(CS 425-02) DATABASE ORGANISATION

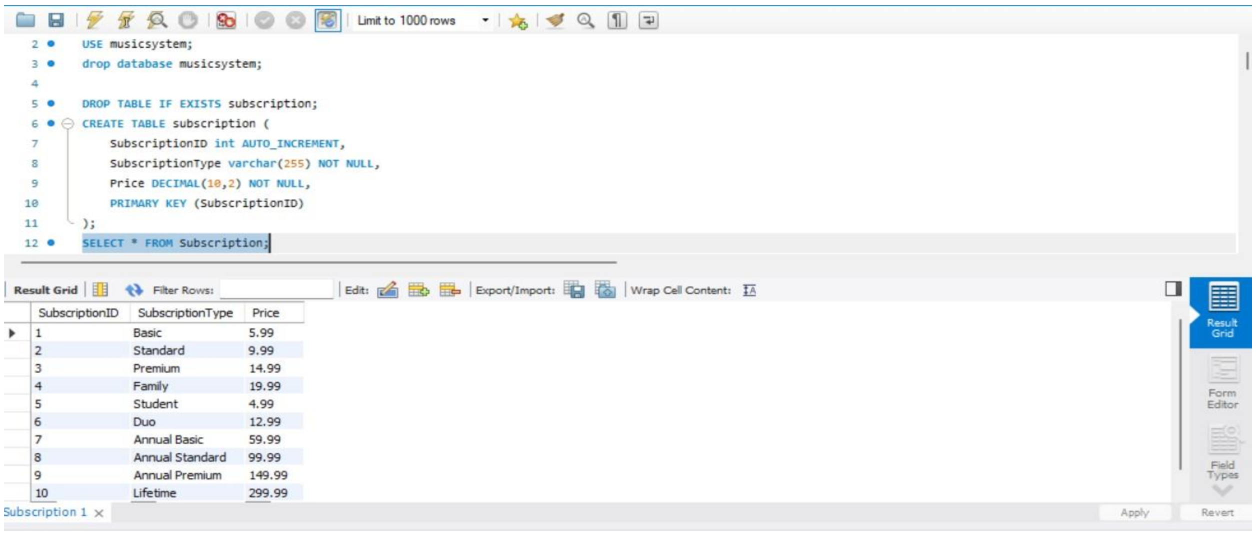
Homework 1.1

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

Deliverable 2

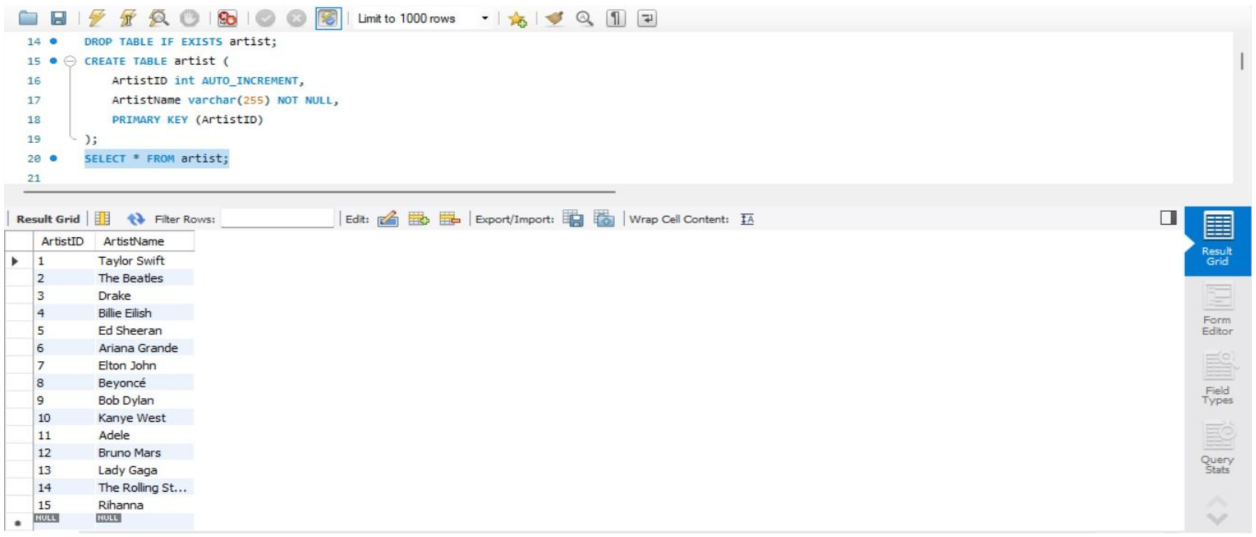
1) CREATE TABLE subscription (
    SubscriptionID int AUTO_INCREMENT,
    SubscriptionType varchar(255) NOT NULL,
    Price DECIMAL(10,2) NOT NULL,
    PRIMARY KEY (SubscriptionID)
);



```
USE musicsystem;
drop database musicsystem;

DROP TABLE IF EXISTS subscription;
CREATE TABLE subscription (
    SubscriptionID int AUTO_INCREMENT,
    SubscriptionType varchar(255) NOT NULL,
    Price DECIMAL(10,2) NOT NULL,
    PRIMARY KEY (SubscriptionID)
);
SELECT * FROM Subscription;
```

SubscriptionID	SubscriptionType	Price
1	Basic	5.99
2	Standard	9.99
3	Premium	14.99
4	Family	19.99
5	Student	4.99
6	Duo	12.99
7	Annual Basic	59.99
8	Annual Standard	99.99
9	Annual Premium	149.99
10	Lifetime	299.99

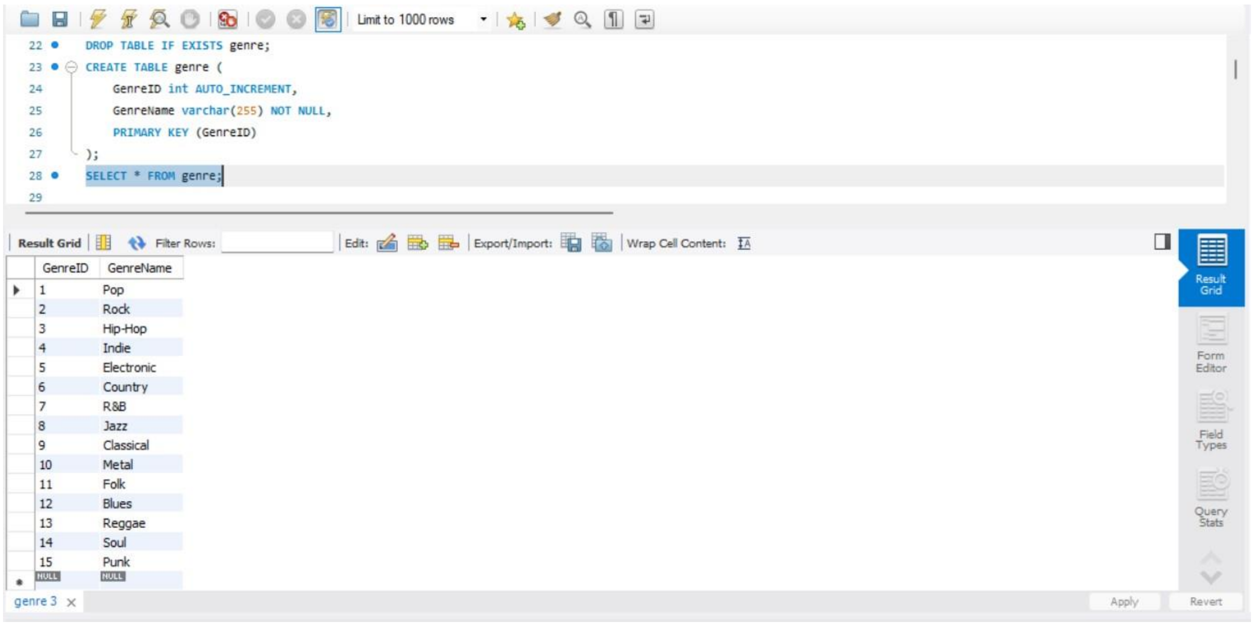
2) CREATE TABLE artist (
    ArtistID int AUTO_INCREMENT,
    ArtistName varchar(255) NOT NULL,
    PRIMARY KEY (ArtistID)
);



```
DROP TABLE IF EXISTS artist;
CREATE TABLE artist (
    ArtistID int AUTO_INCREMENT,
    ArtistName varchar(255) NOT NULL,
    PRIMARY KEY (ArtistID)
);
SELECT * FROM artist;
```

ArtistID	ArtistName
1	Taylor Swift
2	The Beatles
3	Drake
4	Billie Eilish
5	Ed Sheeran
6	Ariana Grande
7	Elton John
8	Beyoncé
9	Bob Dylan
10	Kanye West
11	Adele
12	Bruno Mars
13	Lady Gaga
14	The Rolling St...
15	Rihanna
NULL	NULL

3) CREATE TABLE genre (
    GenreID int AUTO_INCREMENT,
    GenreName varchar(255) NOT NULL,
    PRIMARY KEY (GenreID)
  );

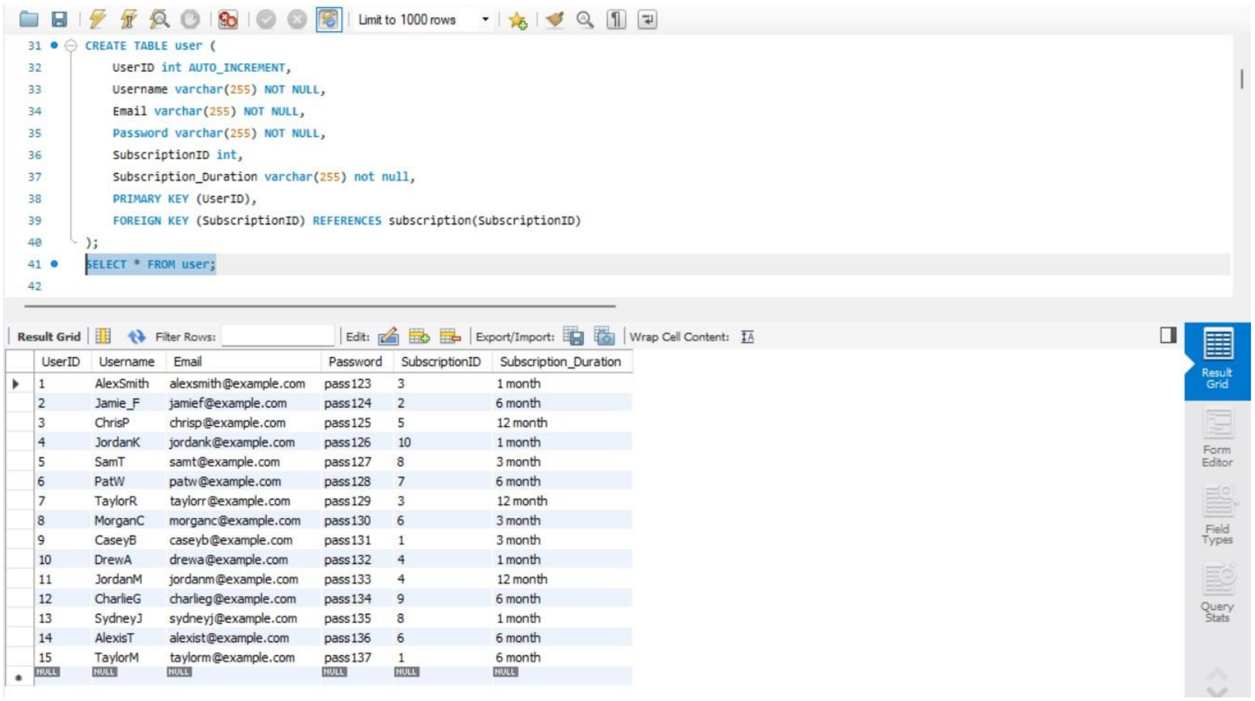


```
DROP TABLE IF EXISTS genre;
CREATE TABLE genre (
    GenreID int AUTO_INCREMENT,
    GenreName varchar(255) NOT NULL,
    PRIMARY KEY (GenreID)
);
SELECT * FROM genre;
```

GenreID	GenreName
1	Pop
2	Rock
3	Hip-Hop
4	Indie
5	Electronic
6	Country
7	R&B
8	Jazz
9	Classical
10	Metal
11	Folk
12	Blues
13	Reggae
14	Soul
15	Punk
NULL	NULL

4) CREATE TABLE user (

```
    UserID int AUTO_INCREMENT,
    Username varchar(255) NOT NULL,
    Email varchar(255) NOT NULL,
    Password varchar(255) NOT NULL,
    SubscriptionID int,
    Subscription_Duration varchar(255) not null,
    PRIMARY KEY (UserID),
    FOREIGN KEY (SubscriptionID) REFERENCES subscription(SubscriptionID)
);
```

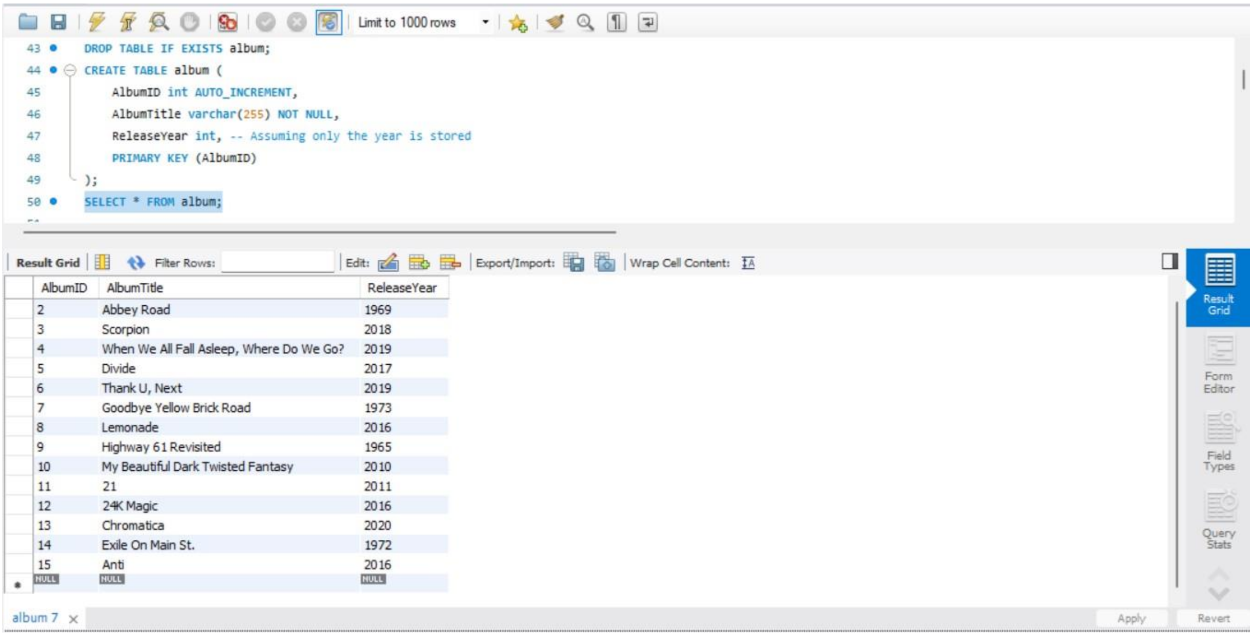


```
CREATE TABLE user (
    UserID int AUTO_INCREMENT,
    Username varchar(255) NOT NULL,
    Email varchar(255) NOT NULL,
    Password varchar(255) NOT NULL,
    SubscriptionID int,
    Subscription_Duration varchar(255) not null,
    PRIMARY KEY (UserID),
    FOREIGN KEY (SubscriptionID) REFERENCES subscription(SubscriptionID)
);
SELECT * FROM user;
```

UserID	Username	Email	Password	SubscriptionID	Subscription_Duration
1	AlexSmith	alexsmith@example.com	pass123	3	1 month
2	Jamie_F	jamief@example.com	pass124	2	6 month
3	ChrisP	chrisp@example.com	pass125	5	12 month
4	JordanK	jordank@example.com	pass126	10	1 month
5	SamT	samt@example.com	pass127	8	3 month
6	PatW	patw@example.com	pass128	7	6 month
7	TaylorR	taylorr@example.com	pass129	3	12 month
8	MorganC	morganc@example.com	pass130	6	3 month
9	CaseyB	caseyb@example.com	pass131	1	3 month
10	DrewA	drewa@example.com	pass132	4	1 month
11	JordanM	jordanm@example.com	pass133	4	12 month
12	CharlieG	charlieg@example.com	pass134	9	6 month
13	SydneyJ	sydneyj@example.com	pass135	8	1 month
14	AlexisT	alexist@example.com	pass136	6	6 month
15	TaylorM	taylorm@example.com	pass137	1	6 month
NULL	NULL	NULL	NULL	NULL	NULL

5) CREATE TABLE album (

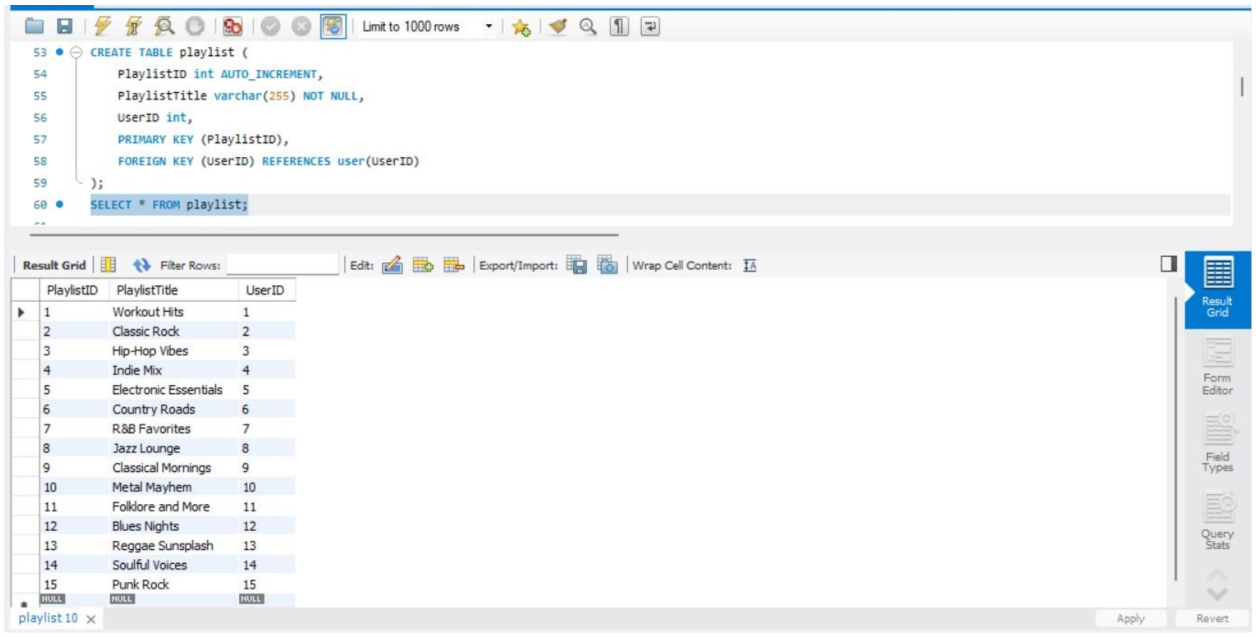
```
    AlbumID int AUTO_INCREMENT,
    AlbumTitle varchar(255) NOT NULL,
    ReleaseYear int, -- Assuming only the year is stored
    PRIMARY KEY (AlbumID)
);
```



```
DROP TABLE IF EXISTS album;
CREATE TABLE album (
    AlbumID int AUTO_INCREMENT,
    AlbumTitle varchar(255) NOT NULL,
    ReleaseYear int, -- Assuming only the year is stored
    PRIMARY KEY (AlbumID)
);
SELECT * FROM album;
```

AlbumID	AlbumTitle	ReleaseYear
2	Abbey Road	1969
3	Scorpion	2018
4	When We All Fall Asleep, Where Do We Go?	2019
5	Divide	2017
6	Thank U, Next	2019
7	Goodbye Yellow Brick Road	1973
8	Lemonade	2016
9	Highway 61 Revisited	1965
10	My Beautiful Dark Twisted Fantasy	2010
11	21	2011
12	24K Magic	2016
13	Chromatica	2020
14	Exile On Main St.	1972
15	Anti	2016
NULL	NULL	NULL

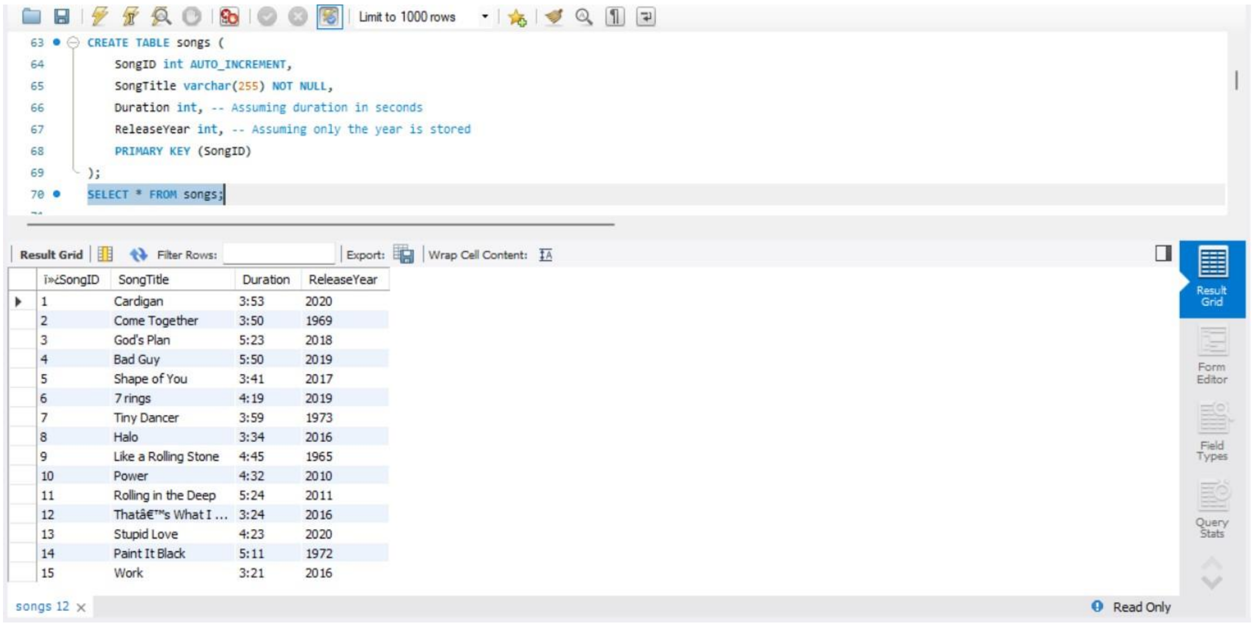
6) CREATE TABLE playlist (
    PlaylistID int AUTO_INCREMENT,
    PlaylistTitle varchar(255) NOT NULL,
    UserID int,
    PRIMARY KEY (PlaylistID),
    FOREIGN KEY (UserID) REFERENCES user(UserID)
);



```
CREATE TABLE playlist (
    PlaylistID int AUTO_INCREMENT,
    PlaylistTitle varchar(255) NOT NULL,
    UserID int,
    PRIMARY KEY (PlaylistID),
    FOREIGN KEY (UserID) REFERENCES user(UserID)
);
SELECT * FROM playlist;
```

PlaylistID	PlaylistTitle	UserID
1	Workout Hits	1
2	Classic Rock	2
3	Hip-Hop Vibes	3
4	Indie Mix	4
5	Electronic Essentials	5
6	Country Roads	6
7	R&B Favorites	7
8	Jazz Lounge	8
9	Classical Mornings	9
10	Metal Mayhem	10
11	Folklore and More	11
12	Blues Nights	12
13	Reggae Sunsplash	13
14	Soulful Voices	14
15	Punk Rock	15
NULL	NULL	NULL

7)
```
CREATE TABLE songs (
    SongID int AUTO_INCREMENT,
    SongTitle varchar(255) NOT NULL,
    Duration int, -- Assuming duration in seconds
    ReleaseYear int, -- Assuming only the year is stored
    PRIMARY KEY (SongID)
);
```



```
CREATE TABLE songs (
    SongID int AUTO_INCREMENT,
    SongTitle varchar(255) NOT NULL,
    Duration int, -- Assuming duration in seconds
    ReleaseYear int, -- Assuming only the year is stored
    PRIMARY KEY (SongID)
);
SELECT * FROM songs;
```

ï»¿SongID	SongTitle	Duration	ReleaseYear
1	Cardigan	3:53	2020
2	Come Together	3:50	1969
3	God's Plan	5:23	2018
4	Bad Guy	5:50	2019
5	Shape of You	3:41	2017
6	7 rings	4:19	2019
7	Tiny Dancer	3:59	1973
8	Halo	3:34	2016
9	Like a Rolling Stone	4:45	1965
10	Power	4:32	2010
11	Rolling in the Deep	5:24	2011
12	Thatâ€™s What I ...	3:24	2016
13	Stupid Love	4:23	2020
14	Paint It Black	5:11	1972
15	Work	3:21	2016

8) CREATE TABLE SongPlaylist (

SongID int NOT NULL,

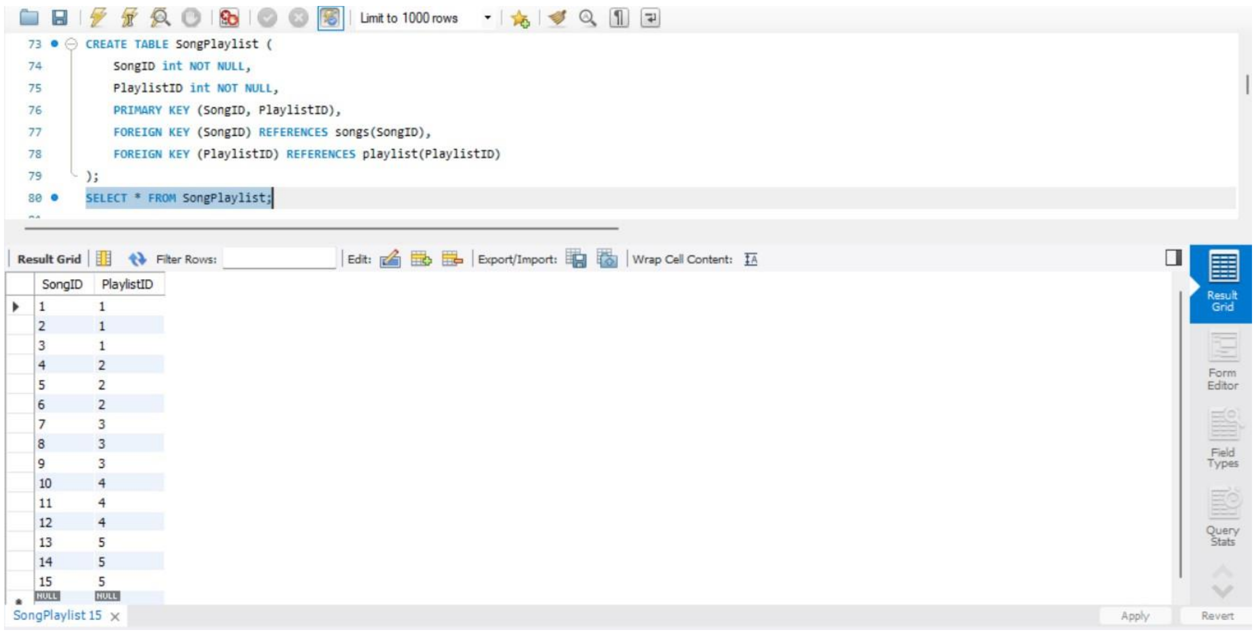
PlaylistID int NOT NULL,

PRIMARY KEY (SongID, PlaylistID),

FOREIGN KEY (SongID) REFERENCES songs(SongID),

FOREIGN KEY (PlaylistID) REFERENCES playlist(PlaylistID)

);



```
CREATE TABLE SongPlaylist (
    SongID int NOT NULL,
    PlaylistID int NOT NULL,
    PRIMARY KEY (SongID, PlaylistID),
    FOREIGN KEY (SongID) REFERENCES songs(SongID),
    FOREIGN KEY (PlaylistID) REFERENCES playlist(PlaylistID)
);
SELECT * FROM SongPlaylist;
```

SongID	PlaylistID
1	1
2	1
3	1
4	2
5	2
6	2
7	3
8	3
9	3
10	4
11	4
12	4
13	5
14	5
15	5
NULL	NULL

9) CREATE TABLE SongArtist (

SongID int NOT NULL,

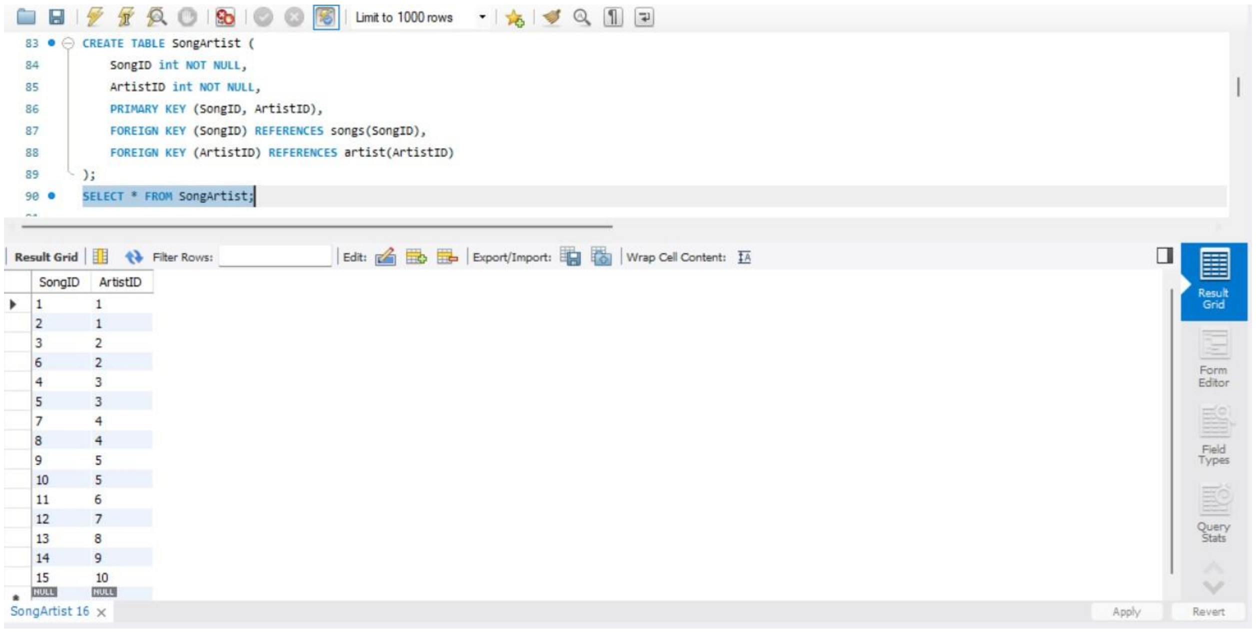
ArtistID int NOT NULL,

PRIMARY KEY (SongID, ArtistID),

FOREIGN KEY (SongID) REFERENCES songs(SongID),

FOREIGN KEY (ArtistID) REFERENCES artist(ArtistID)

);



```
CREATE TABLE SongArtist (
    SongID int NOT NULL,
    ArtistID int NOT NULL,
    PRIMARY KEY (SongID, ArtistID),
    FOREIGN KEY (SongID) REFERENCES songs(SongID),
    FOREIGN KEY (ArtistID) REFERENCES artist(ArtistID)
);
SELECT * FROM SongArtist;
```

SongID	ArtistID
1	1
2	1
3	2
6	2
4	3
5	3
7	4
8	4
9	5
10	5
11	6
12	7
13	8
14	9
15	10
NULL	NULL

10) CREATE TABLE AlbumGenre (

AlbumID int NOT NULL,

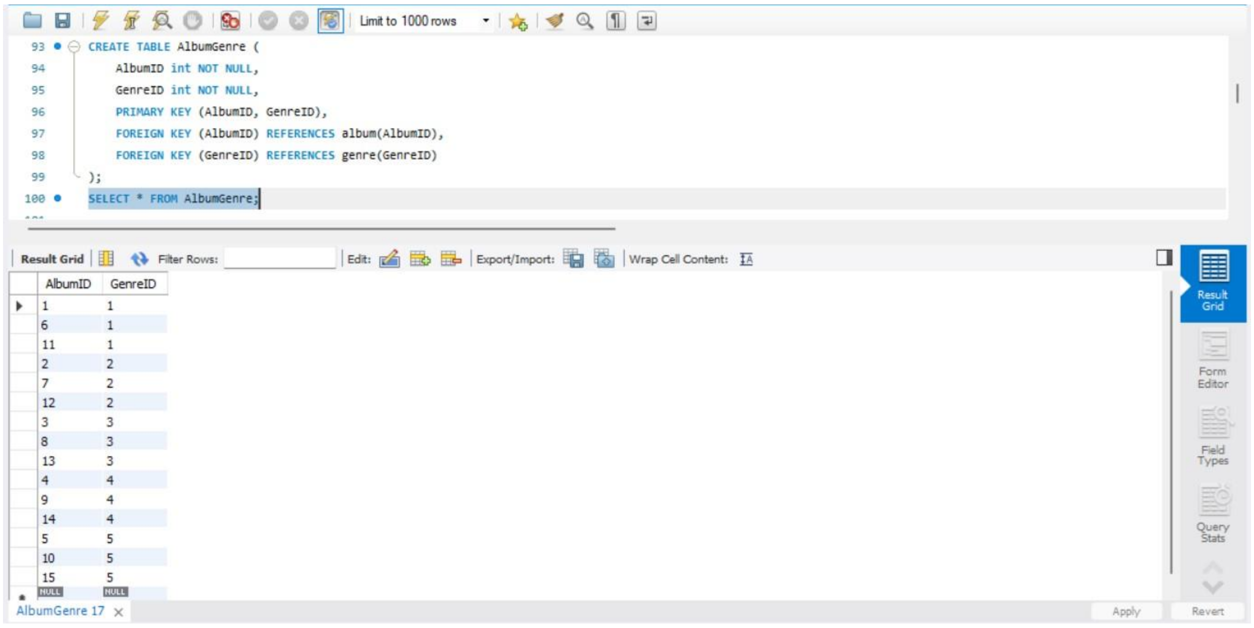
GenreID int NOT NULL,

PRIMARY KEY (AlbumID, GenreID),

FOREIGN KEY (AlbumID) REFERENCES album(AlbumID),

FOREIGN KEY (GenreID) REFERENCES genre(GenreID)

);



```
CREATE TABLE AlbumGenre (
    AlbumID int NOT NULL,
    GenreID int NOT NULL,
    PRIMARY KEY (AlbumID, GenreID),
    FOREIGN KEY (AlbumID) REFERENCES album(AlbumID),
    FOREIGN KEY (GenreID) REFERENCES genre(GenreID)
);
SELECT * FROM AlbumGenre;
```

AlbumID	GenreID
1	1
6	1
11	1
2	2
7	2
12	2
3	3
8	3
13	3
4	4
9	4
14	4
5	5
10	5
15	5
NULL	NULL

11) CREATE TABLE SongGenre (

SongID int NOT NULL,

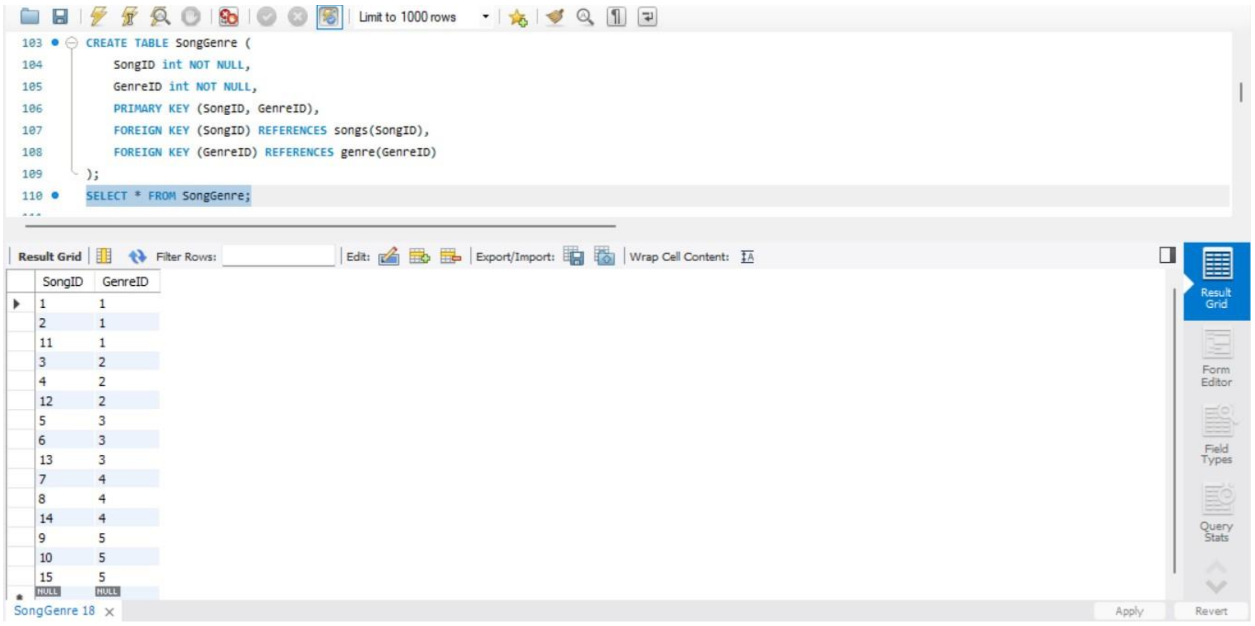
GenreID int NOT NULL,

PRIMARY KEY (SongID, GenreID),

FOREIGN KEY (SongID) REFERENCES songs(SongID),

FOREIGN KEY (GenreID) REFERENCES genre(GenreID)

);



```
CREATE TABLE SongGenre (
    SongID int NOT NULL,
    GenreID int NOT NULL,
    PRIMARY KEY (SongID, GenreID),
    FOREIGN KEY (SongID) REFERENCES songs(SongID),
    FOREIGN KEY (GenreID) REFERENCES genre(GenreID)
);
SELECT * FROM SongGenre;
```

SongID	GenreID
1	1
2	1
11	1
3	2
4	2
12	2
5	3
6	3
13	3
7	4
8	4
14	4
9	5
10	5
15	5
NULL	NULL

12) CREATE TABLE SongAlbum (

SongID int NOT NULL,

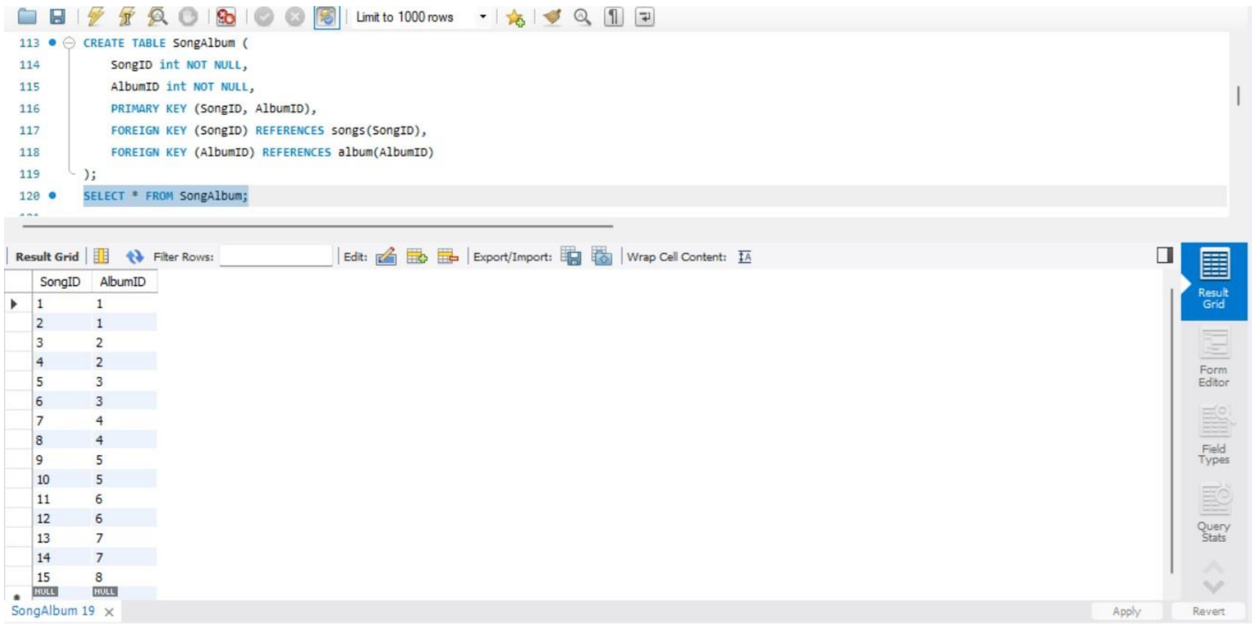
AlbumID int NOT NULL,

PRIMARY KEY (SongID, AlbumID),

FOREIGN KEY (SongID) REFERENCES songs(SongID),

FOREIGN KEY (AlbumID) REFERENCES album(AlbumID)

);



```
CREATE TABLE SongAlbum (
    SongID int NOT NULL,
    AlbumID int NOT NULL,
    PRIMARY KEY (SongID, AlbumID),
    FOREIGN KEY (SongID) REFERENCES songs(SongID),
    FOREIGN KEY (AlbumID) REFERENCES album(AlbumID)
);
SELECT * FROM SongAlbum;
```

SongID	AlbumID
1	1
2	1
3	2
4	2
5	3
6	3
7	4
8	4
9	5
10	5
11	6
12	6
13	7
14	7
15	8
NULL	NULL

13) CREATE TABLE ArtistAlbum (

ArtistID int NOT NULL,

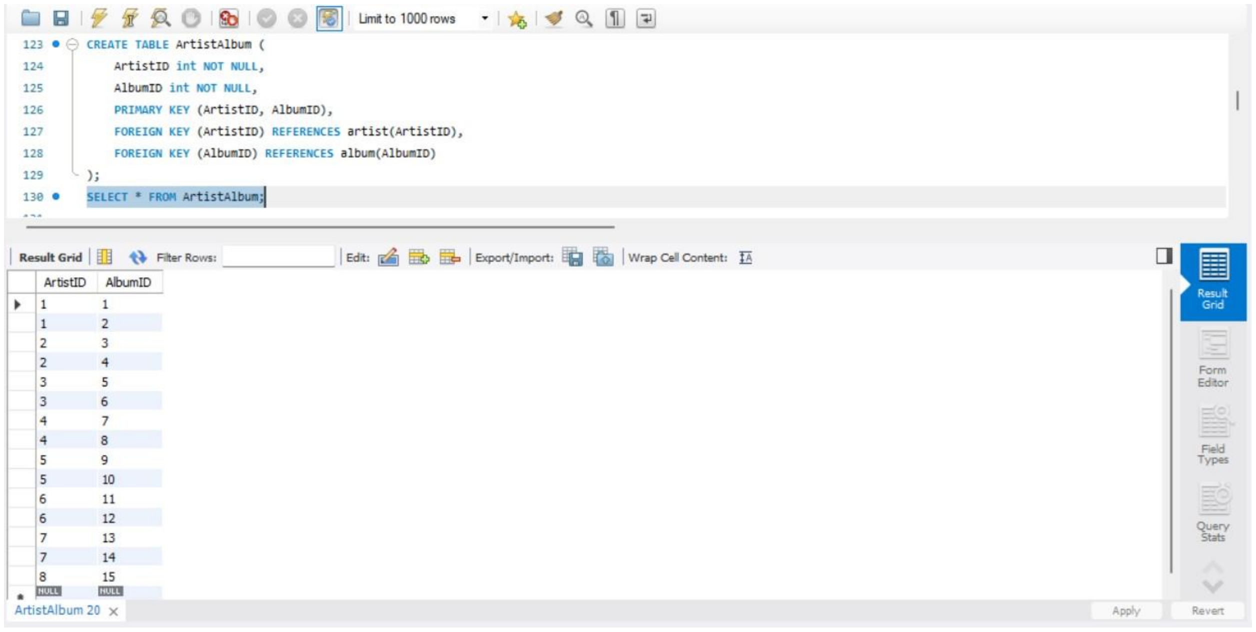
AlbumID int NOT NULL,

PRIMARY KEY (ArtistID, AlbumID),

FOREIGN KEY (ArtistID) REFERENCES artist(ArtistID),

FOREIGN KEY (AlbumID) REFERENCES album(AlbumID)

);



```
CREATE TABLE ArtistAlbum (
    ArtistID int NOT NULL,
    AlbumID int NOT NULL,
    PRIMARY KEY (ArtistID, AlbumID),
    FOREIGN KEY (ArtistID) REFERENCES artist(ArtistID),
    FOREIGN KEY (AlbumID) REFERENCES album(AlbumID)
);
SELECT * FROM ArtistAlbum;
```

ArtistID	AlbumID
1	1
1	2
2	3
2	4
3	5
3	6
4	7
4	8
5	9
5	10
6	11
6	12
7	13
7	14
8	15
NULL	NULL

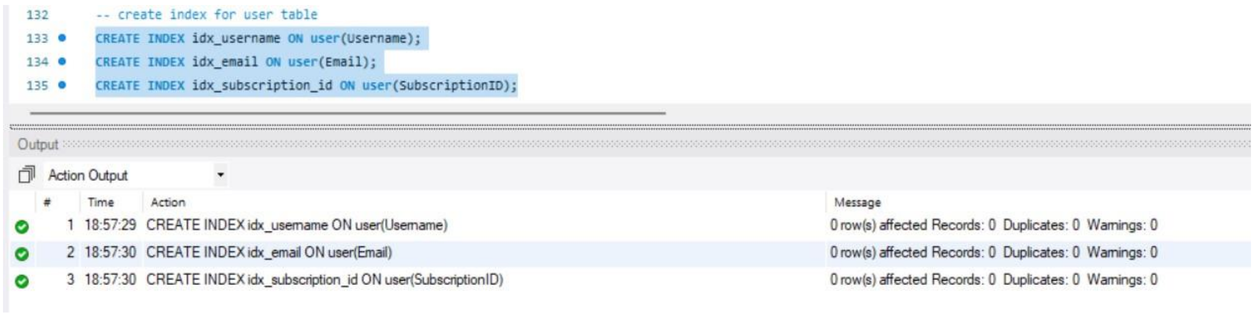
Creating Index:

-- create index for user table

CREATE INDEX idx_username ON user(Username);

CREATE INDEX idx_email ON user(Email);

CREATE INDEX idx_subscription_id ON user(SubscriptionID);

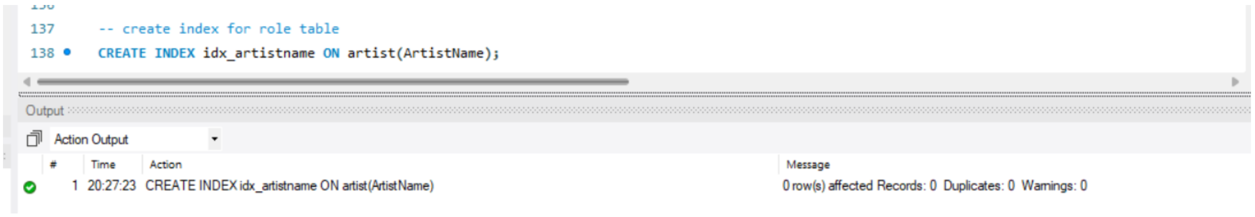


```
-- create index for user table
CREATE INDEX idx_username ON user(Username);
CREATE INDEX idx_email ON user(Email);
CREATE INDEX idx_subscription_id ON user(SubscriptionID);
```

```
Output
Action Output
#  Time      Action                                                  Message
1  18:57:29  CREATE INDEX idx_usemame ON user(Usemame)               0 row(s) affected Records: 0  Duplicates: 0  Wamings: 0
2  18:57:30  CREATE INDEX idx_email ON user(Email)                   0 row(s) affected Records: 0  Duplicates: 0  Wamings: 0
3  18:57:30  CREATE INDEX idx_subscription_id ON user(SubscriptionID)  0 row(s) affected Records: 0  Duplicates: 0  Wamings: 0
```

-- create index for role table

CREATE INDEX idx_artist_name ON artist(ArtistName);

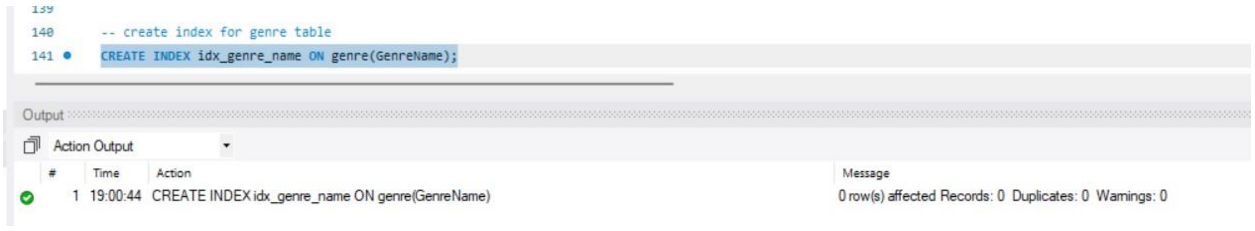


```
-- create index for role table
CREATE INDEX idx_artistname ON artist(ArtistName);
```

```
Output
Action Output
#  Time      Action                                         Message
1  20:27:23  CREATE INDEX idx_artistname ON artist(ArtistName)  0 row(s) affected Records: 0  Duplicates: 0  Wamings: 0
```

-- create index for genre table

CREATE INDEX idx_genre_name ON genre(GenreName);



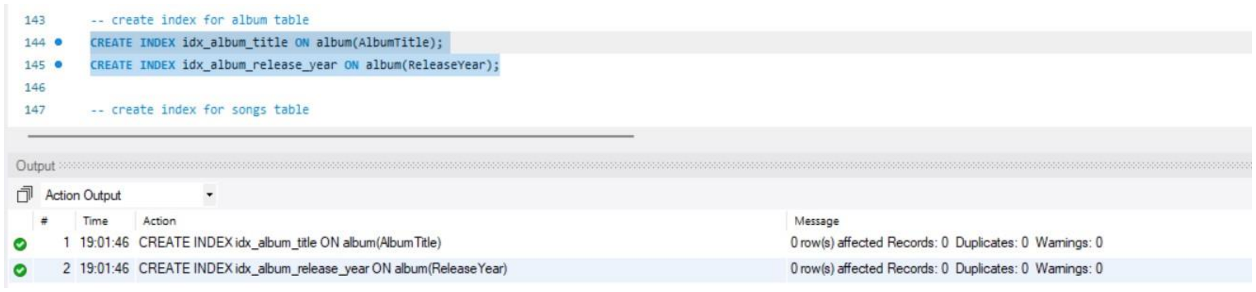
```
-- create index for genre table
CREATE INDEX idx_genre_name ON genre(GenreName);
```

```
Output
Action Output
#  Time      Action                                          Message
1  19:00:44  CREATE INDEX idx_genre_name ON genre(GenreName)  0 row(s) affected Records: 0  Duplicates: 0  Wamings: 0
```

-- create index for album table

CREATE INDEX idx_album_title ON album(AlbumTitle);

CREATE INDEX idx_album_release_year ON album(ReleaseYear);



```
-- create index for album table
CREATE INDEX idx_album_title ON album(AlbumTitle);
CREATE INDEX idx_album_release_year ON album(ReleaseYear);

-- create index for songs table
```

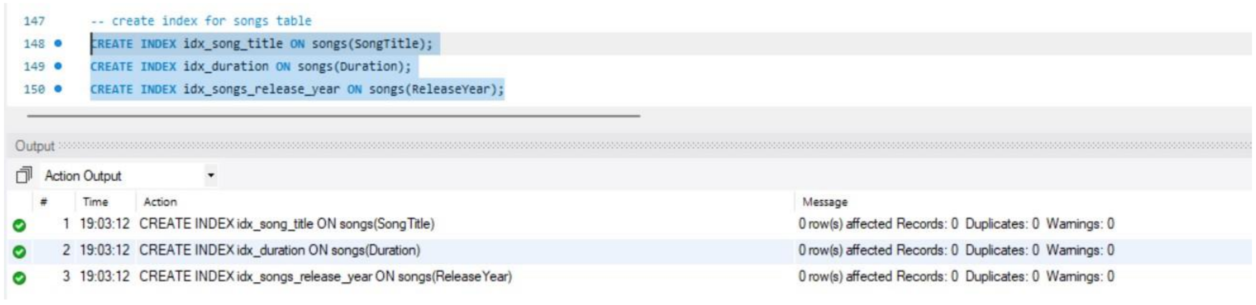
```
Output
Action Output
 #  Time      Action                                                  Message
 1  19:01:46  CREATE INDEX idx_album_title ON album(AlbumTitle)        0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
 2  19:01:46  CREATE INDEX idx_album_release_year ON album(ReleaseYear) 0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
```

-- create index for songs table

CREATE INDEX idx_song_title ON songs(SongTitle);

CREATE INDEX idx_duration ON songs(Duration);

CREATE INDEX idx_songs_release_year ON songs(ReleaseYear);



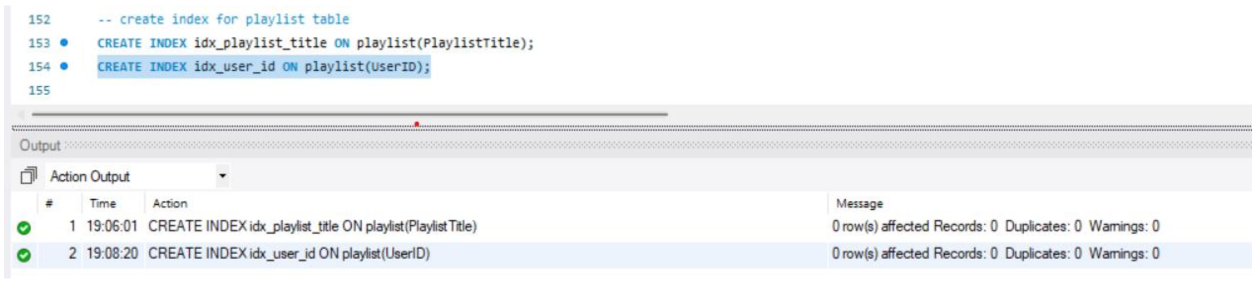
```
-- create index for songs table
CREATE INDEX idx_song_title ON songs(SongTitle);
CREATE INDEX idx_duration ON songs(Duration);
CREATE INDEX idx_songs_release_year ON songs(ReleaseYear);
```

```
Output
Action Output
 #  Time      Action                                                   Message
 1  19:03:12  CREATE INDEX idx_song_title ON songs(SongTitle)           0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
 2  19:03:12  CREATE INDEX idx_duration ON songs(Duration)              0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
 3  19:03:12  CREATE INDEX idx_songs_release_year ON songs(ReleaseYear) 0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
```

-- create index for playlist table

CREATE INDEX idx_playlist_title ON playlist(PlaylistTitle);

CREATE INDEX idx_user_id ON playlist(UserID);

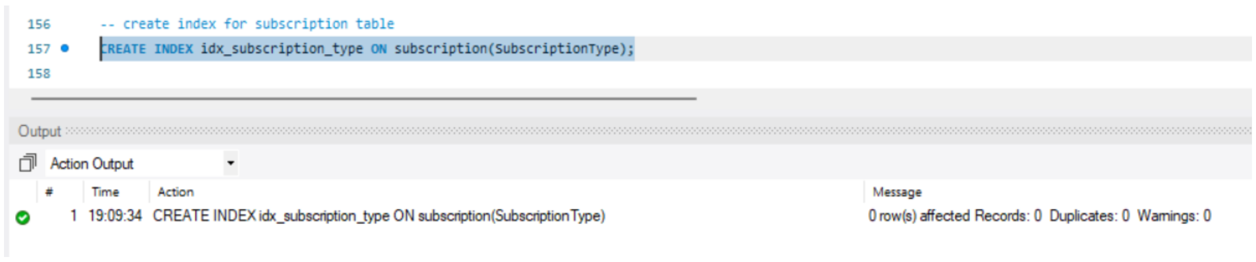


```
-- create index for playlist table
CREATE INDEX idx_playlist_title ON playlist(PlaylistTitle);
CREATE INDEX idx_user_id ON playlist(UserID);

```

```
Output
Action Output
 #  Time      Action                                                   Message
 1  19:06:01  CREATE INDEX idx_playlist_title ON playlist(PlaylistTitle) 0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
 2  19:08:20  CREATE INDEX idx_user_id ON playlist(UserID)              0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
```

-- create index for subscription table

CREATE INDEX idx_subscription_type ON subscription(SubscriptionType);



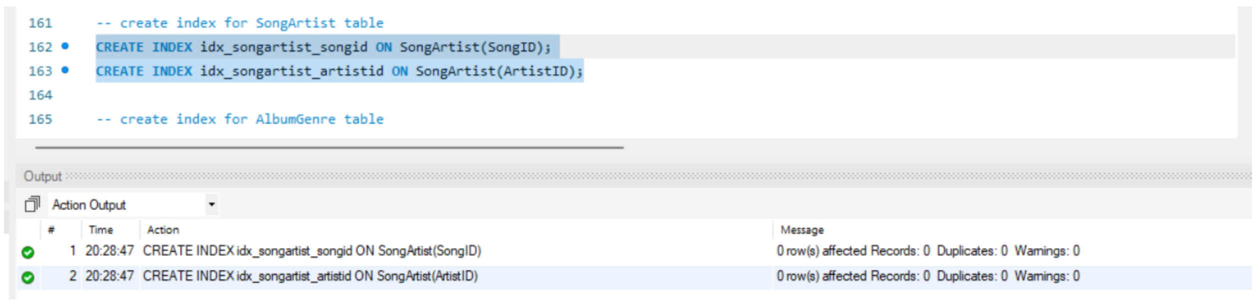
```
-- create index for subscription table
CREATE INDEX idx_subscription_type ON subscription(SubscriptionType);
```

```
Output
Action Output
#  Time      Action                                                          Message
1  19:09:34  CREATE INDEX idx_subscription_type ON subscription(SubscriptionType)  0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
```

-- create index for SongArtist table

CREATE INDEX idx_songartist_songid ON SongArtist(SongID);

CREATE INDEX idx_songartist_artistid ON SongArtist(ArtistID);



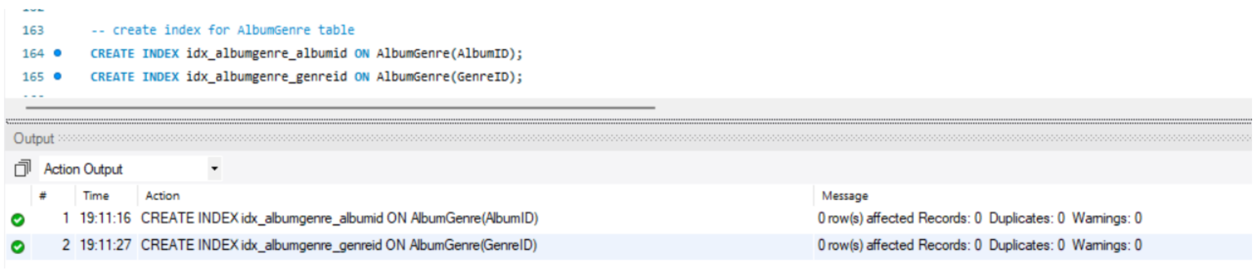
```
-- create index for SongArtist table
CREATE INDEX idx_songartist_songid ON SongArtist(SongID);
CREATE INDEX idx_songartist_artistid ON SongArtist(ArtistID);

-- create index for AlbumGenre table
```

```
Output
Action Output
#  Time      Action                                                      Message
1  20:28:47  CREATE INDEX idx_songartist_songid ON SongArtist(SongID)      0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
2  20:28:47  CREATE INDEX idx_songartist_artistid ON SongArtist(ArtistID)  0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
```

-- create index for AlbumGenre table

CREATE INDEX idx_albumgenre_albumid ON AlbumGenre(AlbumID);

CREATE INDEX idx_albumgenre_genreid ON AlbumGenre(GenreID);



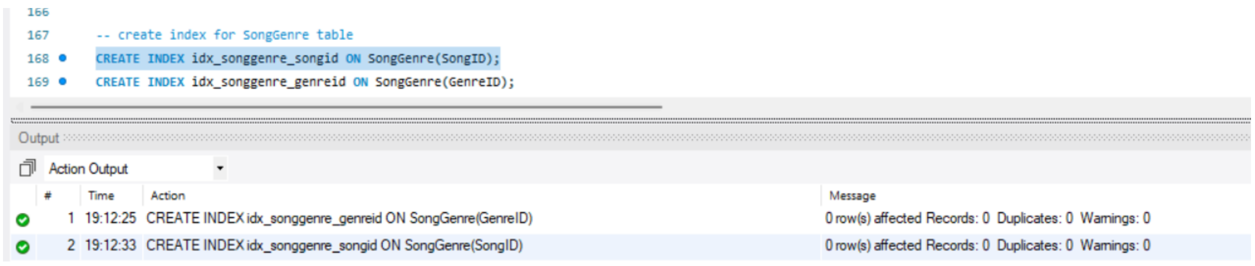
```
-- create index for AlbumGenre table
CREATE INDEX idx_albumgenre_albumid ON AlbumGenre(AlbumID);
CREATE INDEX idx_albumgenre_genreid ON AlbumGenre(GenreID);
```

```
Output
Action Output
#  Time      Action                                                      Message
1  19:11:16  CREATE INDEX idx_albumgenre_albumid ON AlbumGenre(AlbumID)    0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
2  19:11:27  CREATE INDEX idx_albumgenre_genreid ON AlbumGenre(GenreID)    0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
```

-- create index for SongGenre table

CREATE INDEX idx_songgenre_songid ON SongGenre(SongID);

CREATE INDEX idx_songgenre_genreid ON SongGenre(GenreID);



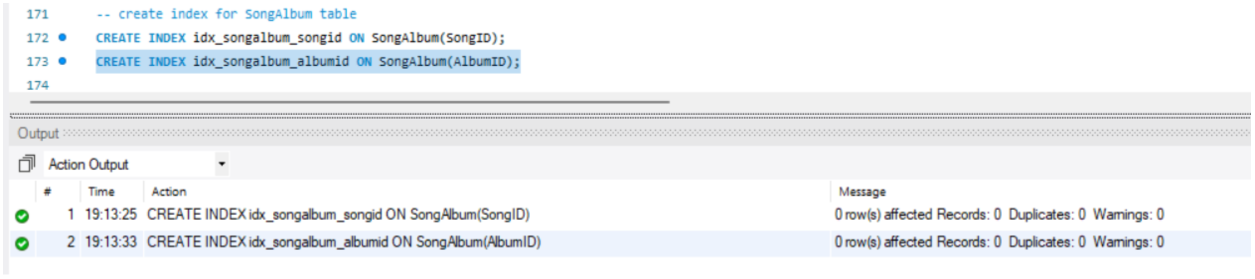
```
-- create index for SongGenre table
CREATE INDEX idx_songgenre_songid ON SongGenre(SongID);
CREATE INDEX idx_songgenre_genreid ON SongGenre(GenreID);
```

```
Output
Action Output
#  Time      Action                                                    Message
1  19:12:25  CREATE INDEX idx_songgenre_genreid ON SongGenre(GenreID)  0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
2  19:12:33  CREATE INDEX idx_songgenre_songid ON SongGenre(SongID)    0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
```

-- create index for SongAlbum table

CREATE INDEX idx_songalbum_songid ON SongAlbum(SongID);

CREATE INDEX idx_songalbum_albumid ON SongAlbum(AlbumID);



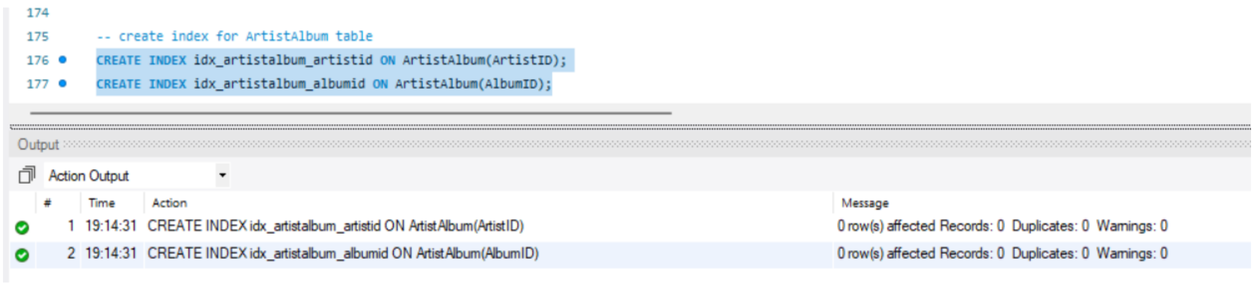
```
-- create index for SongAlbum table
CREATE INDEX idx_songalbum_songid ON SongAlbum(SongID);
CREATE INDEX idx_songalbum_albumid ON SongAlbum(AlbumID);
```

```
Output
Action Output
#  Time      Action                                                    Message
1  19:13:25  CREATE INDEX idx_songalbum_songid ON SongAlbum(SongID)    0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
2  19:13:33  CREATE INDEX idx_songalbum_albumid ON SongAlbum(AlbumID)  0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
```

-- create index for ArtistAlbum table

CREATE INDEX idx_artistalbum_artistid ON ArtistAlbum(ArtistID);

CREATE INDEX idx_artistalbum_albumid ON ArtistAlbum(AlbumID);



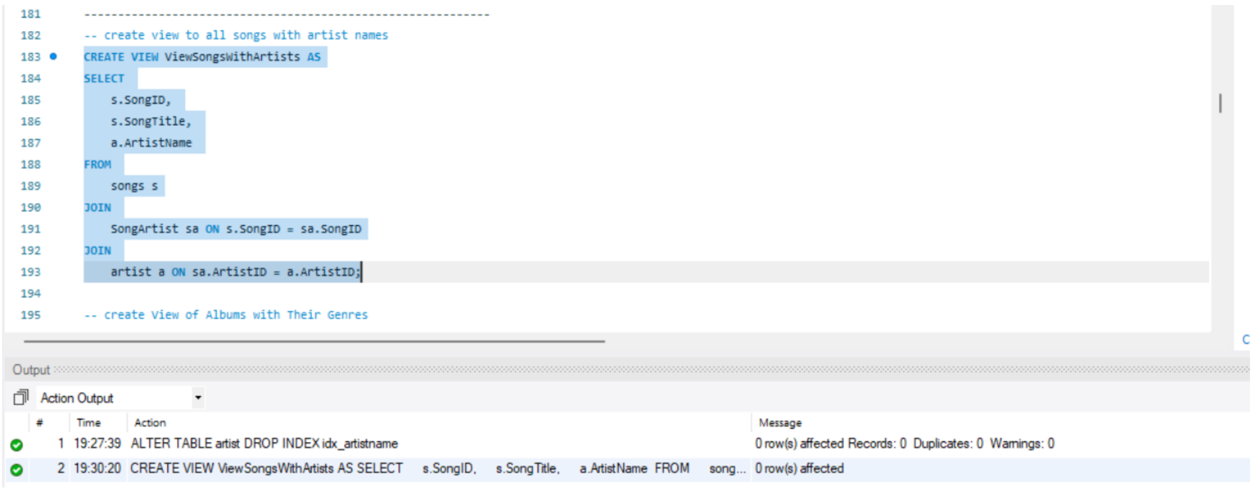
```
-- create index for ArtistAlbum table
CREATE INDEX idx_artistalbum_artistid ON ArtistAlbum(ArtistID);
CREATE INDEX idx_artistalbum_albumid ON ArtistAlbum(AlbumID);
```

```
Output
Action Output
#  Time      Action                                                        Message
1  19:14:31  CREATE INDEX idx_artistalbum_artistid ON ArtistAlbum(ArtistID)  0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
2  19:14:31  CREATE INDEX idx_artistalbum_albumid ON ArtistAlbum(AlbumID)    0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
```

Create View:

1)-- create view to all songs with artist names

```
CREATE VIEW ViewSongsWithArtists AS
SELECT
    s.SongID,
    s.SongTitle,
    a.ArtistName
FROM
    songs s
JOIN
    SongArtist sa ON s.SongID = sa.SongID
JOIN
    artist a ON sa.ArtistID = a.ArtistID;
```

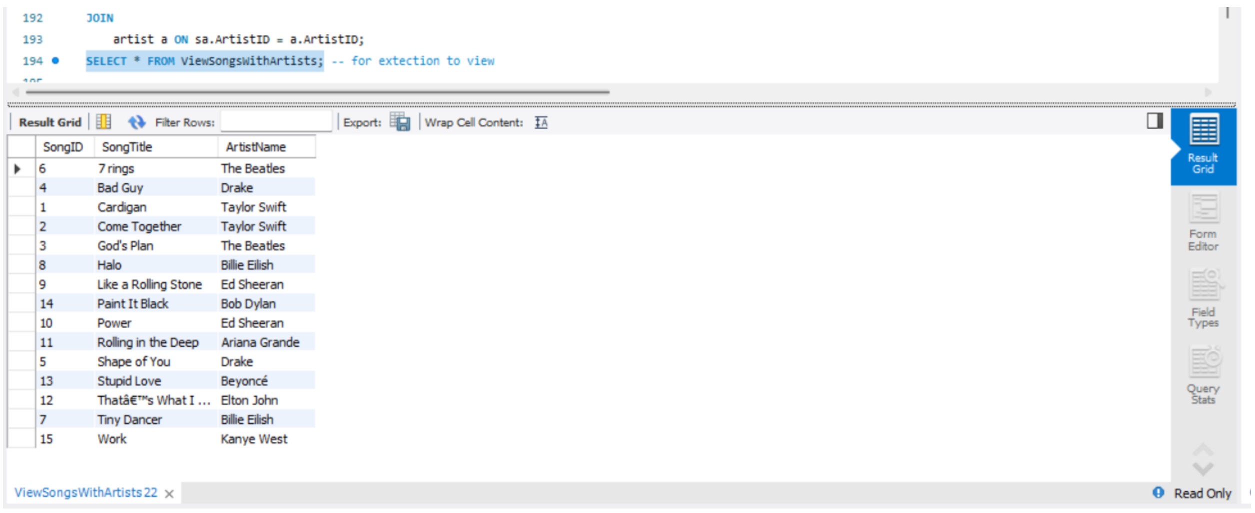


```
------------------------------------------------------------
-- create view to all songs with artist names
CREATE VIEW ViewSongsWithArtists AS
SELECT
    s.SongID,
    s.SongTitle,
    a.ArtistName
FROM
    songs s
JOIN
    SongArtist sa ON s.SongID = sa.SongID
JOIN
    artist a ON sa.ArtistID = a.ArtistID;

-- create View of Albums with Their Genres
```

Output

Action Output

#	Time	Action	Message
1	19:27:39	ALTER TABLE artist DROP INDEX idx_artistname	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
2	19:30:20	CREATE VIEW ViewSongsWithArtists AS SELECT s.SongID, s.SongTitle, a.ArtistName FROM song...	0 row(s) affected

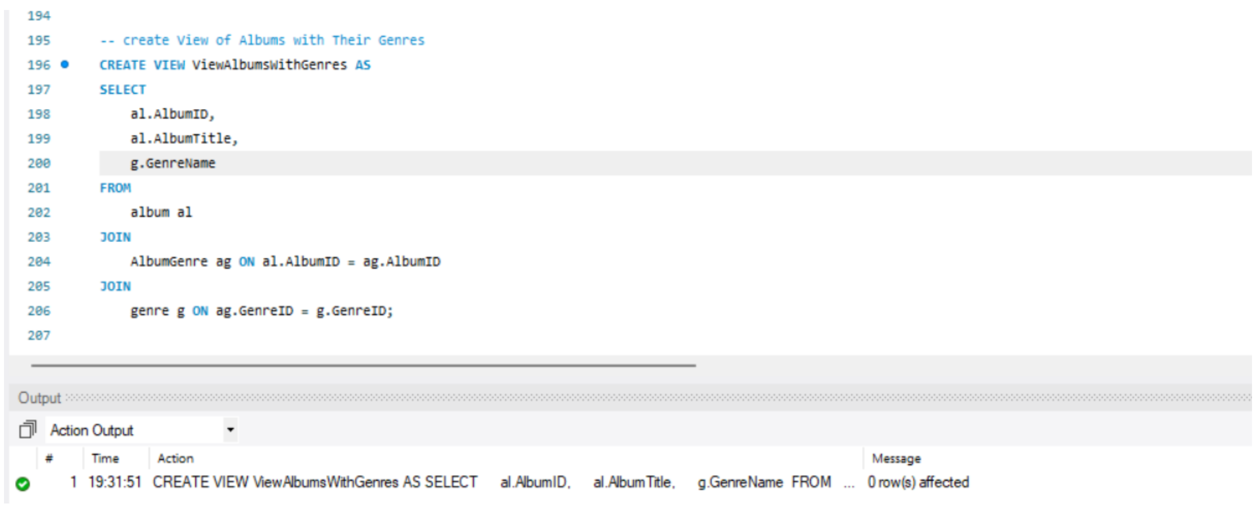


```
    JOIN
        artist a ON sa.ArtistID = a.ArtistID;
    SELECT * FROM ViewSongsWithArtists; -- for extection to view
```

SongID	SongTitle	ArtistName
6	7 rings	The Beatles
4	Bad Guy	Drake
1	Cardigan	Taylor Swift
2	Come Together	Taylor Swift
3	God's Plan	The Beatles
8	Halo	Billie Eilish
9	Like a Rolling Stone	Ed Sheeran
14	Paint It Black	Bob Dylan
10	Power	Ed Sheeran
11	Rolling in the Deep	Ariana Grande
5	Shape of You	Drake
13	Stupid Love	Beyoncé
12	Thatâ€™s What I ...	Elton John
7	Tiny Dancer	Billie Eilish
15	Work	Kanye West

2) -- create View of Albums with Their Genres

```
CREATE VIEW ViewAlbumsWithGenres AS
SELECT
    al.AlbumID,
    al.AlbumTitle,
    g.GenreName
FROM
    album al
JOIN
    AlbumGenre ag ON al.AlbumID = ag.AlbumID
JOIN
    genre g ON ag.GenreID = g.GenreID;
```

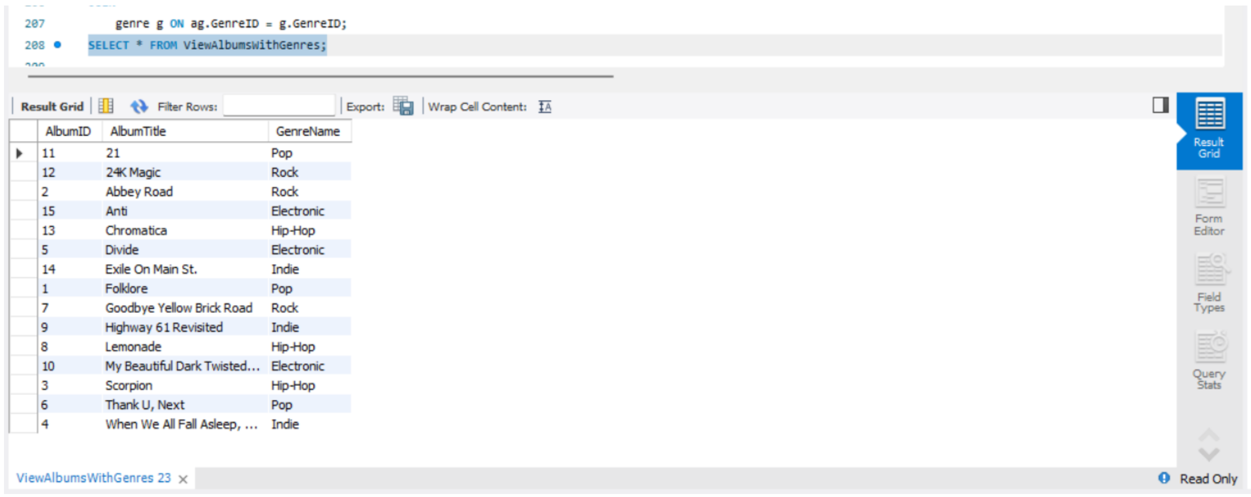


```
-- create View of Albums with Their Genres
CREATE VIEW ViewAlbumsWithGenres AS
SELECT
    al.AlbumID,
    al.AlbumTitle,
    g.GenreName
FROM
    album al
JOIN
    AlbumGenre ag ON al.AlbumID = ag.AlbumID
JOIN
    genre g ON ag.GenreID = g.GenreID;
```

Output

Action Output

	#	Time	Action	Message
✓	1	19:31:51	CREATE VIEW ViewAlbumsWithGenres AS SELECT al.AlbumID, al.AlbumTitle, g.GenreName FROM ...	0 row(s) affected



```
    genre g ON ag.GenreID = g.GenreID;
SELECT * FROM ViewAlbumsWithGenres;
```

AlbumID	AlbumTitle	GenreName
11	21	Pop
12	24K Magic	Rock
2	Abbey Road	Rock
15	Anti	Electronic
13	Chromatica	Hip-Hop
5	Divide	Electronic
14	Exile On Main St.	Indie
1	Folklore	Pop
7	Goodbye Yellow Brick Road	Rock
9	Highway 61 Revisited	Indie
8	Lemonade	Hip-Hop
10	My Beautiful Dark Twisted...	Electronic
3	Scorpion	Hip-Hop
6	Thank U, Next	Pop
4	When We All Fall Asleep, ...	Indie

ViewAlbumsWithGenres 23 | Read Only

3) -- create View of User Subscriptions

CREATE VIEW ViewUserSubscriptions AS

SELECT

u.UserID,

u.Username,

u.Email,

s.SubscriptionType,

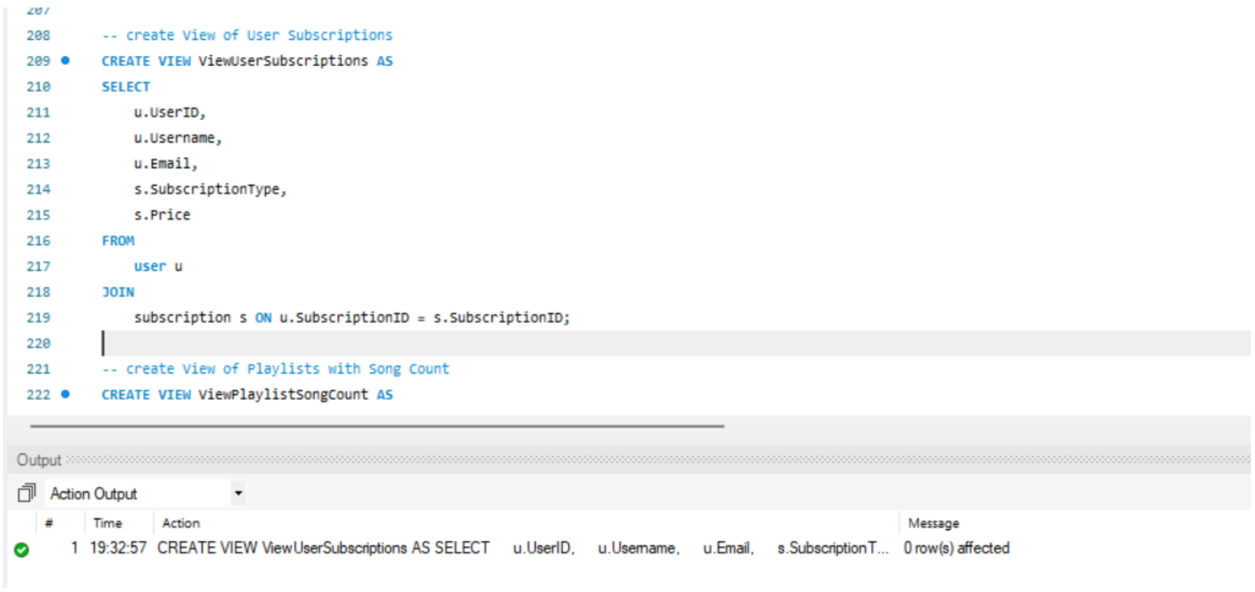
s.Price

FROM

user u

JOIN

subscription s ON u.SubscriptionID = s.SubscriptionID;

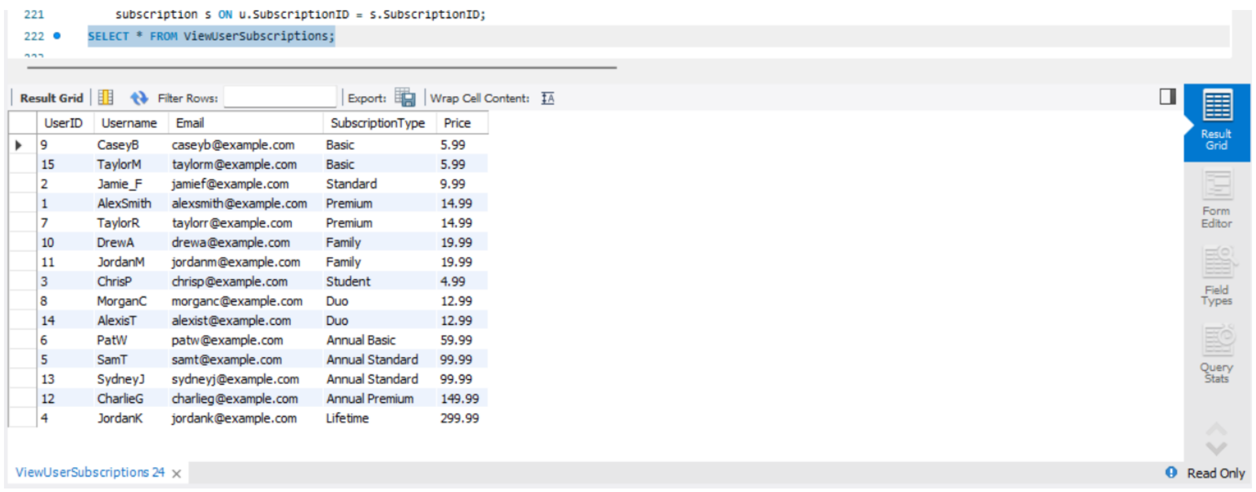


```
-- create View of User Subscriptions
CREATE VIEW ViewUserSubscriptions AS
SELECT
    u.UserID,
    u.Username,
    u.Email,
    s.SubscriptionType,
    s.Price
FROM
    user u
JOIN
    subscription s ON u.SubscriptionID = s.SubscriptionID;

-- create View of Playlists with Song Count
CREATE VIEW ViewPlaylistSongCount AS
```

Output

Action Output

```
#  Time      Action                                                                                        Message
1  19:32:57  CREATE VIEW ViewUserSubscriptions AS SELECT    u.UserID,    u.Username,    u.Email,    s.SubscriptionT...  0 row(s) affected
```

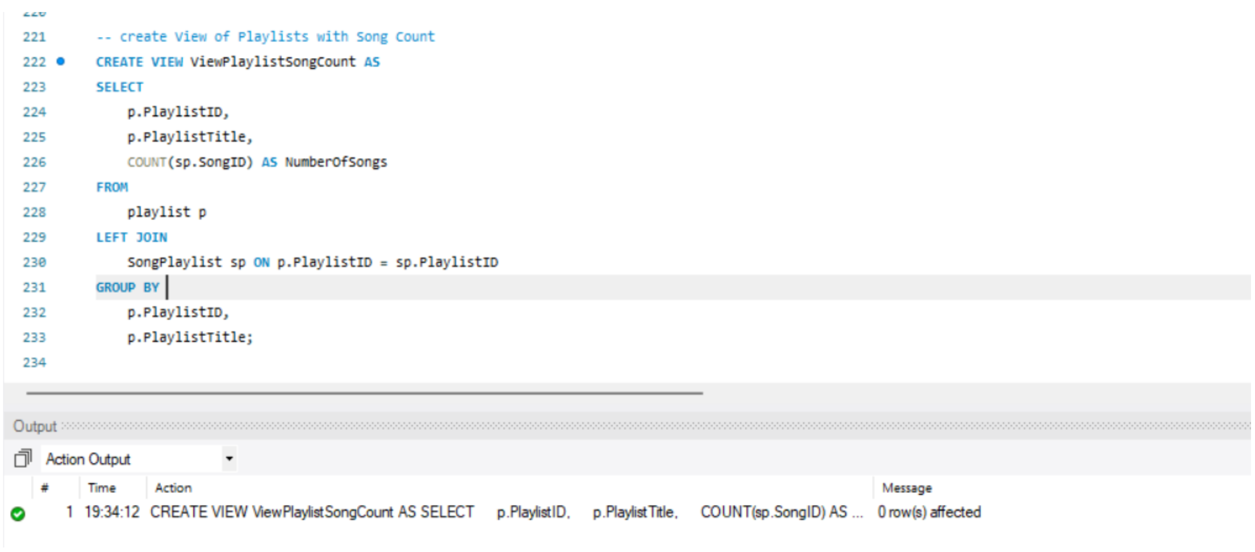


```
    subscription s ON u.SubscriptionID = s.SubscriptionID;
SELECT * FROM ViewUserSubscriptions;
```

UserID	Username	Email	SubscriptionType	Price
9	CaseyB	caseyb@example.com	Basic	5.99
15	TaylorM	taylorm@example.com	Basic	5.99
2	Jamie_F	jamief@example.com	Standard	9.99
1	AlexSmith	alexsmith@example.com	Premium	14.99
7	TaylorR	taylorr@example.com	Premium	14.99
10	DrewA	drewa@example.com	Family	19.99
11	JordanM	jordanm@example.com	Family	19.99
3	ChrisP	chrisp@example.com	Student	4.99
8	MorganC	morganc@example.com	Duo	12.99
14	AlexisT	alexist@example.com	Duo	12.99
6	PatW	patw@example.com	Annual Basic	59.99
5	SamT	samt@example.com	Annual Standard	99.99
13	SydneyJ	sydneyj@example.com	Annual Standard	99.99
12	CharlieG	charlieg@example.com	Annual Premium	149.99
4	JordanK	jordank@example.com	Lifetime	299.99

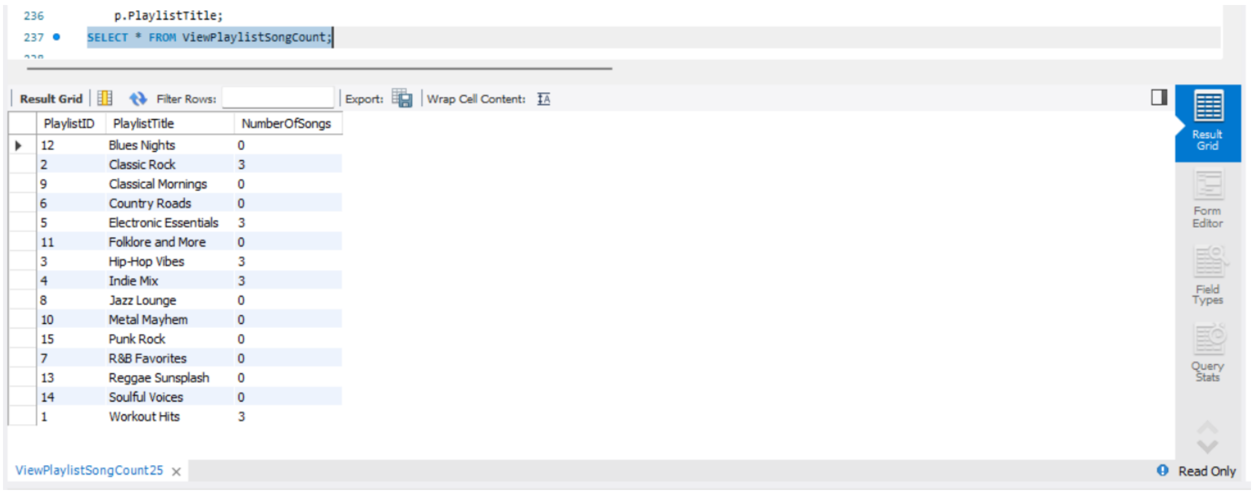
4) -- create View of Playlists with Song Count

```
CREATE VIEW ViewPlaylistSongCount AS
SELECT
    p.PlaylistID,
    p.PlaylistTitle,
    COUNT(sp.SongID) AS NumberOfSongs
FROM
    playlist p
LEFT JOIN
    SongPlaylist sp ON p.PlaylistID = sp.PlaylistID
GROUP BY
    p.PlaylistID,
    p.PlaylistTitle;
```



```
-- create View of Playlists with Song Count
CREATE VIEW ViewPlaylistSongCount AS
SELECT
    p.PlaylistID,
    p.PlaylistTitle,
    COUNT(sp.SongID) AS NumberOfSongs
FROM
    playlist p
LEFT JOIN
    SongPlaylist sp ON p.PlaylistID = sp.PlaylistID
GROUP BY
    p.PlaylistID,
    p.PlaylistTitle;
```

```
Output
Action Output
#   Time      Action                                                                                  Message
1   19:34:12  CREATE VIEW ViewPlaylistSongCount AS SELECT    p.PlaylistID,    p.PlaylistTitle,    COUNT(sp.SongID) AS ...   0 row(s) affected
```



```
    p.PlaylistTitle;
SELECT * FROM ViewPlaylistSongCount;
```

PlaylistID	PlaylistTitle	NumberOfSongs
12	Blues Nights	0
2	Classic Rock	3
9	Classical Mornings	0
6	Country Roads	0
5	Electronic Essentials	3
11	Folklore and More	0
3	Hip-Hop Vibes	3
4	Indie Mix	3
8	Jazz Lounge	0
10	Metal Mayhem	0
15	Punk Rock	0
7	R&B Favorites	0
13	Reggae Sunsplash	0
14	Soulful Voices	0
1	Workout Hits	3

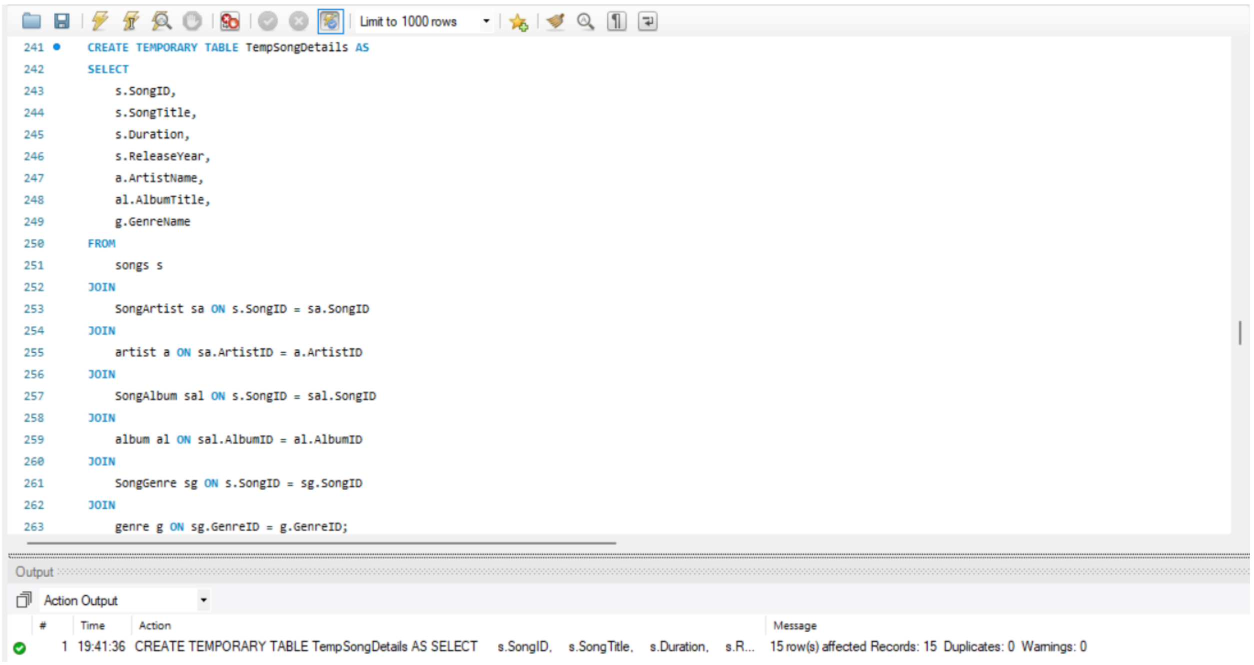
<u>Temporary Table:</u>

1)-- create a temporary Table for Detailed Song Information

```
CREATE TEMPORARY TABLE TempSongDetails AS
SELECT
    s.SongID,
    s.SongTitle,
    s.Duration,
    s.ReleaseYear,
    a.ArtistName,
    al.AlbumTitle,
    g.GenreName
FROM
    songs s
JOIN
    SongArtist sa ON s.SongID = sa.SongID
JOIN
    artist a ON sa.ArtistID = a.ArtistID
JOIN
    SongAlbum sal ON s.SongID = sal.SongID
JOIN
    album al ON sal.AlbumID = al.AlbumID
JOIN
    SongGenre sg ON s.SongID = sg.SongID
```

JOIN

genre g ON sg.GenreID = g.GenreID;



```
CREATE TEMPORARY TABLE TempSongDetails AS
SELECT
    s.SongID,
    s.SongTitle,
    s.Duration,
    s.ReleaseYear,
    a.ArtistName,
    al.AlbumTitle,
    g.GenreName
FROM
    songs s
JOIN
    SongArtist sa ON s.SongID = sa.SongID
JOIN
    artist a ON sa.ArtistID = a.ArtistID
JOIN
    SongAlbum sal ON s.SongID = sal.SongID
JOIN
    album al ON sal.AlbumID = al.AlbumID
JOIN
    SongGenre sg ON s.SongID = sg.SongID
JOIN
    genre g ON sg.GenreID = g.GenreID;
```

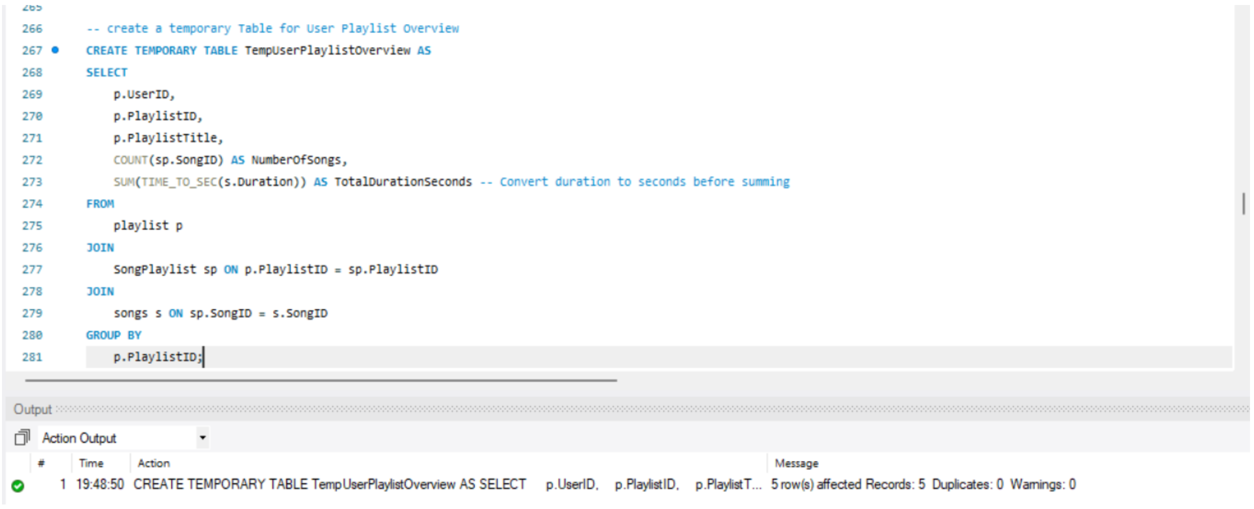
Output

Action Output

```
#  Time      Action                                                                                  Message
1  19:41:36  CREATE TEMPORARY TABLE TempSongDetails AS SELECT   s.SongID,  s.SongTitle,  s.Duration,  s.R...   15 row(s) affected Records: 15  Duplicates: 0  Warnings: 0
```

2)-- create a temporary Table for User Playlist Overview

```
CREATE TEMPORARY TABLE TempUserPlaylistOverview AS
SELECT
    p.UserID,
    p.PlaylistID,
    p.PlaylistTitle,
    COUNT(sp.SongID) AS NumberOfSongs,
    SUM(s.Duration) AS TotalDuration
FROM
    playlist p
JOIN
    SongPlaylist sp ON p.PlaylistID = sp.PlaylistID
JOIN
    songs s ON sp.SongID = s.SongID
GROUP BY
    p.PlaylistID;
```

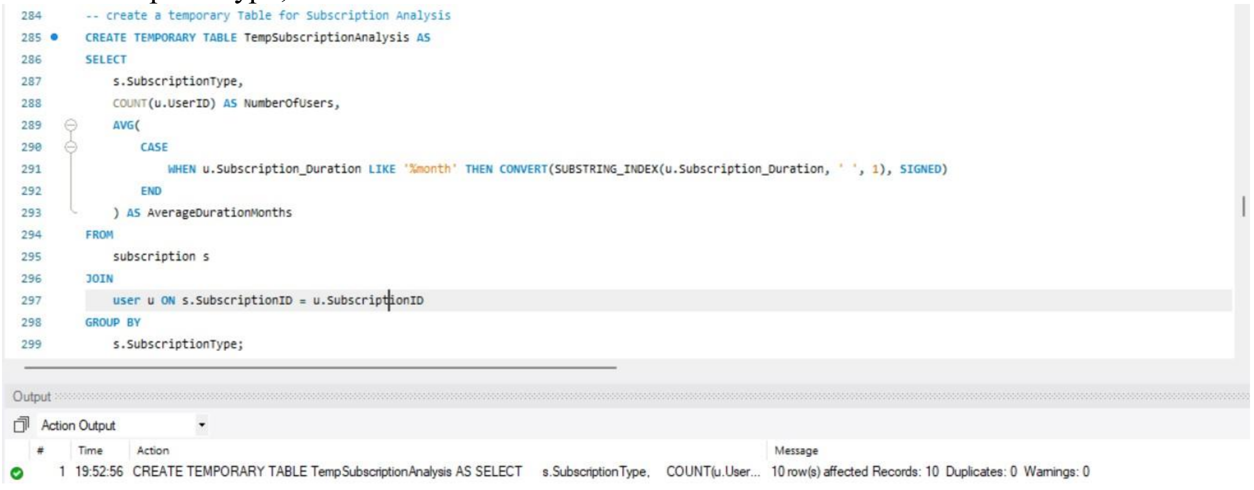


```
-- create a temporary Table for User Playlist Overview
CREATE TEMPORARY TABLE TempUserPlaylistOverview AS
SELECT
    p.UserID,
    p.PlaylistID,
    p.PlaylistTitle,
    COUNT(sp.SongID) AS NumberOfSongs,
    SUM(TIME_TO_SEC(s.Duration)) AS TotalDurationSeconds -- Convert duration to seconds before summing
FROM
    playlist p
JOIN
    SongPlaylist sp ON p.PlaylistID = sp.PlaylistID
JOIN
    songs s ON sp.SongID = s.SongID
GROUP BY
    p.PlaylistID;
```

```
#   Time      Action                                                                              Message
1   19:48:50  CREATE TEMPORARY TABLE TempUserPlaylistOverview AS SELECT    p.UserID,    p.PlaylistID,    p.PlaylistT...  5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0
```

3)-- create a temporary Table for Subscription Analysis

```
CREATE TEMPORARY TABLE TempSubscriptionAnalysis AS

SELECT

    s.SubscriptionType,

    COUNT(u.UserID) AS NumberOfUsers,

    AVG(

        CASE

            WHEN u.Subscription_Duration LIKE '%month' THEN
CONVERT(SUBSTRING_INDEX(u.Subscription_Duration, ' ', 1), SIGNED)

        END

    ) AS AverageDurationMonths

FROM

    subscription s

JOIN

    user u ON s.SubscriptionID = u.SubscriptionID

GROUP BY

    s.SubscriptionType;
```



```
-- create a temporary Table for Subscription Analysis
CREATE TEMPORARY TABLE TempSubscriptionAnalysis AS
SELECT
    s.SubscriptionType,
    COUNT(u.UserID) AS NumberOfUsers,
    AVG(
        CASE
            WHEN u.Subscription_Duration LIKE '%month' THEN CONVERT(SUBSTRING_INDEX(u.Subscription_Duration, ' ', 1), SIGNED)
        END
    ) AS AverageDurationMonths
FROM
    subscription s
JOIN
    user u ON s.SubscriptionID = u.SubscriptionID
GROUP BY
    s.SubscriptionType;
```

```
#   Time      Action                                                                              Message
1   19:52:56  CREATE TEMPORARY TABLE TempSubscriptionAnalysis AS SELECT   s.SubscriptionType,   COUNT(u.User...  10 row(s) affected Records: 10  Duplicates: 0  Warnings: 0
```

Trigger:

1)-- Trigger to Log When a New Artist is Added

DELIMITER $$

CREATE TRIGGER AfterArtistInsert

AFTER INSERT ON artist

FOR EACH ROW

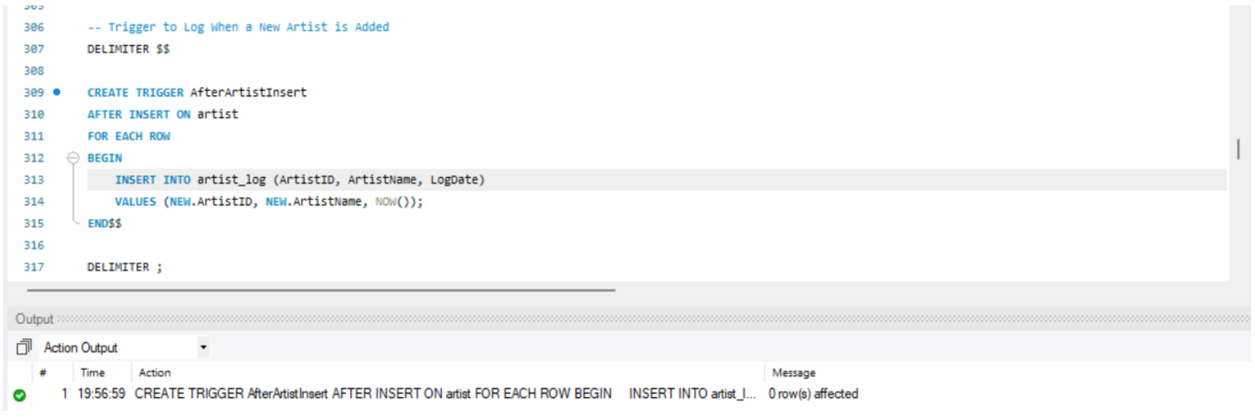
BEGIN

    INSERT INTO artist_log (ArtistID, ArtistName, LogDate)

    VALUES (NEW.ArtistID, NEW.ArtistName, NOW());

END$$

DELIMITER ;

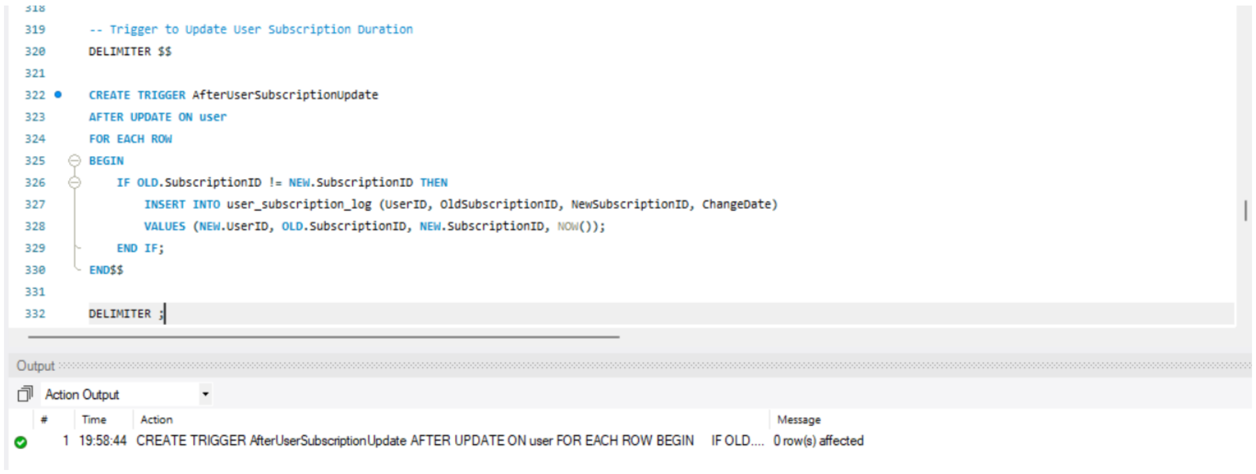


```
-- Trigger to Log When a New Artist is Added
DELIMITER $$

CREATE TRIGGER AfterArtistInsert
AFTER INSERT ON artist
FOR EACH ROW
BEGIN
    INSERT INTO artist_log (ArtistID, ArtistName, LogDate)
    VALUES (NEW.ArtistID, NEW.ArtistName, NOW());
END$$

DELIMITER ;
```

```
#   Time      Action                                                                          Message
1   19:56:59  CREATE TRIGGER AfterArtistInsert AFTER INSERT ON artist FOR EACH ROW BEGIN   INSERT INTO artist_l...   0 row(s) affected
```

2)-- Trigger to Update User Subscription Duration

```
DELIMITER $$

CREATE TRIGGER AfterUserSubscriptionUpdate

AFTER UPDATE ON user

FOR EACH ROW

BEGIN

    IF OLD.SubscriptionID != NEW.SubscriptionID THEN

        INSERT INTO user_subscription_log (UserID, OldSubscriptionID, NewSubscriptionID, ChangeDate)

        VALUES (NEW.UserID, OLD.SubscriptionID, NEW.SubscriptionID, NOW());

    END IF;

END$$

DELIMITER ;
```



```
-- Trigger to Update User Subscription Duration
DELIMITER $$

CREATE TRIGGER AfterUserSubscriptionUpdate
AFTER UPDATE ON user
FOR EACH ROW
BEGIN
    IF OLD.SubscriptionID != NEW.SubscriptionID THEN
        INSERT INTO user_subscription_log (UserID, OldSubscriptionID, NewSubscriptionID, ChangeDate)
        VALUES (NEW.UserID, OLD.SubscriptionID, NEW.SubscriptionID, NOW());
    END IF;
END$$

DELIMITER ;
```

Output

Action Output

```
#  Time      Action                                                                              Message
1  19:58:44  CREATE TRIGGER AfterUserSubscriptionUpdate AFTER UPDATE ON user FOR EACH ROW BEGIN    IF OLD....  0 row(s) affected
```

3)-- Trigger for Deleting Songs from Playlists When a Song is Deleted

DELIMITER $$

CREATE TRIGGER BeforeSongDelete

BEFORE DELETE ON songs

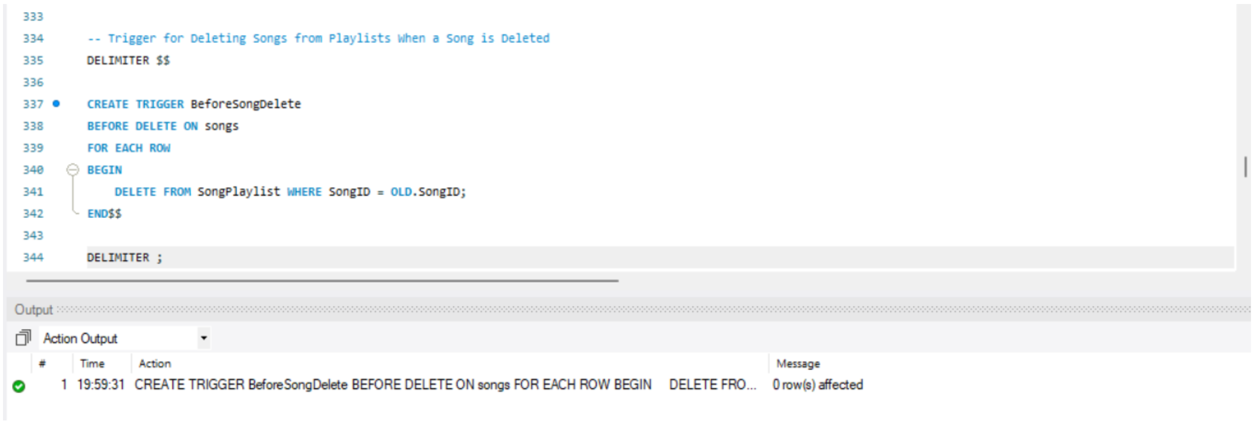
FOR EACH ROW

BEGIN

    DELETE FROM SongPlaylist WHERE SongID = OLD.SongID;

END$$

DELIMITER ;



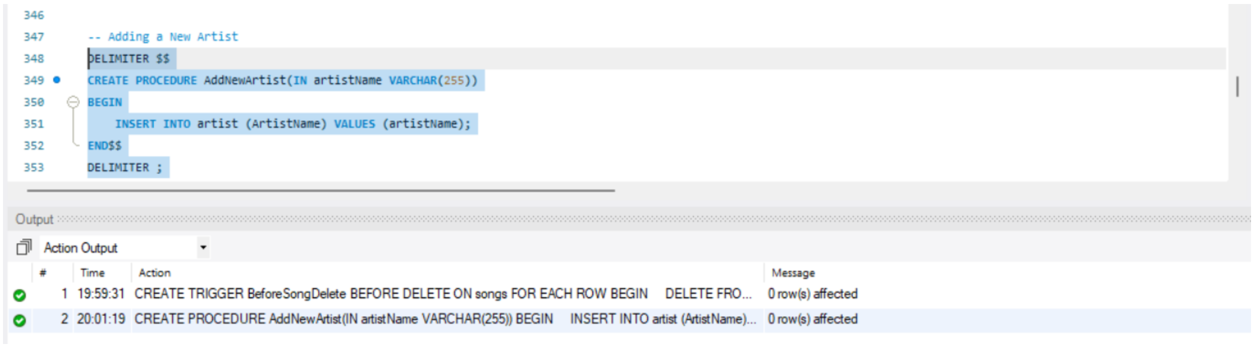
```
-- Trigger for Deleting Songs from Playlists When a Song is Deleted
DELIMITER $$

CREATE TRIGGER BeforeSongDelete
BEFORE DELETE ON songs
FOR EACH ROW
BEGIN
    DELETE FROM SongPlaylist WHERE SongID = OLD.SongID;
END$$

DELIMITER ;
```

```
Output
Action Output
#   Time      Action                                                                         Message
1   19:59:31  CREATE TRIGGER BeforeSongDelete BEFORE DELETE ON songs FOR EACH ROW BEGIN   DELETE FRO...   0 row(s) affected
```

STORED PROCEDURE:

1)-- Adding a New Artist

```
DELIMITER $$

CREATE PROCEDURE AddNewArtist(IN artistName VARCHAR(255))

BEGIN

    INSERT INTO artist (ArtistName) VALUES (artistName);

END$$

DELIMITER ;
```



```

-- Adding a New Artist
DELIMITER $$
CREATE PROCEDURE AddNewArtist(IN artistName VARCHAR(255))
BEGIN
    INSERT INTO artist (ArtistName) VALUES (artistName);
END$$
DELIMITER ;
```

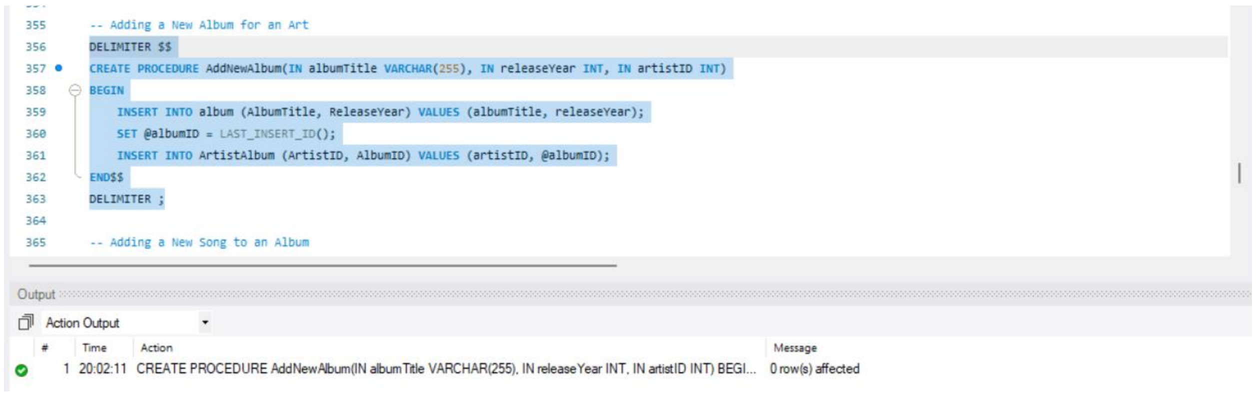
Output

Action Output

```
 #  Time      Action                                                                                   Message
 1  19:59:31  CREATE TRIGGER BeforeSongDelete BEFORE DELETE ON songs FOR EACH ROW BEGIN    DELETE FRO...  0 row(s) affected
 2  20:01:19  CREATE PROCEDURE AddNewArtist(IN artistName VARCHAR(255)) BEGIN    INSERT INTO artist (ArtistName)...  0 row(s) affected
```

2) -- Adding a New Album for an Art

```
DELIMITER $$

CREATE PROCEDURE AddNewAlbum(IN albumTitle VARCHAR(255), IN releaseYear INT,
IN artistID INT)

BEGIN

    INSERT INTO album (AlbumTitle, ReleaseYear) VALUES (albumTitle, releaseYear);

    SET @albumID = LAST_INSERT_ID();

    INSERT INTO ArtistAlbum (ArtistID, AlbumID) VALUES (artistID, @albumID);

END$$

DELIMITER ;
```

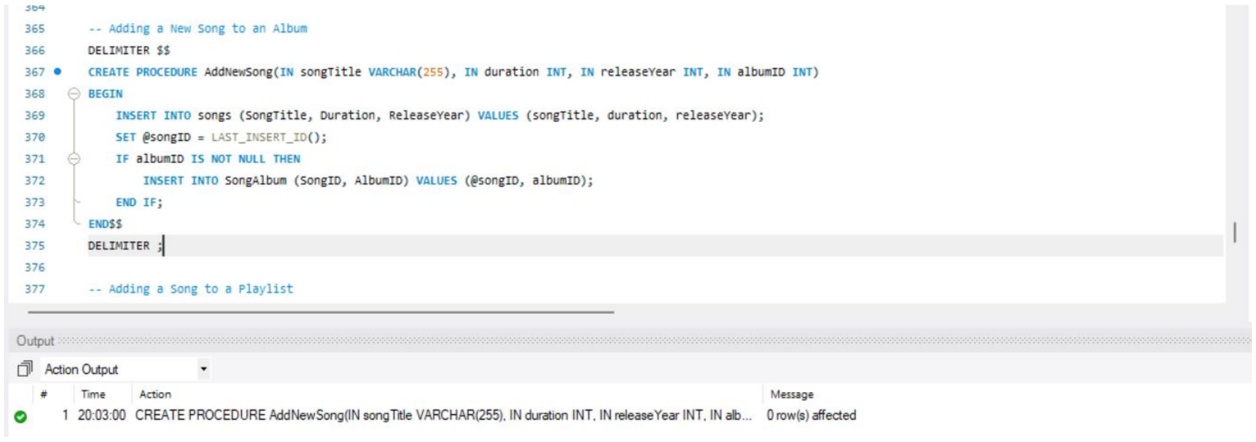


```
-- Adding a New Album for an Art
DELIMITER $$
CREATE PROCEDURE AddNewAlbum(IN albumTitle VARCHAR(255), IN releaseYear INT, IN artistID INT)
BEGIN
    INSERT INTO album (AlbumTitle, ReleaseYear) VALUES (albumTitle, releaseYear);
    SET @albumID = LAST_INSERT_ID();
    INSERT INTO ArtistAlbum (ArtistID, AlbumID) VALUES (artistID, @albumID);
END$$
DELIMITER ;

-- Adding a New Song to an Album
```

```
Output
Action Output
#   Time       Action                                                                                       Message
1   20:02:11   CREATE PROCEDURE AddNewAlbum(IN albumTitle VARCHAR(255), IN releaseYear INT, IN artistID INT) BEGI...   0 row(s) affected
```

3)-- Adding a New Song to an Album

```
DELIMITER $$

CREATE PROCEDURE AddNewSong(IN songTitle VARCHAR(255), IN duration INT, IN
releaseYear INT, IN albumID INT)

BEGIN

    INSERT INTO songs (SongTitle, Duration, ReleaseYear) VALUES (songTitle, duration,
releaseYear);

    SET @songID = LAST_INSERT_ID();

    IF albumID IS NOT NULL THEN

        INSERT INTO SongAlbum (SongID, AlbumID) VALUES (@songID, albumID);

    END IF;

END$$

DELIMITER ;
```



```
-- Adding a New Song to an Album
DELIMITER $$
CREATE PROCEDURE AddNewSong(IN songTitle VARCHAR(255), IN duration INT, IN releaseYear INT, IN albumID INT)
BEGIN
    INSERT INTO songs (SongTitle, Duration, ReleaseYear) VALUES (songTitle, duration, releaseYear);
    SET @songID = LAST_INSERT_ID();
    IF albumID IS NOT NULL THEN
        INSERT INTO SongAlbum (SongID, AlbumID) VALUES (@songID, albumID);
    END IF;
END$$
DELIMITER ;

-- Adding a Song to a Playlist
```

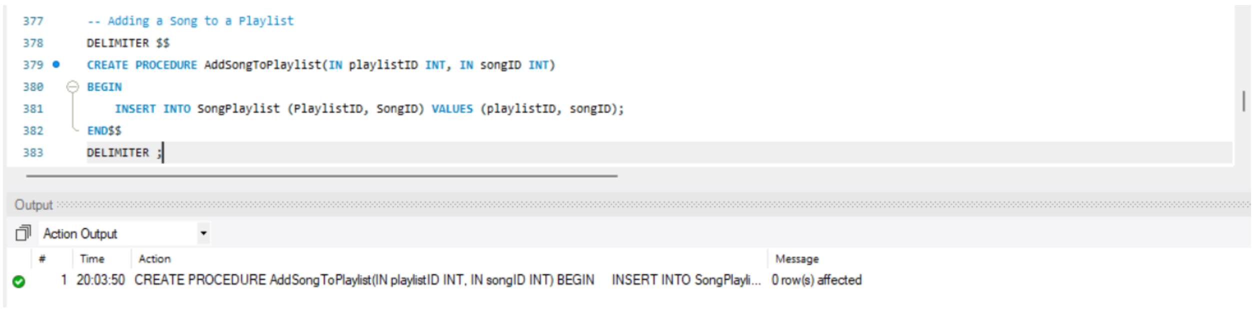
Output

Action Output

```
   #  Time      Action                                                                                         Message
   1  20:03:00  CREATE PROCEDURE AddNewSong(IN songTitle VARCHAR(255), IN duration INT, IN releaseYear INT, IN alb...  0 row(s) affected
```

4)-- Adding a Song to a Playlist

```
DELIMITER $$

CREATE PROCEDURE AddSongToPlaylist(IN playlistID INT, IN songID INT)

BEGIN

    INSERT INTO SongPlaylist (PlaylistID, SongID) VALUES (playlistID, songID);

END$$

DELIMITER ;
```



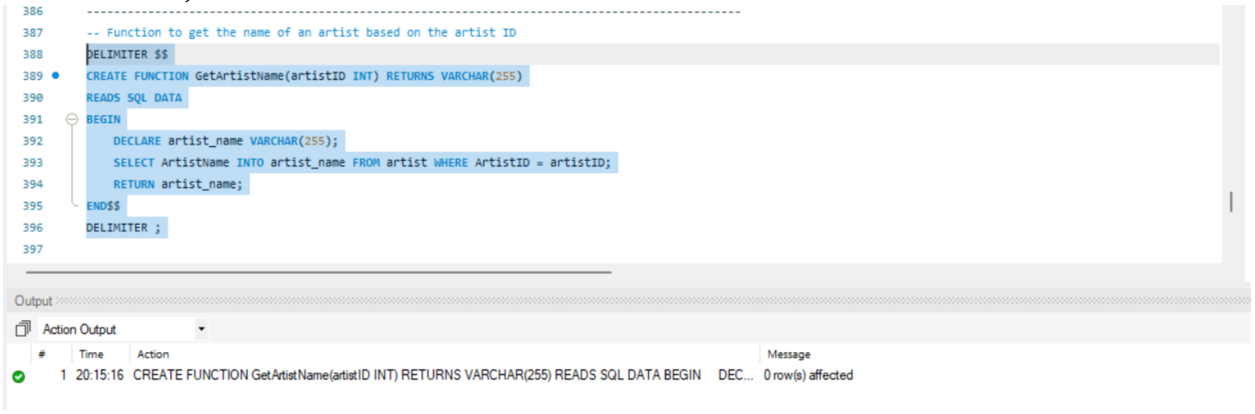
```
-- Adding a Song to a Playlist
DELIMITER $$
CREATE PROCEDURE AddSongToPlaylist(IN playlistID INT, IN songID INT)
BEGIN
    INSERT INTO SongPlaylist (PlaylistID, SongID) VALUES (playlistID, songID);
END$$
DELIMITER ;
```

```
#   Time      Action                                                                                         Message
1   20:03:50  CREATE PROCEDURE AddSongToPlaylist(IN playlistID INT, IN songID INT) BEGIN    INSERT INTO SongPlayli...  0 row(s) affected
```

<u>Function:</u>

1)-- Function to get the name of an artist based on the artist ID

```
DELIMITER $$

CREATE FUNCTION GetArtistName(artistID INT) RETURNS VARCHAR(255)

READS SQL DATA

BEGIN

    DECLARE artist_name VARCHAR(255);

    SELECT ArtistName INTO artist_name FROM artist WHERE ArtistID = artistID;

    RETURN artist_name;

END$$

DELIMITER ;
```

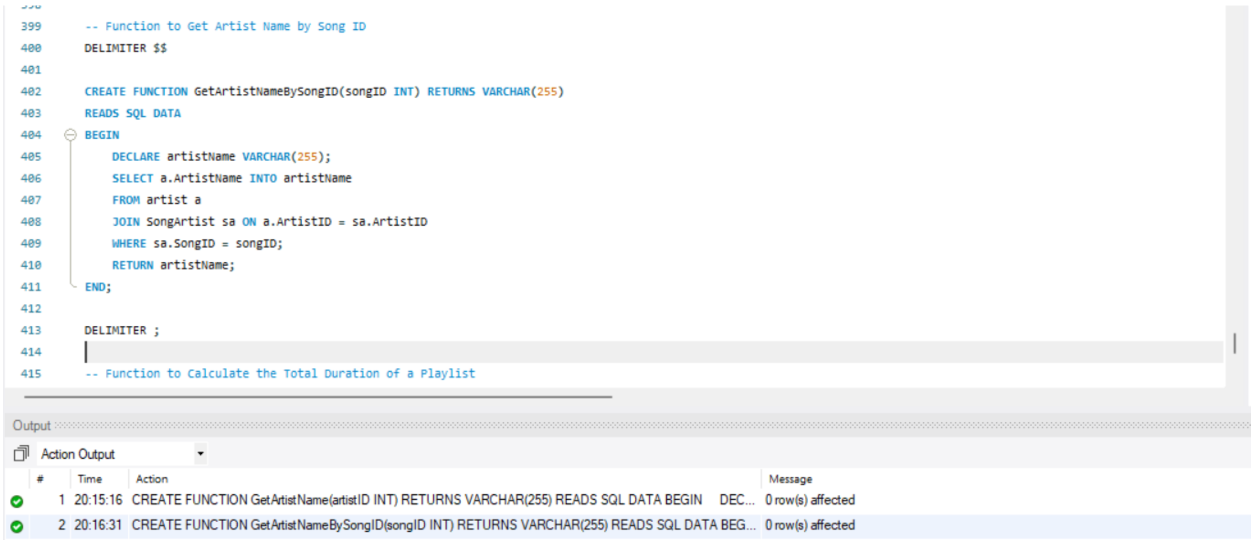


```
------------------------------------------------------------------------------------------------
-- Function to get the name of an artist based on the artist ID
DELIMITER $$
CREATE FUNCTION GetArtistName(artistID INT) RETURNS VARCHAR(255)
READS SQL DATA
BEGIN
    DECLARE artist_name VARCHAR(255);
    SELECT ArtistName INTO artist_name FROM artist WHERE ArtistID = artistID;
    RETURN artist_name;
END$$
DELIMITER ;
```

```
#   Time      Action                                                                                  Message
1   20:15:16  CREATE FUNCTION GetArtistName(artistID INT) RETURNS VARCHAR(255) READS SQL DATA BEGIN   DEC...  0 row(s) affected
```

2)-- Function to Get Artist Name by Song ID

```
DELIMITER $$

CREATE FUNCTION GetArtistNameBySongID(songID INT) RETURNS VARCHAR(255)

READS SQL DATA

BEGIN

    DECLARE artistName VARCHAR(255);

    SELECT a.ArtistName INTO artistName

    FROM artist a

    JOIN SongArtist sa ON a.ArtistID = sa.ArtistID

    WHERE sa.SongID = songID;

    RETURN artistName;

END;

DELIMITER ;
```

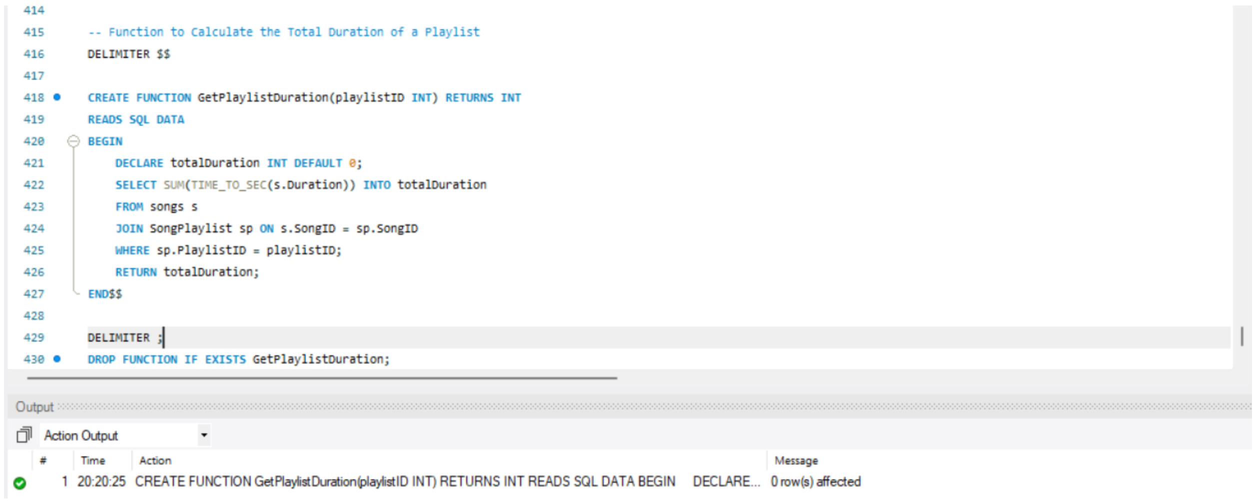


```
-- Function to Get Artist Name by Song ID
DELIMITER $$

CREATE FUNCTION GetArtistNameBySongID(songID INT) RETURNS VARCHAR(255)
READS SQL DATA
BEGIN
    DECLARE artistName VARCHAR(255);
    SELECT a.ArtistName INTO artistName
    FROM artist a
    JOIN SongArtist sa ON a.ArtistID = sa.ArtistID
    WHERE sa.SongID = songID;
    RETURN artistName;
END;

DELIMITER ;

-- Function to Calculate the Total Duration of a Playlist
```

```
Output
Action Output
   #  Time      Action                                                                                     Message
   1  20:15:16  CREATE FUNCTION GetArtistName(artistID INT) RETURNS VARCHAR(255) READS SQL DATA BEGIN    DEC...  0 row(s) affected
   2  20:16:31  CREATE FUNCTION GetArtistNameBySongID(songID INT) RETURNS VARCHAR(255) READS SQL DATA BEG...     0 row(s) affected
```

3)-- Function to Calculate the Total Duration of a Playlist

```
DELIMITER $$

CREATE FUNCTION GetPlaylistDuration(playlistID INT) RETURNS INT

READS SQL DATA

BEGIN

    DECLARE totalDuration INT DEFAULT 0;

    SELECT SUM(TIME_TO_SEC(s.Duration)) INTO totalDuration

    FROM songs s

    JOIN SongPlaylist sp ON s.SongID = sp.SongID

    WHERE sp.PlaylistID = playlistID;

    RETURN totalDuration;

END$$

DELIMITER ;
```



```

-- Function to Calculate the Total Duration of a Playlist
DELIMITER $$

CREATE FUNCTION GetPlaylistDuration(playlistID INT) RETURNS INT
READS SQL DATA
BEGIN
    DECLARE totalDuration INT DEFAULT 0;
    SELECT SUM(TIME_TO_SEC(s.Duration)) INTO totalDuration
    FROM songs s
    JOIN SongPlaylist sp ON s.SongID = sp.SongID
    WHERE sp.PlaylistID = playlistID;
    RETURN totalDuration;
END$$

DELIMITER ;
DROP FUNCTION IF EXISTS GetPlaylistDuration;
```

Output

Action Output

#	Time	Action	Message
1	20:20:25	CREATE FUNCTION GetPlaylistDuration(playlistID INT) RETURNS INT READS SQL DATA BEGIN DECLARE...	0 row(s) affected