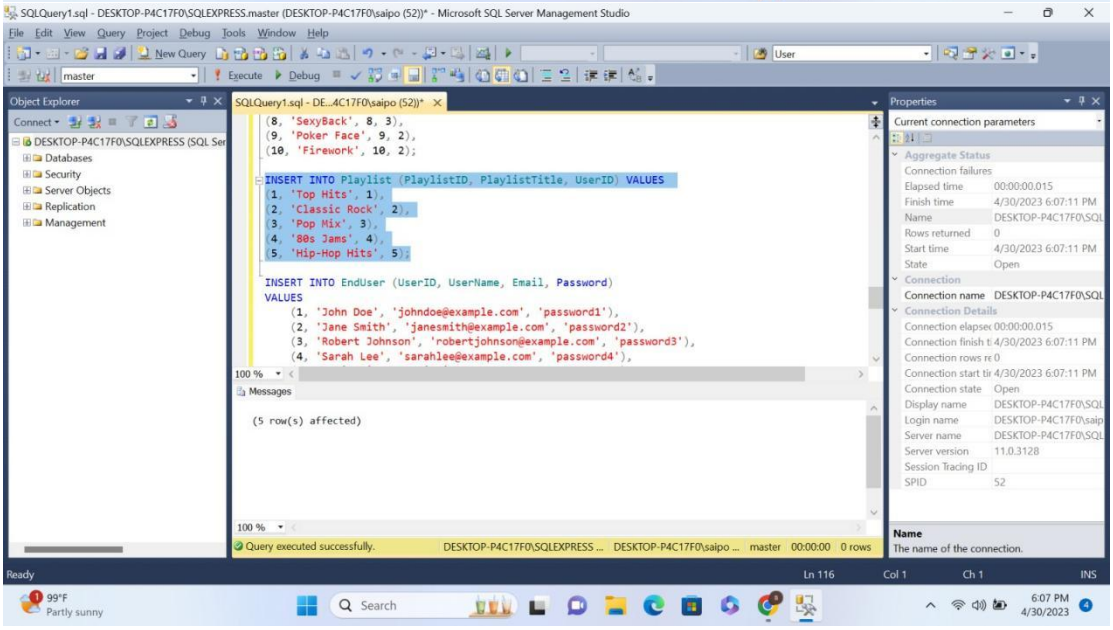
(9, 5),

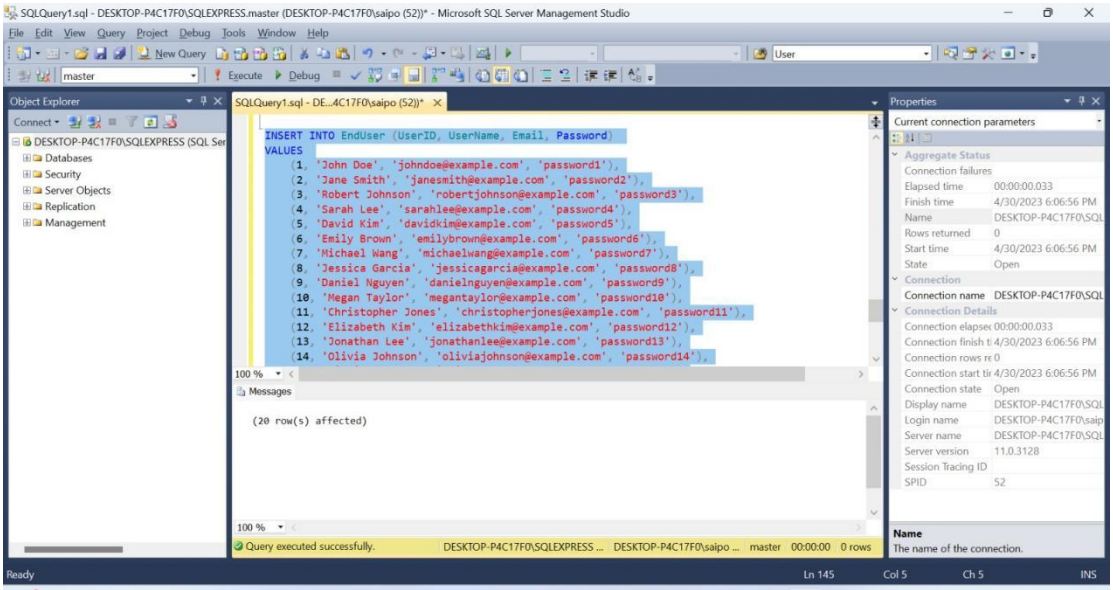
(10, 5);Job Flow

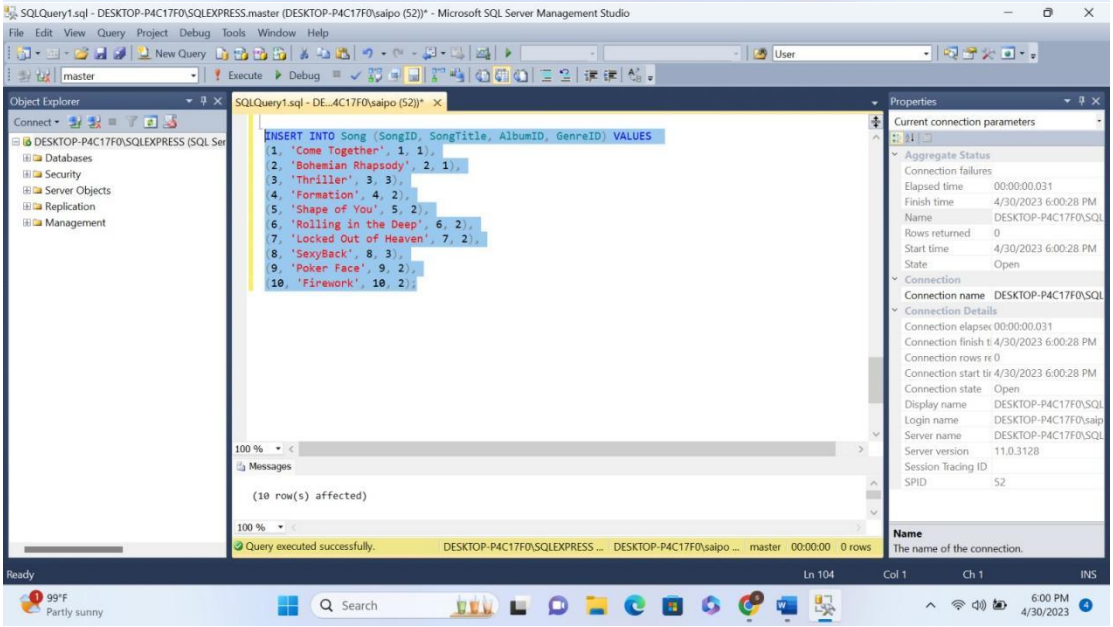

SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
INSERT INTO Artist (ArtistID, ArtistName, ArtistBio) VALUES
(1, 'The Beatles', 'The Beatles were an English rock band formed in Liverpool in 1960.'),
(2, 'Bob Dylan', 'Bob Dylan is an American singer-songwriter, author, and visual artist.'),
(3, 'Michael Jackson', 'Michael Jackson was an American singer, songwriter, and dancer.'),
(4, 'Prince', 'Prince was an American singer-songwriter, musician, record producer, and filmmak
(5, 'Beyoncé', 'Beyoncé is an American singer, songwriter, actress, and record producer.'),
(6, 'Jay-Z', 'Jay-Z is an American rapper, songwriter, record executive, and businessman.'),
(7, 'Taylor Swift', 'Taylor Swift is an American singer-songwriter.'),
(8, 'Adele', 'Adele is an English singer-songwriter.'),
(9, 'Ed Sheeran', 'Ed Sheeran is an English singer-songwriter and musician.'),
(10, 'Lady Gaga', 'Lady Gaga is an American singer, songwriter, and actress.'),
(11, 'Rihanna', 'Rihanna is a Barbadian singer, actress, and businesswoman.'),
(12, 'Justin Bieber', 'Justin Bieber is a Canadian singer, songwriter, and actor.'),
(13, 'Drake', 'Drake is a Canadian rapper, singer, songwriter, and actor.'),
(14, 'Kanye West', 'Kanye West is an American rapper, singer, songwriter, and record producer.'
(15, 'Eminem', 'Eminem is an American rapper, songwriter, and record producer.'),
(16, 'Lana Del Rey', 'Lana Del Rey is an American singer-songwriter.'),
(17, 'Billie Eilish', 'Billie Eilish is an American singer-songwriter.'),
(18, 'Harry Styles', 'Harry Styles is an English singer, songwriter, and actor.'),
(19, 'Dua Lipa', 'Dua Lipa is an English singer, songwriter, and model.'),
(20, 'Bruno Mars', 'Bruno Mars is an American singer, songwriter, and record producer.');
(20 row(s) affected)
Query executed successfully.
Ready

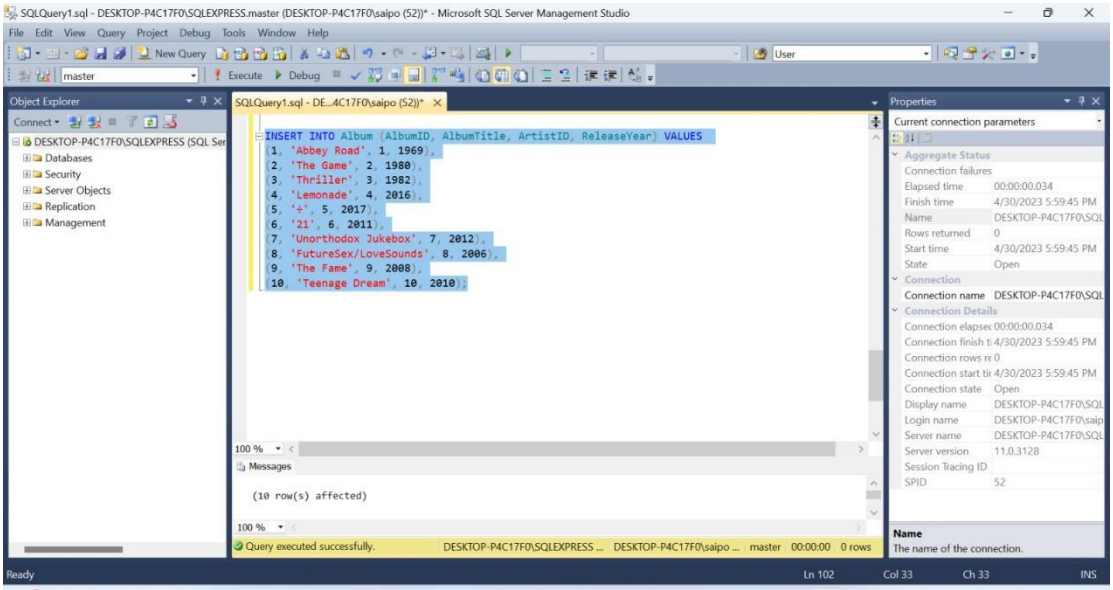

SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
INSERT INTO User_Playlist (UserID, PlaylistID)
VALUES
(1, 3),
(1, 4),
(2, 1),
(2, 2),
(3, 4);
(5 row(s) affected)
Query executed successfully.
Ready

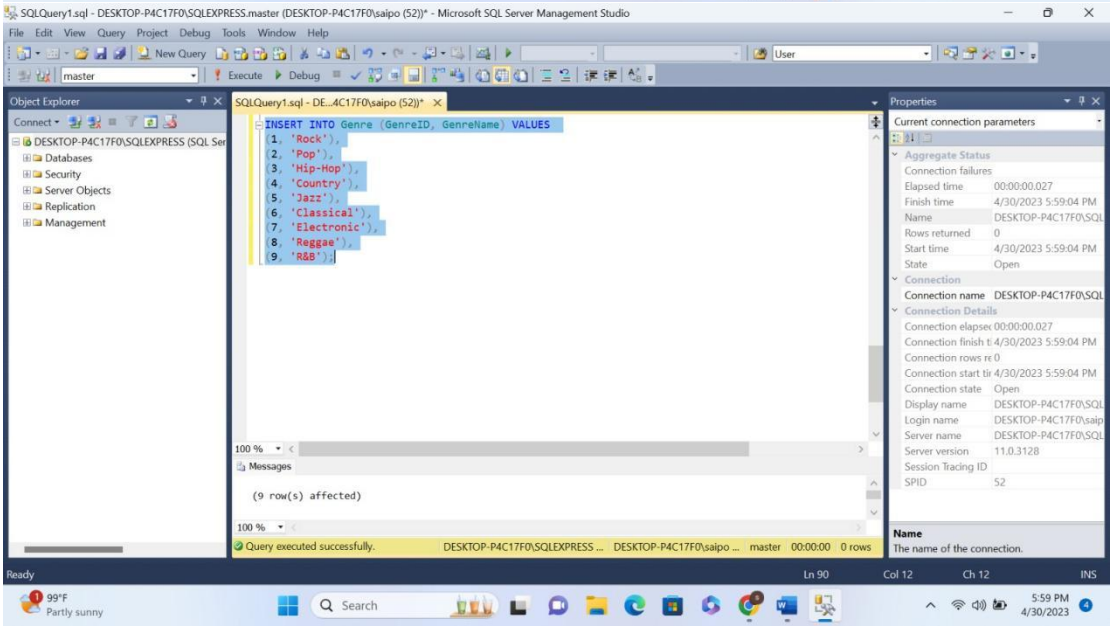

SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
INSERT INTO Playlist_Song (SongID, PlaylistID)
VALUES
(1, 1),
(2, 1),
(3, 1),
(4, 1),
(5, 1),
(6, 2),
(7, 2),
(8, 2),
(9, 3),
(10, 3),
(1, 4),
(2, 4),
(3, 4),
(4, 5),
(20 row(s) affected)
Query executed successfully.

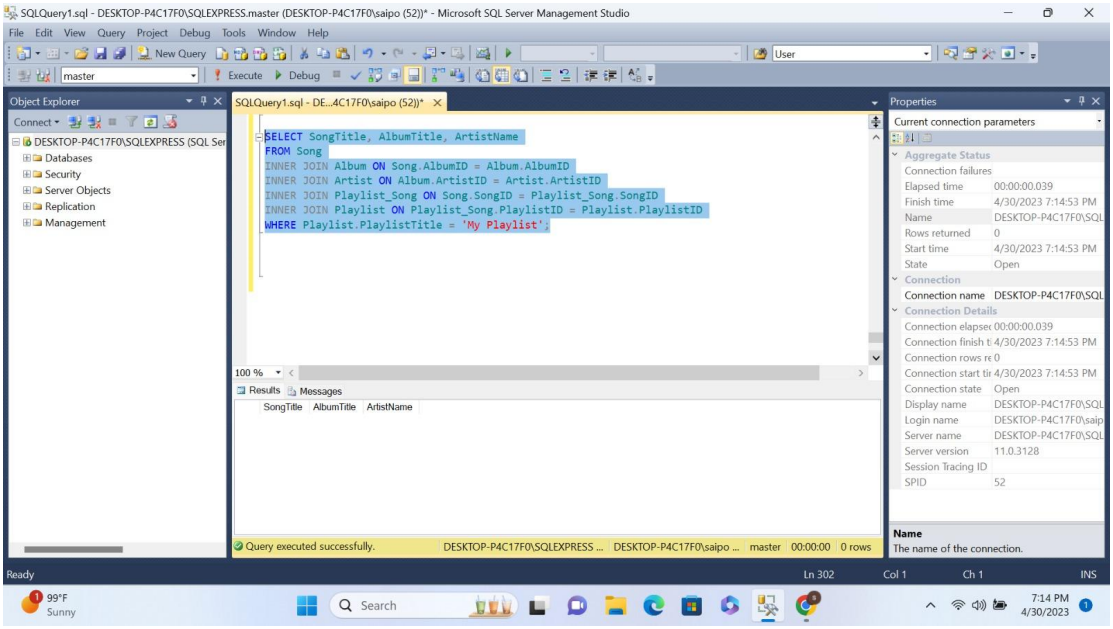

SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
(8, 'SexyBack', 8, 3),
(9, 'Poker Face', 9, 2),
(10, 'Firework', 10, 2);
INSERT INTO Playlist (PlaylistID, PlaylistTitle, UserID) VALUES
(1, 'Top Hits', 1),
(2, 'Classic Rock', 2),
(3, 'Pop Mix', 3),
(4, '80s Jams', 4),
(5, 'Hip-Hop Hits', 5);
INSERT INTO EndUser (UserID, UserName, Email, Password)
VALUES
(1, 'John Doe', 'johndoe@example.com', 'password1'),
(2, 'Jane Smith', 'janesmith@example.com', 'password2'),
(3, 'Robert Johnson', 'robertjohnson@example.com', 'password3'),
(4, 'Sarah Lee', 'sarahlee@example.com', 'password4'),
(5 row(s) affected)
Query executed successfully.

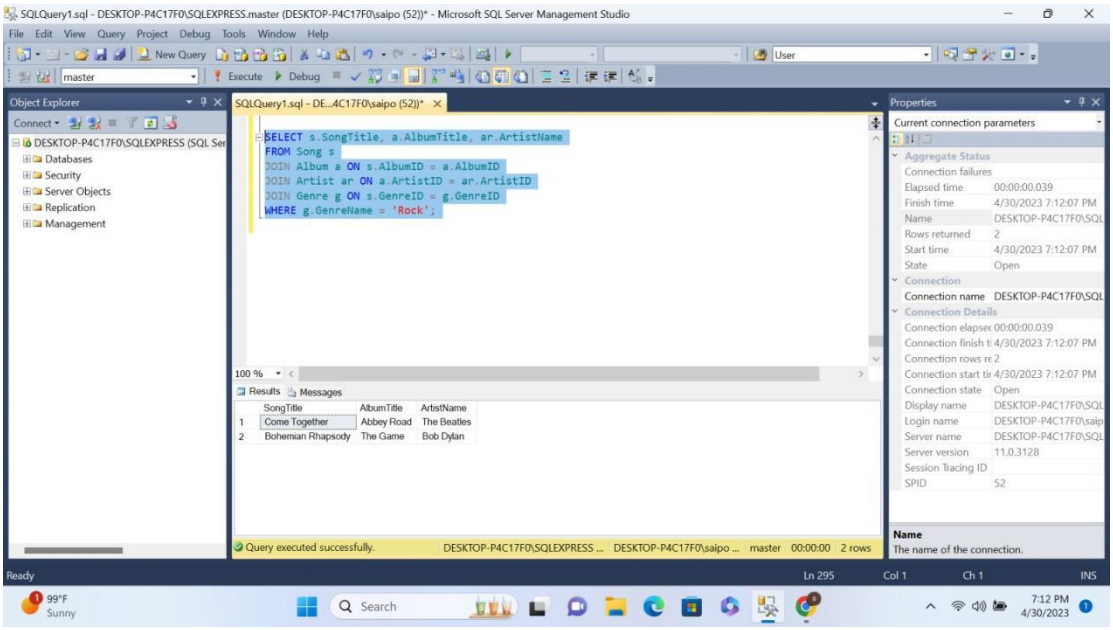

SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
INSERT INTO EndUser (UserID, UserName, Email, Password)
VALUES
(1, 'John Doe', 'johndoe@example.com', 'password1'),
(2, 'Jane Smith', 'janesmith@example.com', 'password2'),
(3, 'Robert Johnson', 'robertjohnson@example.com', 'password3'),
(4, 'Sarah Lee', 'sarahlee@example.com', 'password4'),
(5, 'David Kim', 'davidkim@example.com', 'password5'),
(6, 'Emily Brown', 'emilybrown@example.com', 'password6'),
(7, 'Michael Wang', 'michaelwang@example.com', 'password7'),
(8, 'Jessica Garcia', 'jessicagarcia@example.com', 'password8'),
(9, 'Daniel Nguyen', 'danielnguyen@example.com', 'password9'),
(10, 'Megan Taylor', 'megantaylor@example.com', 'password10'),
(11, 'Christopher Jones', 'christopherjones@example.com', 'password11'),
(12, 'Elizabeth Kim', 'elizabethkim@example.com', 'password12'),
(13, 'Jonathan Lee', 'jonathanlee@example.com', 'password13'),
(14, 'Olivia Johnson', 'oliviajohnson@example.com', 'password14'),
(20 row(s) affected)
Query executed successfully.
Ready

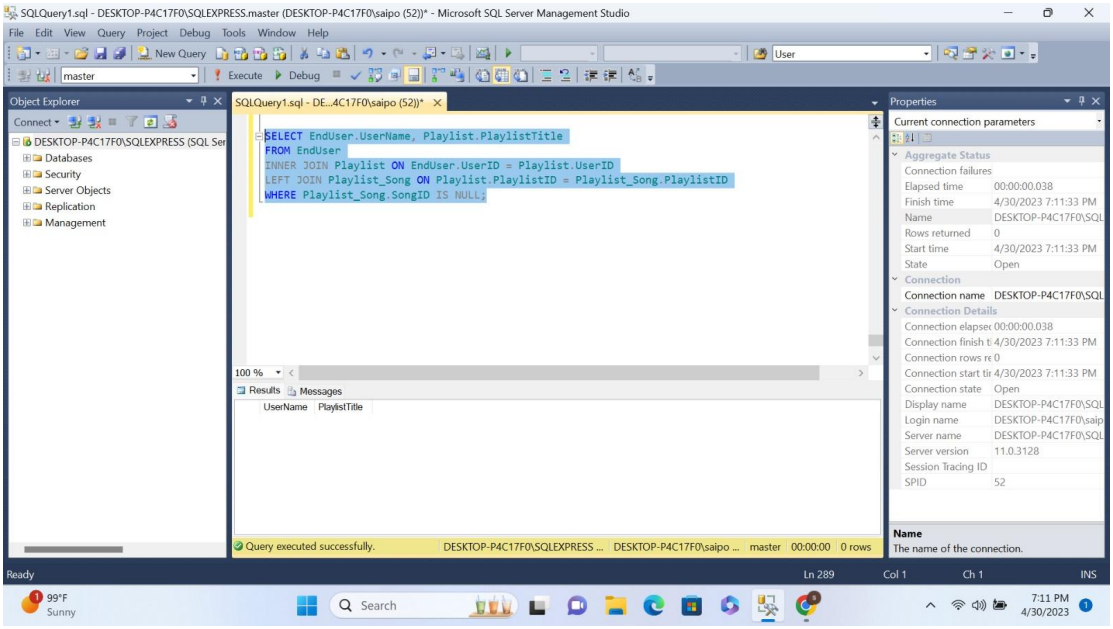

SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
INSERT INTO Song (SongID, SongTitle, AlbumID, GenreID) VALUES
(1, 'Come Together', 1, 1),
(2, 'Bohemian Rhapsody', 2, 1),
(3, 'Thriller', 3, 3),
(4, 'Formation', 4, 2),
(5, 'Shape of You', 5, 2),
(6, 'Rolling in the Deep', 6, 2),
(7, 'Locked Out of Heaven', 7, 2),
(8, 'SexyBack', 8, 3),
(9, 'Poker Face', 9, 2),
(10, 'Firework', 10, 2);
(10 row(s) affected)
Query executed successfully.
Ready

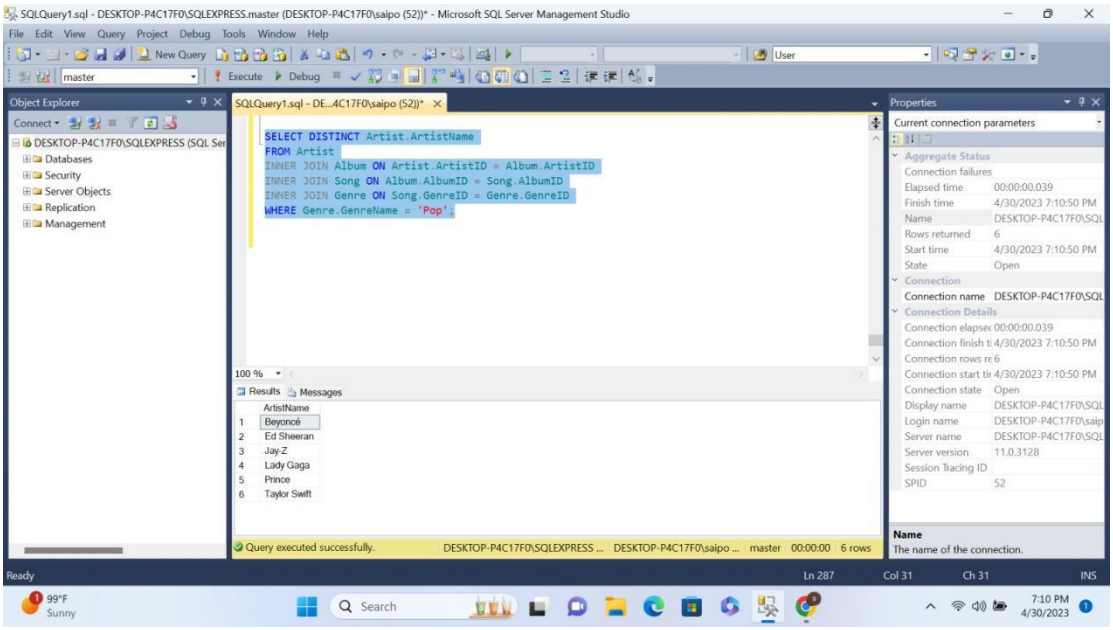

SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
INSERT INTO Album (AlbumID, AlbumTitle, ArtistID, ReleaseYear) VALUES
(1, 'Abbey Road', 1, 1969),
(2, 'The Game', 2, 1980),
(3, 'Thriller', 3, 1982),
(4, 'Lemonade', 4, 2016),
(5, '+', 5, 2017),
(6, '21', 6, 2011),
(7, 'Unorthodox Jukebox', 7, 2012),
(8, 'FutureSex/LoveSounds', 8, 2006),
(9, 'The Fame', 9, 2008),
(10, 'Teenage Dream', 10, 2010);
(10 row(s) affected)
Query executed successfully.

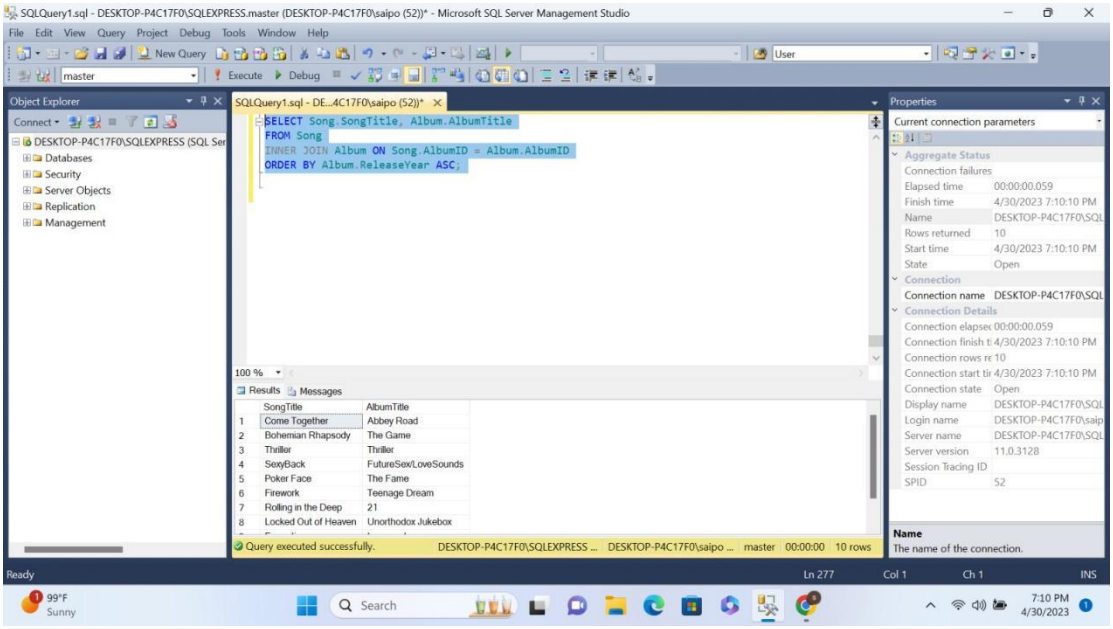

SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
INSERT INTO Genre (GenreID, GenreName) VALUES
(1, 'Rock'),
(2, 'Pop'),
(3, 'Hip-Hop'),
(4, 'Country'),
(5, 'Jazz'),
(6, 'Classical'),
(7, 'Electronic'),
(8, 'Reggae'),
(9, 'R&B');
(9 row(s) affected)
Query executed successfully.


SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
SELECT SongTitle, AlbumTitle, ArtistName
FROM Song
INNER JOIN Album ON Song.AlbumID = Album.AlbumID
INNER JOIN Artist ON Album.ArtistID = Artist.ArtistID
INNER JOIN Playlist_Song ON Song.SongID = Playlist_Song.SongID
INNER JOIN Playlist ON Playlist_Song.PlaylistID = Playlist.PlaylistID
WHERE Playlist.PlaylistTitle = 'My Playlist';
Results
Messages
SongTitle
AlbumTitle
ArtistName
Query executed successfully.
Ready


SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
SELECT s.SongTitle, a.AlbumTitle, ar.ArtistName
FROM Song s
JOIN Album a ON s.AlbumID = a.AlbumID
JOIN Artist ar ON a.ArtistID = ar.ArtistID
JOIN Genre g ON s.GenreID = g.GenreID
WHERE g.GenreName = 'Rock';
Results
Messages
SongTitle
AlbumTitle
ArtistName
1
Come Together
Abbey Road
The Beatles
2
Bohemian Rhapsody
The Game
Bob Dylan
Query executed successfully.
Ready


SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
SELECT EndUser.UserName, Playlist.PlaylistTitle
FROM EndUser
INNER JOIN Playlist ON EndUser.UserID = Playlist.UserID
LEFT JOIN Playlist_Song ON Playlist.PlaylistID = Playlist_Song.PlaylistID
WHERE Playlist_Song.SongID IS NULL;
Results
Messages
UserName
PlaylistTitle
Query executed successfully.


SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
SELECT DISTINCT Artist.ArtistName
FROM Artist
INNER JOIN Album ON Artist.ArtistID = Album.ArtistID
INNER JOIN Song ON Album.AlbumID = Song.AlbumID
INNER JOIN Genre ON Song.GenreID = Genre.GenreID
WHERE Genre.GenreName = 'Pop';
Results
Messages
ArtistName
1 Beyoncé
2 Ed Sheeran
3 Jay-Z
4 Lady Gaga
5 Prince
6 Taylor Swift
Query executed successfully.


SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query
Execute Debug
master
User
Object Explorer
Connect
DESKTOP-P4C17F0\SQLEXPRESS (SQL Ser
Databases
Security
Server Objects
Replication
Management
SQLQuery1.sql - DE...4C17F0\saipo (52))*
SELECT Song.SongTitle, Album.AlbumTitle
FROM Song
INNER JOIN Album ON Song.AlbumID = Album.AlbumID
ORDER BY Album.ReleaseYear ASC;
100 %
Results
Messages
SongTitle AlbumTitle
1 Come Together Abbey Road
2 Bohemian Rhapsody The Game
3 Thriller Thriller
4 SexyBack FutureSex/LoveSounds
5 Poker Face The Fame
6 Firework Teenage Dream
7 Rolling in the Deep 21
8 Locked Out of Heaven Unorthodox Jukebox
Query executed successfully.
DESKTOP-P4C17F0\SQLEXPRESS ...
DESKTOP-P4C17F0\saipo ...
master
00:00:00
10 rows
Properties
Current connection parameters
Aggregate Status
Connection failures
Elapsed time 00:00:00.059
Finish time 4/30/2023 7:10:10 PM
Name DESKTOP-P4C17F0\SQL
Rows returned 10
Start time 4/30/2023 7:10:10 PM
State Open
Connection
Connection name DESKTOP-P4C17F0\SQL
Connection Details
Connection elapse 00:00:00.059
Connection finish t 4/30/2023 7:10:10 PM
Connection rows re 10
Connection start ti 4/30/2023 7:10:10 PM
Connection state Open
Display name DESKTOP-P4C17F0\SQL
Login name DESKTOP-P4C17F0\saip
Server name DESKTOP-P4C17F0\SQL
Server version 11.0.3128
Session Tracing ID
SPID 52
Name
The name of the connection.
Ready
Ln 277
Col 1
Ch 1
INS
99°F
Sunny
Search
7:10 PM
4/30/2023

Stored Procedure

This procedure takes in an Artist's name as an input and gives his most famous song.

```
CREATE PROCEDURE GetTopSongsByGenre

 @GenreID INT

AS

BEGIN

SELECT TOP 10 s.SongID, s.SongTitle,

COUNT(*) AS Popularity

   FROM Playlist_Song ps

 INNER JOIN Song s ON p

s.SongID = s.SongID

WHERE s.GenreID = @GenreID

 GROUP BY s.SongID, s.SongTitle

ORDER BY Popularity DESC

END
```

```
CREATE PROCEDURE sp_SearchSongs

  @SearchString VARCHAR(50)

AS

BEGIN

 SELECT DISTINCT

s.SongID, s.SongTitle, a.AlbumTitle, ar.ArtistName, g.GenreName

  FROM Song s

 JOIN Album a ON s.AlbumID = a.AlbumID

 JOIN Artist ar ON a.ArtistID = ar.ArtistID

JOIN Genre g ON s.GenreID = g.GenreID

WHERE s.SongTitle LIKE '%' +

 @SearchString + '%'

  OR a.AlbumTitle LIKE '%' +

 @SearchString + '%'

 OR ar.ArtistName LIKE '%' +
```

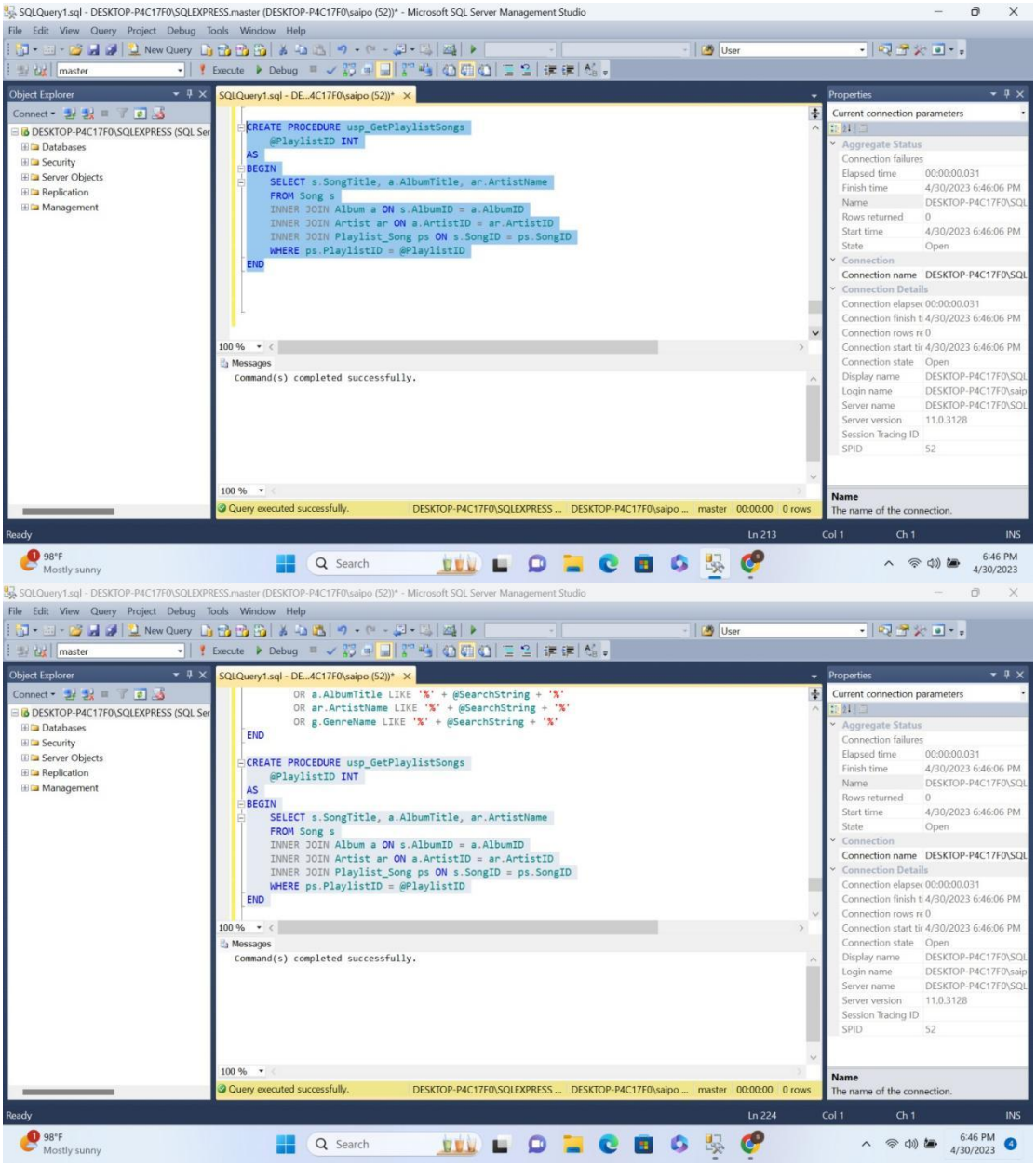
```
@SearchString + '%'

  OR g.GenreName LIKE '%' +

 @SearchString + '%'

END

CREATE PROCEDURE usp_GetPlaylistSongs

  @PlaylistID INT

AS

BEGIN

   SELECT s.SongTitle, a.AlbumTitle,

ar.ArtistName

 FROM Song s

   INNER JOIN Album a ON s.AlbumID =

a.AlbumID

  INNER JOIN Artist ar ON a.ArtistID =

ar.ArtistID
```

```
    INNER JOIN Playlist_Song ps ON s.SongID =

    ps.SongID

WHERE ps.PlaylistID = @PlaylistID

END

CREATE TRIGGER tr_Playlist_DeleteSongs

ON Playlist

INSTEAD OF DELETE

AS

BEGIN

   DELETE FROM Playlist_Song WHERE PlaylistID IN (SELECT PlaylistID FROM
deleted);

 DELETE FROM Playlist WHERE PlaylistID IN (SELECT PlaylistID FROM
deleted);

END



CREATE TRIGGER
```

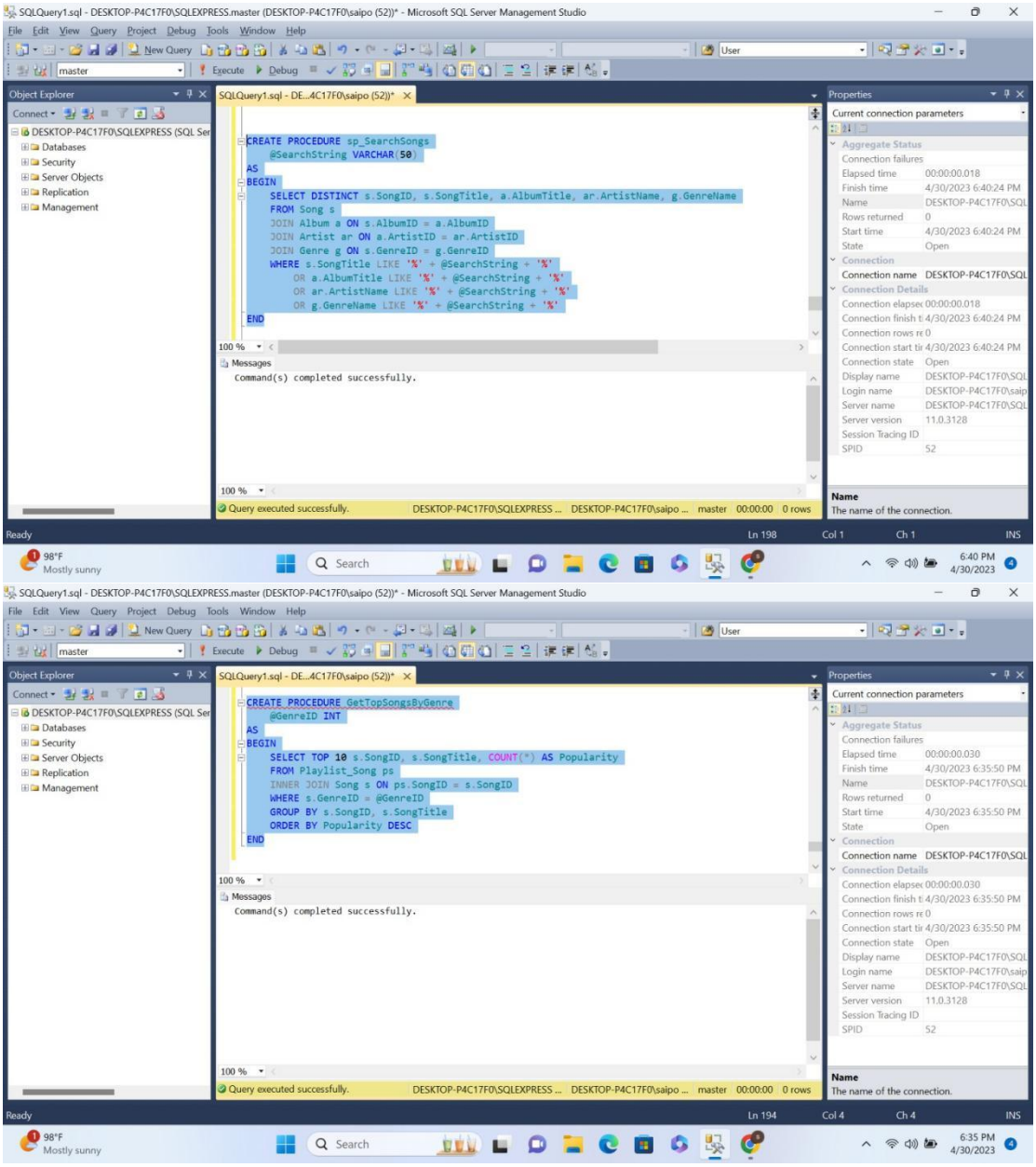
```
tr_update_playlist_song_count

ON Playlist_Song

AFTER INSERT, DELETE

AS

BEGIN

 DECLARE @PlaylistID INT

  SET @PlaylistID = (SELECT PlaylistID FROM inserted UNION SELECT
PlaylistID FROM deleted)

   UPDATE Playlist

 SET PlaylistTitle = (

  SELECT COUNT(*) FROM Playlist_Song WHERE PlaylistID = @PlaylistID

 )

 WHERE PlaylistID = @PlaylistID

END



CREATE TRIGGER tr_prevent_album_deletion
```

```
ON Album

FOR DELETE

AS

BEGIN

 IF EXISTS (SELECT * FROM Song WHERE AlbumID IN (SELECT AlbumID
FROM deleted))

  BEGIN

 RAISERROR('Cannot delete album because it has associated songs', 16, 1)

 ROLLBACK TRANSACTION

 END

END

CREATE TRIGGER tr_update_artist_bio

ON Album

AFTER INSERT

AS

BEGIN
```

```
DECLARE @ArtistID INT

SET @ArtistID = (SELECT ArtistID FROM inserted)

UPDATE Artist

SET ArtistBio = (

SELECT TOP 1 ArtistBio FROM Album WHERE ArtistID = @ArtistID ORDER BY ReleaseYear DESC  )

WHERE ArtistID = @ArtistID

END
```


SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query
Execute Debug
master
User
Object Explorer
Connect
DESKTOP-P4C17F0\SQLEXPRESS (SQL Ser
Databases
Security
Server Objects
Replication
Management
SQLQuery1.sql - DE...4C17F0\saipo (52))*
CREATE PROCEDURE usp_GetPlaylistSongs
@PlaylistID INT
AS
BEGIN
SELECT s.SongTitle, a.AlbumTitle, ar.ArtistName
FROM Song s
INNER JOIN Album a ON s.AlbumID = a.AlbumID
INNER JOIN Artist ar ON a.ArtistID = ar.ArtistID
INNER JOIN Playlist_Song ps ON s.SongID = ps.SongID
WHERE ps.PlaylistID = @PlaylistID
END
100 %
Messages
Command(s) completed successfully.
Query executed successfully.
DESKTOP-P4C17F0\SQLEXPRESS ...
DESKTOP-P4C17F0\saipo ...
master
00:00:00
0 rows
Properties
Current connection parameters
Aggregate Status
Connection failures
Elapsed time 00:00:00.031
Finish time 4/30/2023 6:46:06 PM
Name DESKTOP-P4C17F0\SQL
Rows returned 0
Start time 4/30/2023 6:46:06 PM
State Open
Connection
Connection name DESKTOP-P4C17F0\SQL
Connection Details
Connection elapsed 00:00:00.031
Connection finish ti 4/30/2023 6:46:06 PM
Connection rows re 0
Connection start tir 4/30/2023 6:46:06 PM
Connection state Open
Display name DESKTOP-P4C17F0\SQL
Login name DESKTOP-P4C17F0\saip
Server name DESKTOP-P4C17F0\SQL
Server version 11.0.3128
Session Tracing ID
SPID 52
Name
The name of the connection.
Ready
Ln 213
Col 1
Ch 1
INS
98°F
Mostly sunny
Search
6:46 PM
4/30/2023

SQLQuery1.sql - DESKTOP-P4C17F0\SQLEXPRESS.master (DESKTOP-P4C17F0\saipo (52))* - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query
Execute Debug
master
User
Object Explorer
Connect
DESKTOP-P4C17F0\SQLEXPRESS (SQL Ser
Databases
Security
Server Objects
Replication
Management
SQLQuery1.sql - DE...4C17F0\saipo (52))*
OR a.AlbumTitle LIKE '%' + @SearchString + '%'
OR ar.ArtistName LIKE '%' + @SearchString + '%'
OR g.GenreName LIKE '%' + @SearchString + '%'
END
CREATE PROCEDURE usp_GetPlaylistSongs
@PlaylistID INT
AS
BEGIN
SELECT s.SongTitle, a.AlbumTitle, ar.ArtistName
FROM Song s
INNER JOIN Album a ON s.AlbumID = a.AlbumID
INNER JOIN Artist ar ON a.ArtistID = ar.ArtistID
INNER JOIN Playlist_Song ps ON s.SongID = ps.SongID
WHERE ps.PlaylistID = @PlaylistID
END
100 %
Messages
Command(s) completed successfully.
Query executed successfully.
DESKTOP-P4C17F0\SQLEXPRESS ...
DESKTOP-P4C17F0\saipo ...
master
00:00:00
0 rows
Properties
Current connection parameters
Aggregate Status
Connection failures
Elapsed time 00:00:00.031
Finish time 4/30/2023 6:46:06 PM
Name DESKTOP-P4C17F0\SQL
Rows returned 0
Start time 4/30/2023 6:46:06 PM
State Open
Connection
Connection name DESKTOP-P4C17F0\SQL
Connection Details
Connection elapsed 00:00:00.031
Connection finish ti 4/30/2023 6:46:06 PM
Connection rows re 0
Connection start tir 4/30/2023 6:46:06 PM
Connection state Open
Display name DESKTOP-P4C17F0\SQL
Login name DESKTOP-P4C17F0\saip
Server name DESKTOP-P4C17F0\SQL
Server version 11.0.3128
Session Tracing ID
SPID 52
Name
The name of the connection.
Ready
Ln 224
Col 1
Ch 1
INS
98°F
Mostly sunny
Search
6:46 PM
4/30/2023



Trigger

This is an update trigger on the Album table whenever a new song is released

CREATE TRIGGER tr_Playlist_DeleteSongs

ON Playlist

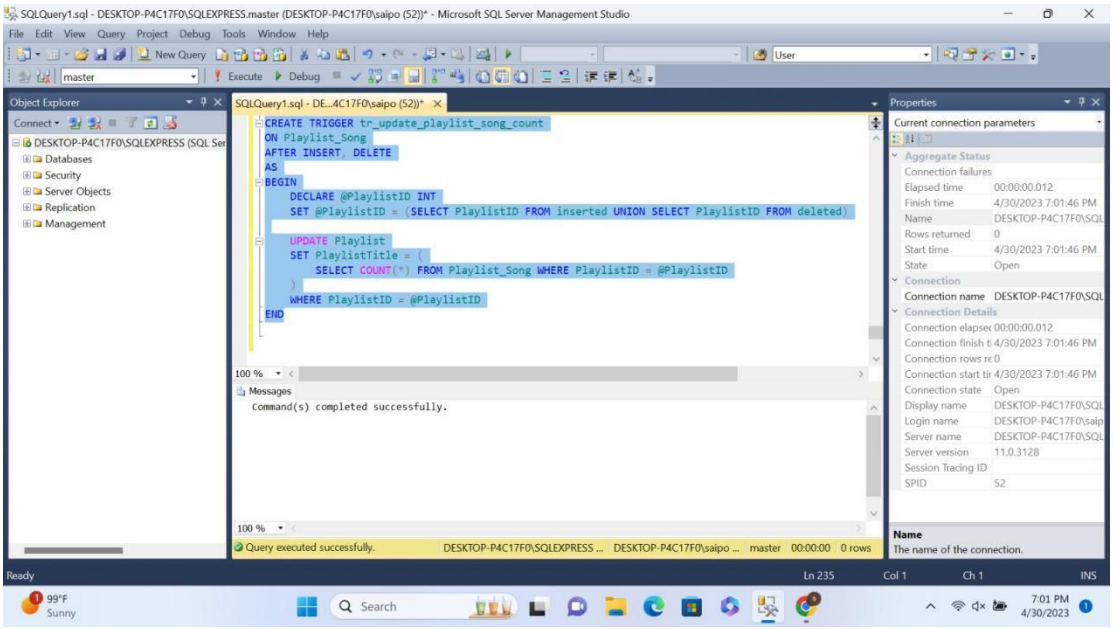
INSTEAD OF DELETE

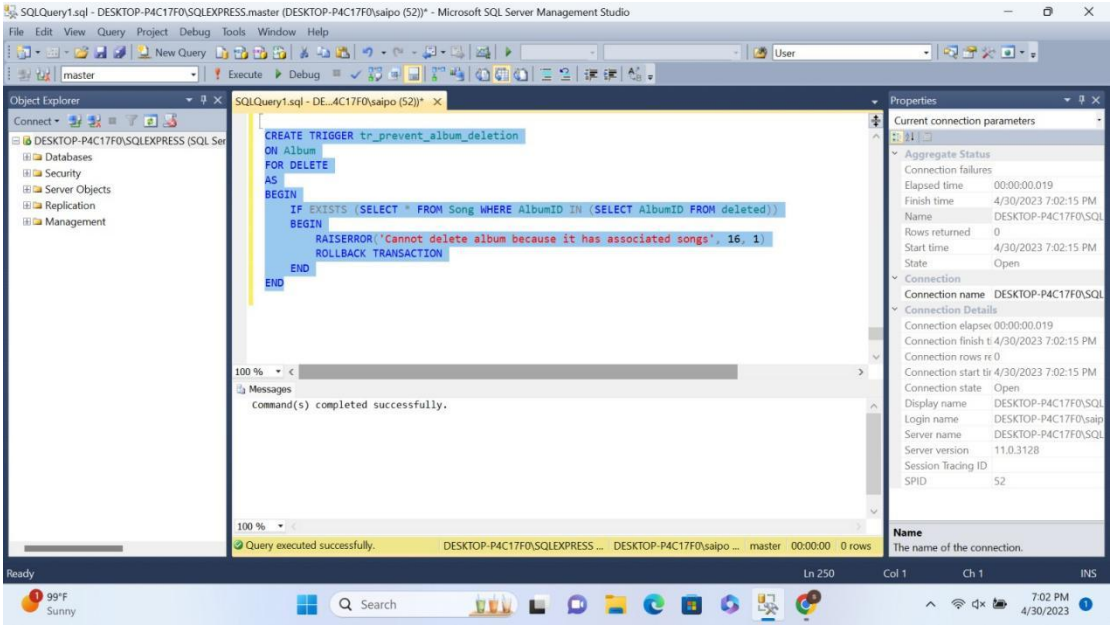
```
AS
BEGIN
   DELETE FROM Playlist_Song WHERE PlaylistID IN (SELECT PlaylistID FROM deleted);
   DELETE FROM Playlist WHERE PlaylistID IN (SELECT PlaylistID FROM deleted);
END

CREATE TRIGGER tr_update_playlist_song_count
ON Playlist_Song
AFTER INSERT, DELETE
AS
BEGIN
   DECLARE @PlaylistID INT
   SET @PlaylistID = (SELECT PlaylistID FROM inserted UNION SELECT PlaylistID FROM deleted)

   UPDATE Playlist
   SET PlaylistTitle = (
      SELECT COUNT(*) FROM Playlist_Song WHERE PlaylistID = @PlaylistID
   )
   WHERE PlaylistID = @PlaylistID
END

CREATE TRIGGER tr_prevent_album_deletion
```

```
ON Album
FOR DELETE
AS
BEGIN
   IF EXISTS (SELECT * FROM Song WHERE AlbumID IN (SELECT AlbumID
FROM deleted))
   BEGIN
      RAISERROR('Cannot delete album because it has associated songs', 16, 1)
      ROLLBACK TRANSACTION
   END
END

CREATE TRIGGER tr_update_artist_bio
ON Album
AFTER INSERT
AS
BEGIN
   DECLARE @ArtistID INT
   SET @ArtistID = (SELECT ArtistID FROM inserted)

   UPDATE Artist
   SET ArtistBio = (
      SELECT TOP 1 ArtistBio FROM Album WHERE ArtistID = @ArtistID
ORDER BY ReleaseYear DESC
   )
```

WHERE ArtistID = @ArtistID

END



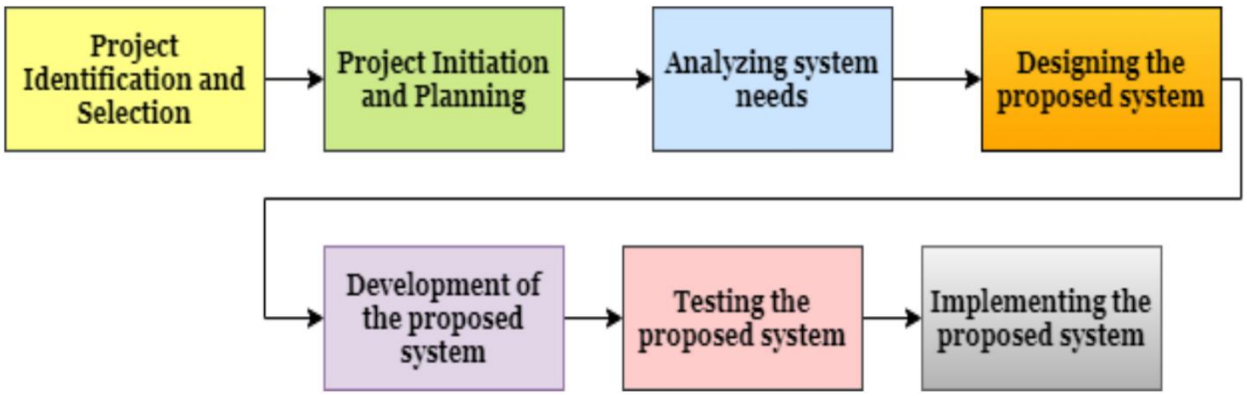


SQL QUERIES:

SELECT Song.SongTitle, Album.AlbumTitle

```
FROM Song
INNER JOIN Album ON Song.AlbumID = Album.AlbumID
ORDER BY Album.ReleaseYear ASC;

SELECT DISTINCT Artist.ArtistName
FROM Artist
INNER JOIN Album ON Artist.ArtistID = Album.ArtistID
INNER JOIN Song ON Album.AlbumID = Song.AlbumID
INNER JOIN Genre ON Song.GenreID = Genre.GenreID
WHERE Genre.GenreName = 'Pop';

SELECT EndUser.UserName, Playlist.PlaylistTitle
FROM EndUser
INNER JOIN Playlist ON EndUser.UserID = Playlist.UserID
LEFT JOIN Playlist_Song ON Playlist.PlaylistID = Playlist_Song.PlaylistID
WHERE Playlist_Song.SongID IS NULL;

SELECT s.SongTitle, a.AlbumTitle, ar.ArtistName
FROM Song s
JOIN Album a ON s.AlbumID = a.AlbumID
JOIN Artist ar ON a.ArtistID = ar.ArtistID
JOIN Genre g ON s.GenreID = g.GenreID
WHERE g.GenreName = 'Rock';

SELECT SongTitle, AlbumTitle, ArtistName
```

```
FROM Song
INNER JOIN Album ON Song.AlbumID = Album.AlbumID
INNER JOIN Artist ON Album.ArtistID = Artist.ArtistID
INNER JOIN Playlist_Song ON Song.SongID = Playlist_Song.SongID
INNER JOIN Playlist ON Playlist_Song.PlaylistID = Playlist.PlaylistID
WHERE Playlist.PlaylistTitle = 'My Playlist';
```



1) **Project identification and selection :**

The goal of this project is to develop a music library management system with an emphasis on managing tracks, albums, artists, users, and the ecosystem as a whole. On the Internet, you may find everything and everything about music. moving from neighborhood specialists to greater scale after starting with open permitted tracks.

2)Project initiation and planning :

We started by determining the project's scope and goal and collecting user requirements for the system. The results of this strategy are as follows: variables to

take into account include scope and constraints, objectives, costs and advantages, suggested system features, and user interface choice.

3)**Analyzing system needs :**

Before drawing a data flow diagram for the existing framework, we looked into it and found problems. For the suggested data set mapping, we also created an information stream graph (DFD) and an item connection chart.

4) Designing the proposed system :

After the research phase, we created the DFD and user interface, a data dictionary, and a relational database model from the ORM diagram.

5) Developing of proposed system :

At this step, we will convert the suggested system's architecture into software. Computer programming will be needed to handle MySQL management and translate design requirements into software.

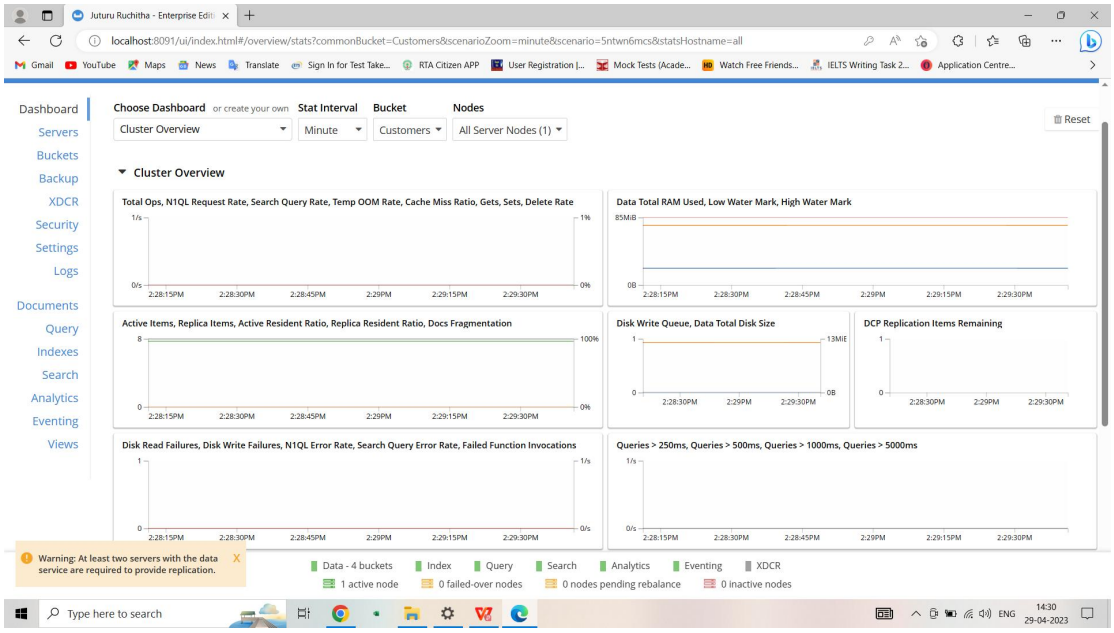
6)testing the proposed system :

During this stage, we make sure that the code will work as intended in our environment. We will address issues as we go in order to develop code that runs as quickly as possible.

7)Implementing the proposed system :

So that people may start using its capabilities and look for music whenever they want, we want to make this system accessible online.

Couchbase Dashboard:

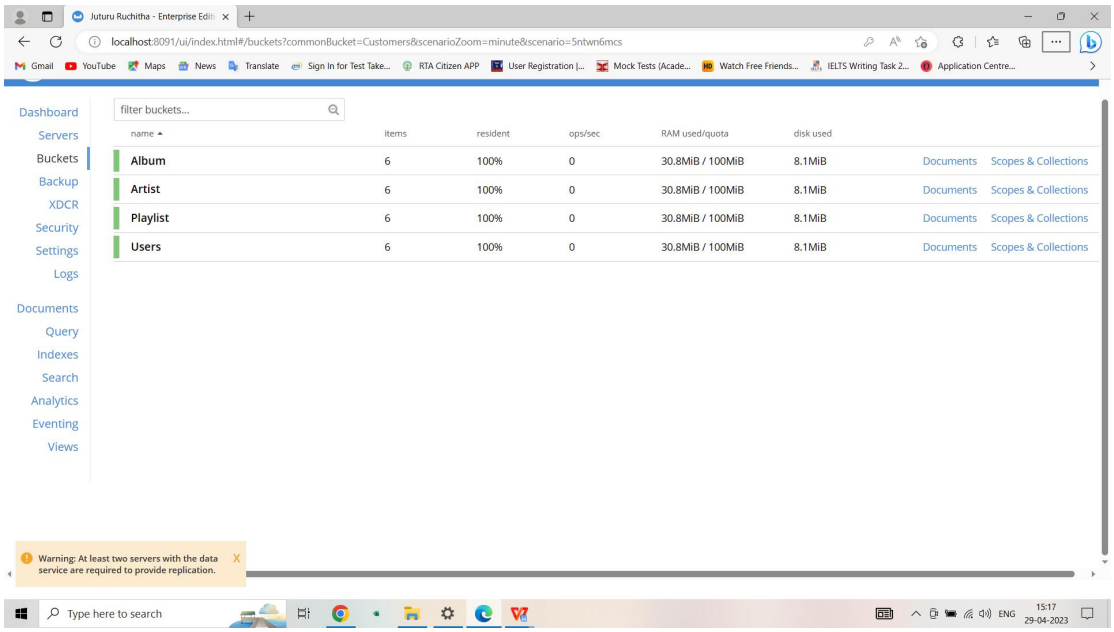


NoSQL based database implementation using Couchbase

Created the following buckets:

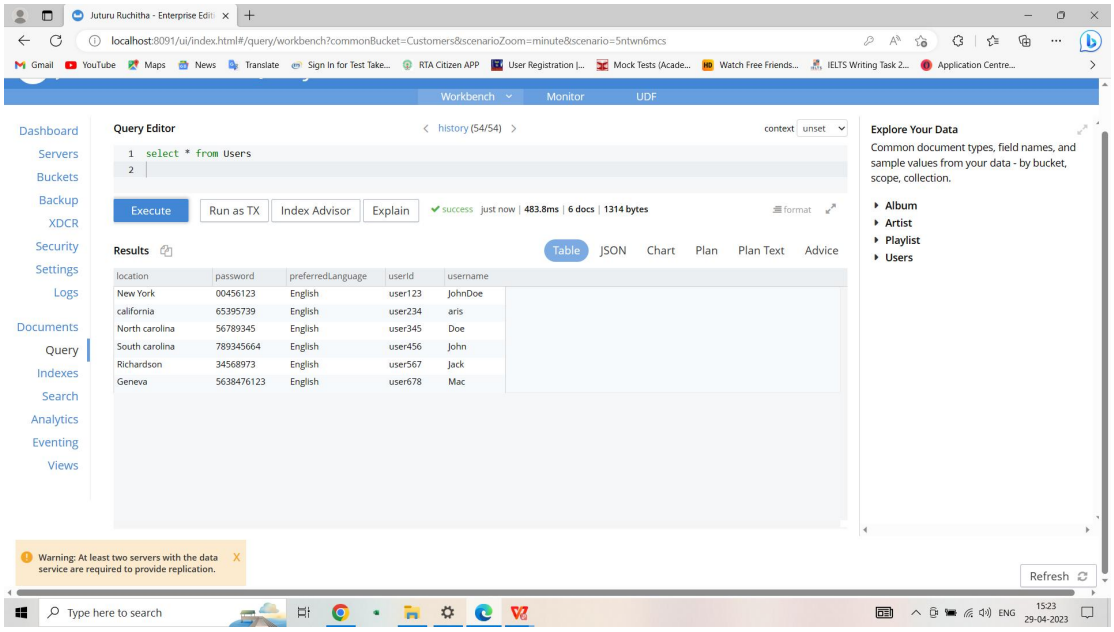
1. Users
2. Artist
3. Album
4. Playlist

Couchbase buckets view

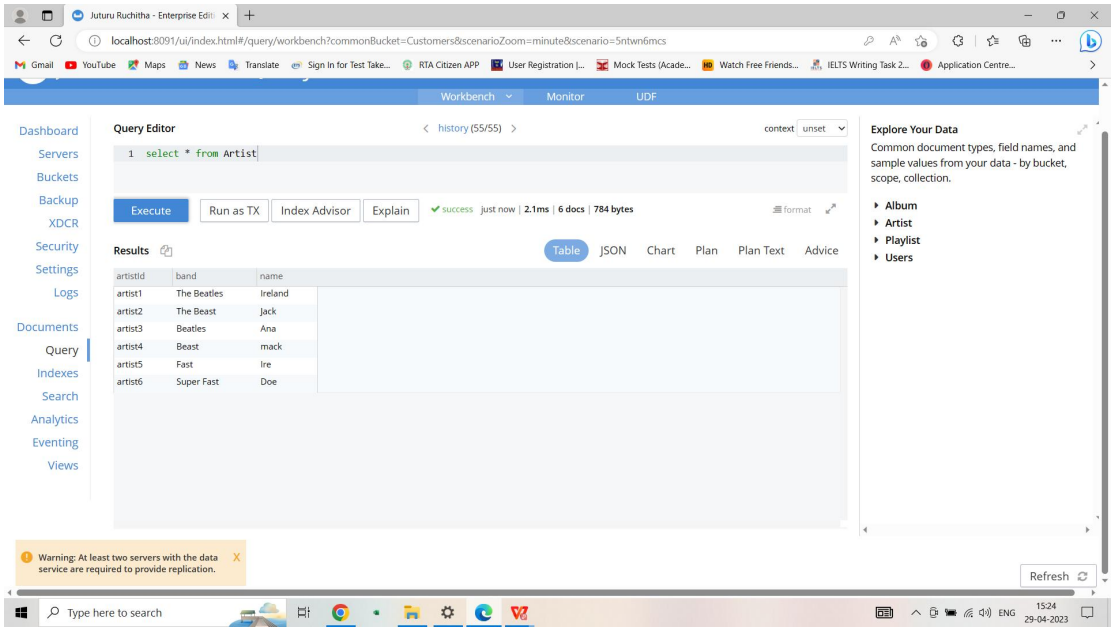


JSON Records:

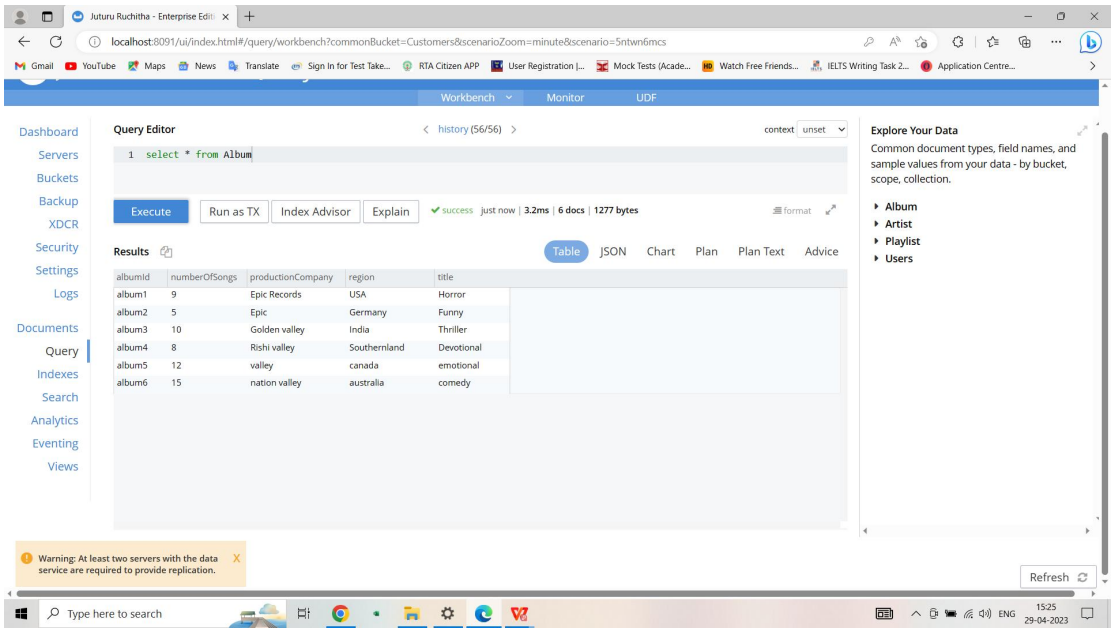
Users Bucket



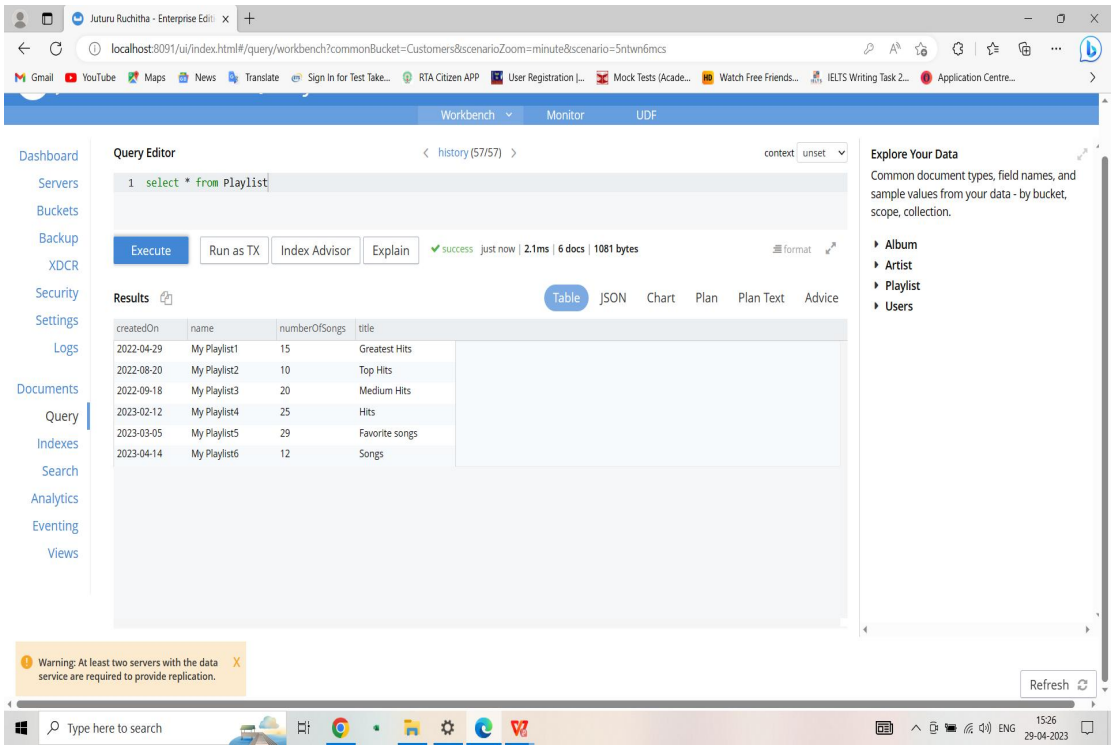
Artist Bucket



Album Bucket



Playlist Bucket

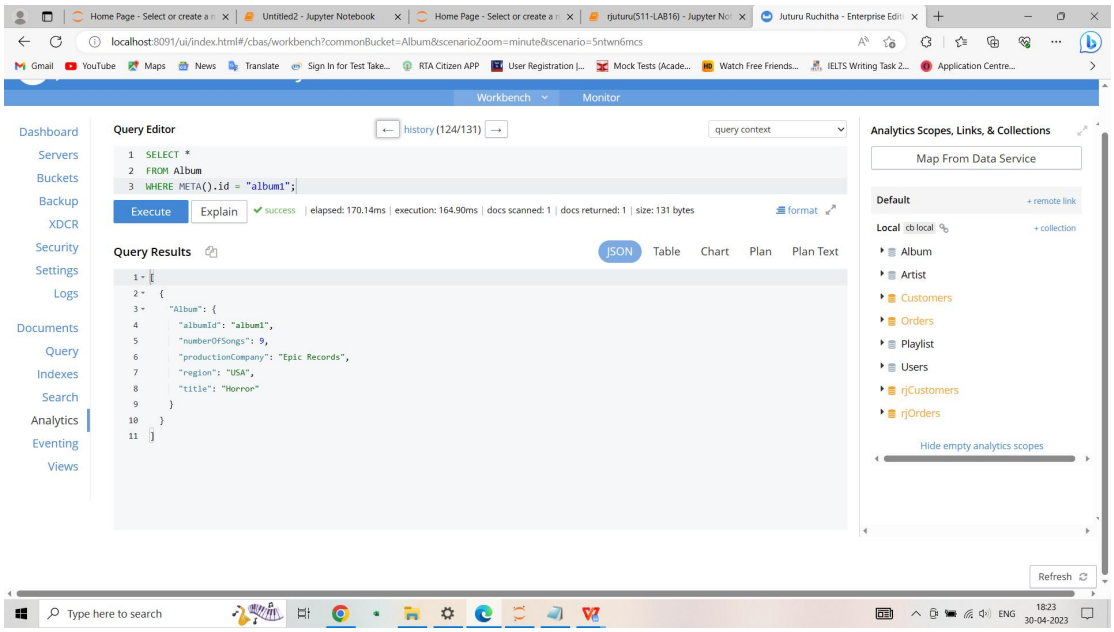


SQL++ queries:

What information is retrieved from the "Album" bucket where the ID is "album1"?

Query:

```
SELECT *
FROM Album
WHERE META().id = "album1";
```



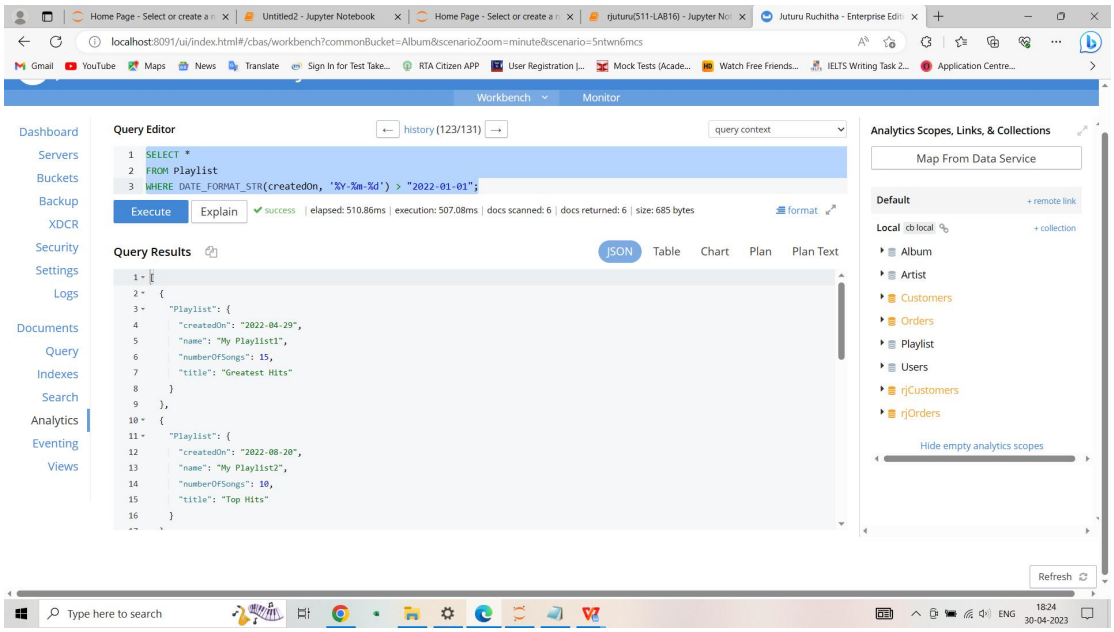
2)What are the playlists created after January 1st, 2022?

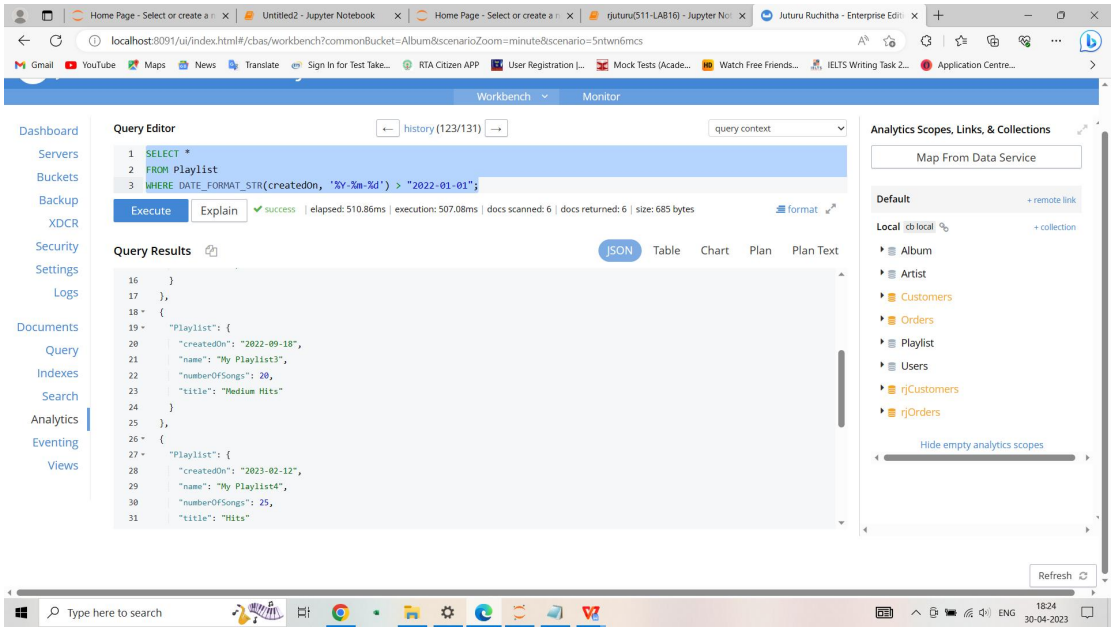
Query

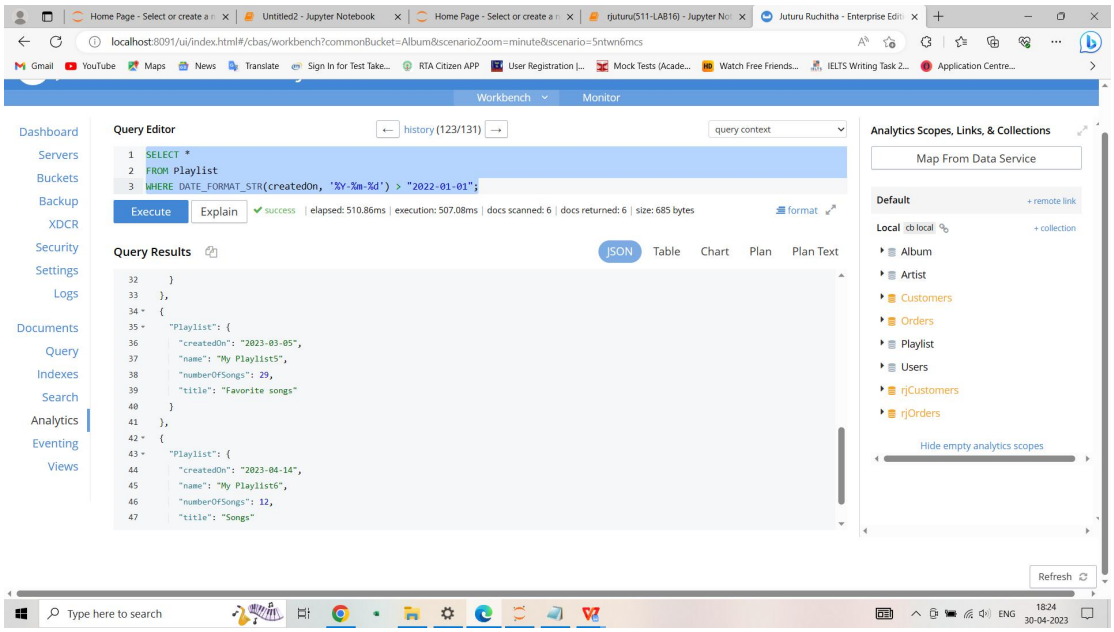
SELECT *

FROM Playlist

WHERE DATE_FORMAT_STR(createdOn, '%Y-%m-%d') > "2022-01-01";


Workbench
Monitor
Query Editor
SELECT *
FROM Playlist
WHERE DATE_FORMAT_STR(createdOn, '%Y-%m-%d') > "2022-01-01";
Execute
Explain
Query Results
JSON
Table
Chart
Plan
Plan Text
Analytics Scopes, Links, & Collections
Map From Data Service
Album
Artist
Customers
Orders
Playlist
Users
rjCustomers
rjOrders
Refresh


Workbench
Monitor
Query Editor
SELECT *
FROM Playlist
WHERE DATE_FORMAT_STR(createdOn, '%Y-%m-%d') > "2022-01-01";
Execute
Explain
Query Results
JSON
Table
Chart
Plan
Plan Text
Analytics Scopes, Links, & Collections
Map From Data Service
Album
Artist
Customers
Orders
Playlist
Users
rjCustomers
rjOrders
Refresh



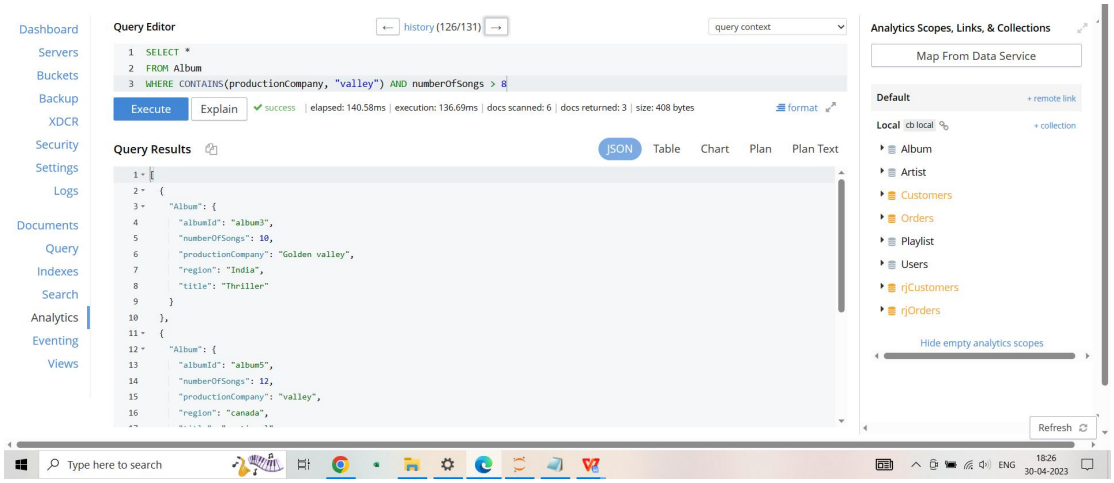
3) What is the minimum number of songs required for a row to be selected?

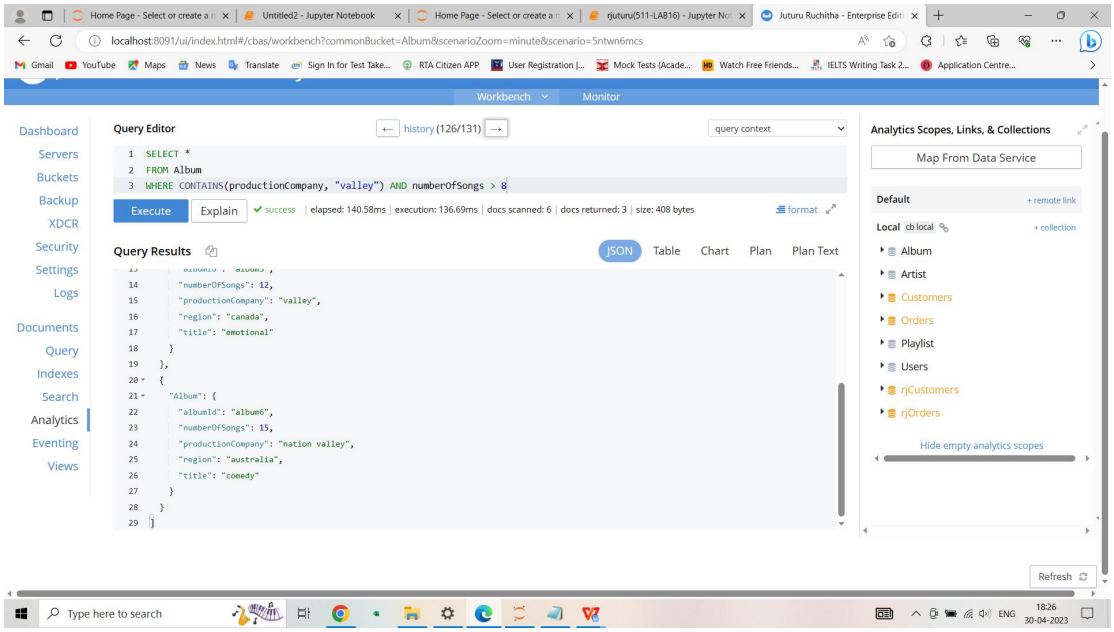
Query:

SELECT *

FROM Album

WHERE CONTAINS(productionCompany, "valley") AND numberOfSongs > 8





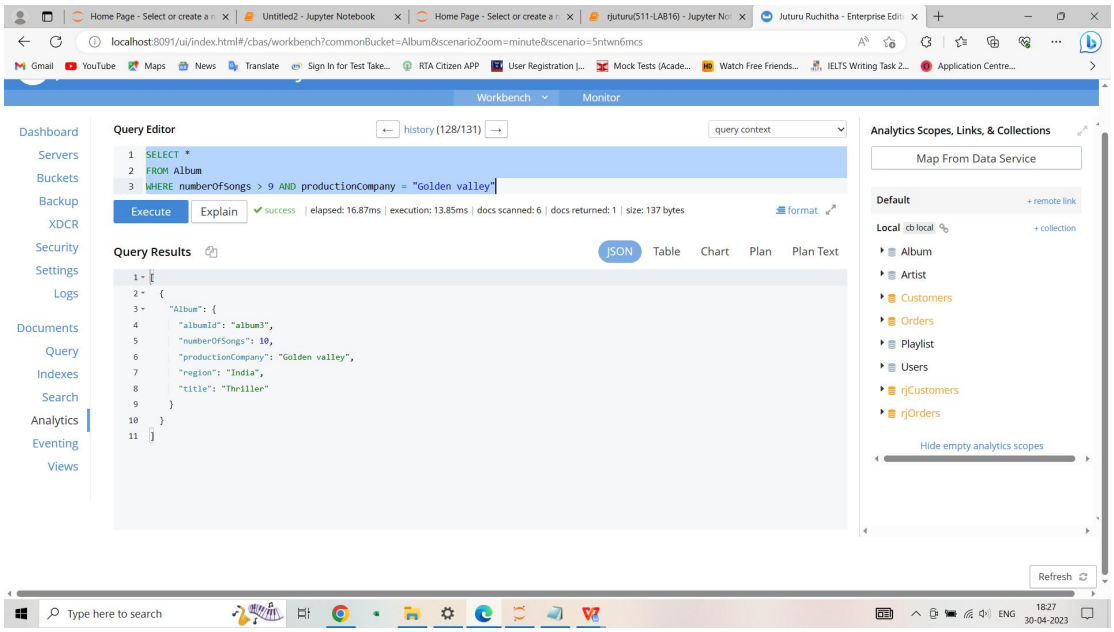
4.What is the total duration of all songs in the albums that satisfy the condition "numberOfSongs > 9 AND productionCompany = 'Golden valley'"?

Query:

SELECT *

FROM Album

WHERE numberOfSongs > 9 AND productionCompany = "Golden valley"


Workbench
Monitor
Dashboard
Servers
Buckets
Backup
XDCR
Security
Settings
Logs
Documents
Query
Indexes
Search
Analytics
Eventing
Views
Query Editor
history (128/131)
query context
SELECT *
FROM Album
WHERE numberOfSongs > 9 AND productionCompany = "Golden valley"
Execute
Explain
success | elapsed: 16.87ms | execution: 13.85ms | docs scanned: 6 | docs returned: 1 | size: 137 bytes
format
Query Results
JSON
Table
Chart
Plan
Plan Text
[
{
"Album": {
"albumId": "album3",
"numberOfSongs": 10,
"productionCompany": "Golden valley",
"region": "India",
"title": "Thriller"
}
}
]
Analytics Scopes, Links, & Collections
Map From Data Service
Default
+ remote link
Local
cb local
+ collection
Album
Artist
Customers
Orders
Playlist
Users
rjCustomers
rjOrders
Hide empty analytics scopes
Refresh
Type here to search
ENG
18:27
30-04-2023

Summary:

The system architecture makes sure that all of the questions in the questionnaire portion are accurately answered by using views, functions, stored procedures, and triggers. Understanding the data allowed us to give the users the pertinent information. It offers information about the numerous artists and the creations they have made. Users can also focus their list of favorite songs or information by using queries. The system might be used by users as a one-stop shop for all things musical.

Conclusion:

Our Music Library Management System does a good job of addressing the gaps we wish to fill and the constraints of the current systems. We discussed the importance of developing the system to be compatible with and easily expandable to new features in this environment that is always evolving, as well as the requirement for a music library platform that is mature and fully functional. It can be difficult to stay organized while building a database management system for the massive music industry because there is so much data available.

In order to identify the necessary entities, properties, and relationships between entities to construct an effective object role model during the design phase, before putting the database into use, we looked at a number of features and the operations of the leading music platforms at the time. After analyzing all of the requirements, we constructed our ORM diagram and turned the ORM into a relational model.

References

Halpin, T. (2015). Object-Role Modeling Fundamentals. Basking Ridge, NJ: Technics Publications.

Syverson, B., & Murach, J. (2016). Murach's SQL Server 2016 for Developers. Fresno, CA: Mike Murach & Associates, Inc.

W3 Schools. https://www.w3schools.com/sql/sql_stored_procedures.asp

W3 Schools. https://www.w3schools.blog/triggers-plsql