A song is a musical composition intended to be performed by the human voice and it is an art of Music. Hence My Project is titled Song List. My database name is MyProject which consists of three tables that they are Song, Singers & Awards. Each table contains different parameters related to the Song.

The scope of the project to

* Create a database on Microsoft SQL Server,
* Execute the API Program on VisualStudio Code,
* Testing the API Program

Here API program uses **CRUD** operation which means

* C – **C**reate
* R - **R**ead
* U - **U**pdate
* D - **D**elete

These Operations are done in the Song List Project.

**GITHUB** is also used to upload all those files done in the project.

**DATABASE**

**DataBase Name** : MyProject

**Tables** : 3

**Entities**  : 5

**Table 1 : Song**

Attributes

* SongId
* Name
* Genre
* Instruments
* Directors

**Table 2 : Singers**

Attributes

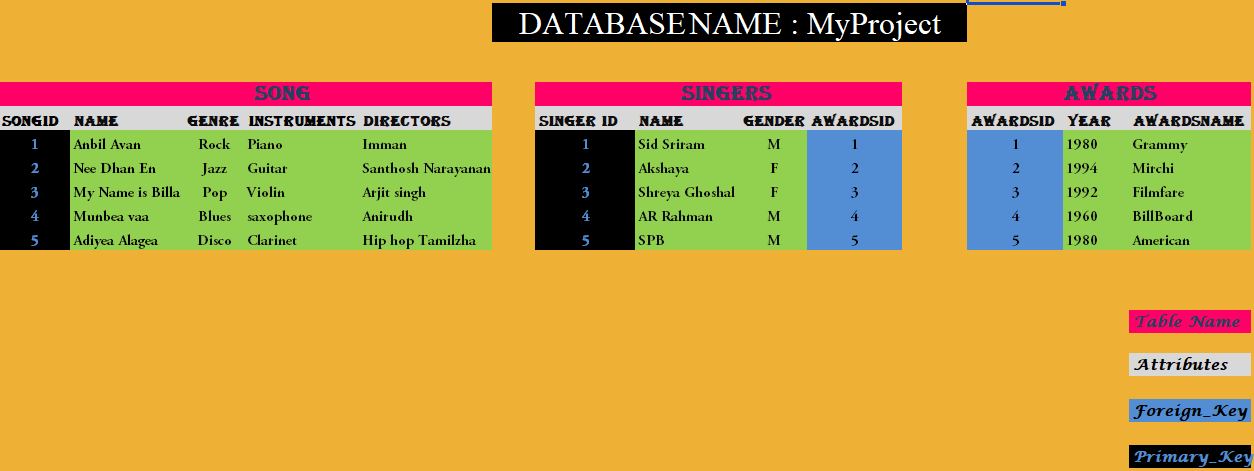
* SingerId
* SingerName
* Gender
* AwardId

**Table 3 : Awards**

Attributes

* AwardId
* Year
* AwardsName
* **Primary Keys** : SongId & SingerId
* **Foreign Key** : AwardsId

**TABLE DESIGN**

****

**SQL QUERIES**

------DDL COMMANDS-----

---CREATE DB----

CREATE DATABASE MyProject;

USE MyProject;

----CREATE Table----

CREATE TABLE Singers

(

SingerId int not null PRIMARY KEY ,

SingerName varchar(20),

Gender varchar(1),

AwardsId int not null ,

CONSTRAINT FK\_AwardSinger FOREIGN KEY (AwardsId)

REFERENCES Awards(AwardsId)

);

-----INSERT Values into the Respective COLUMNS-----

INSERT INTO Singers (SingerName,Gender,AwardsId)

VALUES ('SidSriram','M',1);

INSERT INTO Singers VALUES ('Akshaya','F',2);

INSERT INTO Singers VALUES ('Shreya Ghoshal','F',3);

INSERT INTO Singers VALUES ('AR Rahman','M',4);

INSERT INTO Singers VALUES ('SPB','M',3);

-----SELECT all columns in the TABLE------

SELECT \* FROM Singers

----Delete the Table----

DROP TABLE Singers

----ALTER Table----

ALTER TABLE Singers ADD SingerAward varchar(20);

----To Drop ALTER Table----

ALTER TABLE Singers DROP COLUMN SingerAward;

--- INSERT another Entity to the table---

INSERT INTO Singers VALUES ('GV Prakash','M',4);

TRUNCATE TABLE singers;

---Rename the Table from Singers to Singer ---

sp\_rename 'singer','singers';

---------------------------------------------

CREATE TABLE Singers

(

----- Setting NOT NULL Constraint -----

SingerId int not null,

SingerName varchar(20),

Gender varchar(1),

AwardsId int not null ,

----Setting PRIMARY KEY----

PRIMARY KEY(SingerId),

------Setting FOREIGN KEY -------

CONSTRAINT FK\_AwardSinger FOREIGN KEY (AwardsId)

REFERENCES Awards(AwardsId)

);

INSERT INTO Singers (SingerName,Gender,AwardsId)

VALUES ('SidSriram','M',1);

INSERT INTO Singers VALUES ('Akshaya','F',2);

INSERT INTO Singers VALUES ('Shreya Ghoshal','F',3);

INSERT INTO Singers VALUES ('AR Rahman','M',4);

INSERT INTO Singers VALUES ('SPB','M',3);

SELECT \* FROM Singers

-----Award Table----

CREATE TABLE Awards

(

AwardsId int NOT NULL PRIMARY KEY,

Year int,

AwardsName varchar(20)

);

INSERT INTO Awards (Year,AwardsName) VALUES (1980,'Grammy');

INSERT INTO Awards VALUES (1994,'Mirchi');

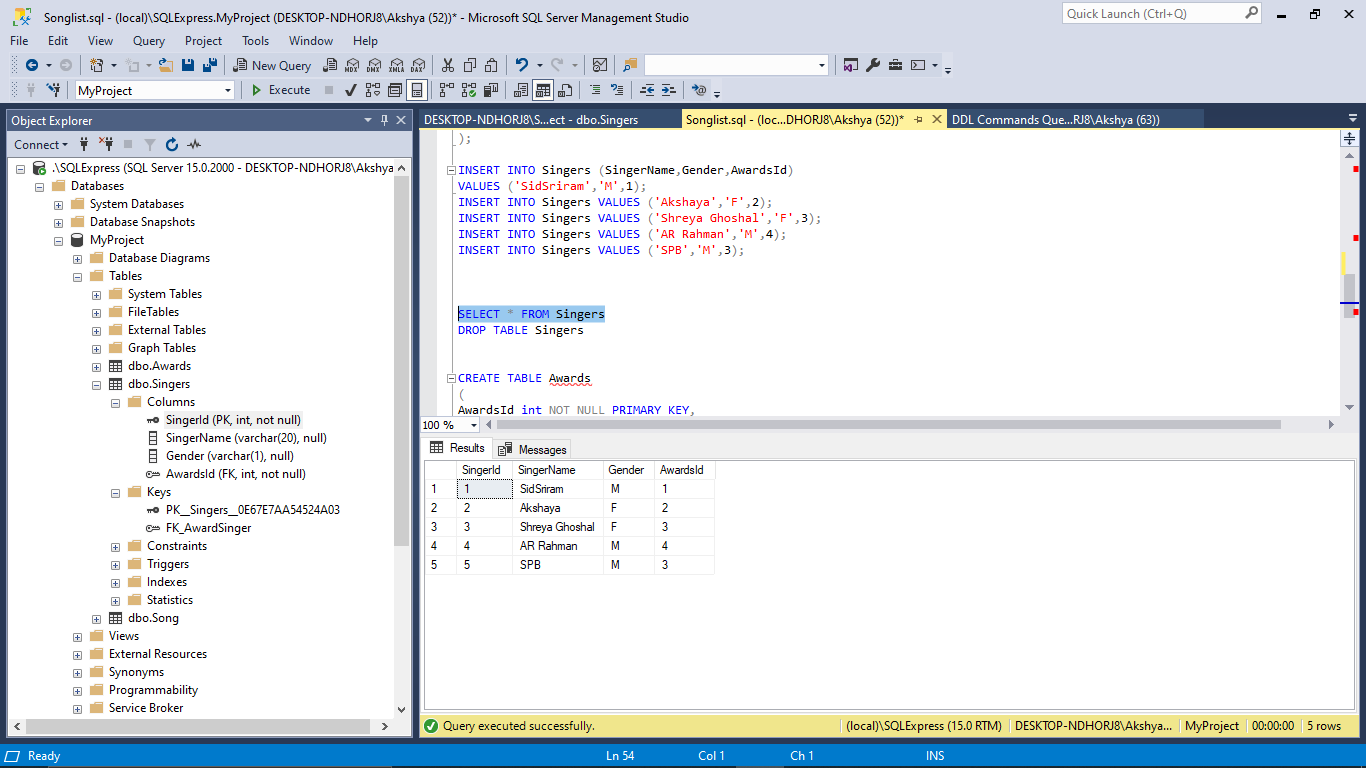
INSERT INTO Awards VALUES (1992,'Filmfare');

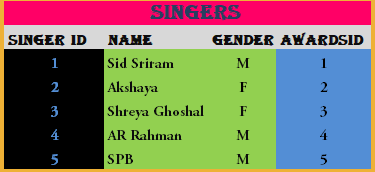
INSERT INTO Awards VALUES (1992,'BillBoard');

INSERT INTO Awards VALUES (1960,'American');

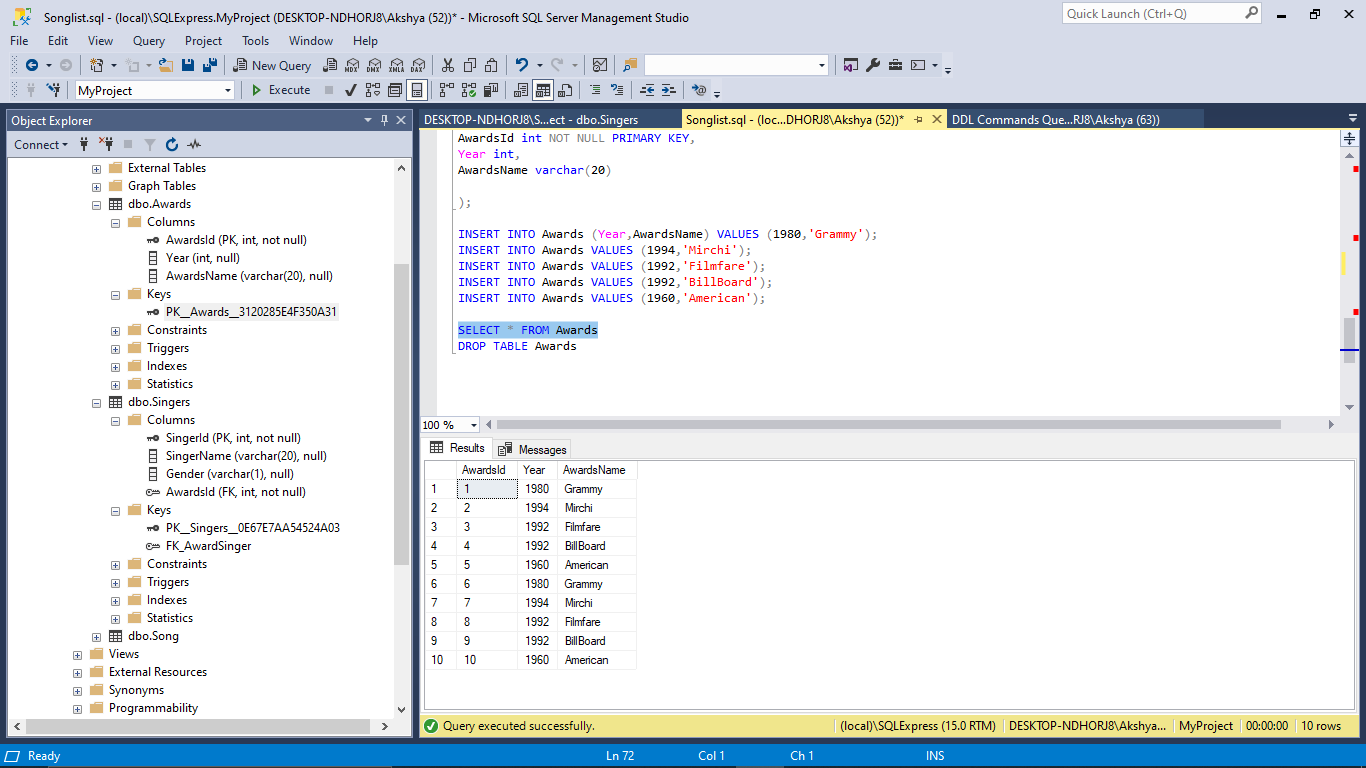
SELECT \* FROM Awards

**SINGERS TABLE**

****

****

**AWARDS TABLE**

****

****

**API**

**Authentication with login method**

**MethodName: LoginTest**

Parameter : userId,Password

Check : Login when the userId &Password is correct (if True)

Return : If it is equal then login

**MethodName: LoginFailTest**

Parameter : userId,Password

Check : Login when the userId &Password is correct (if False)

Return : If it is not equal then return error message

**CRUD API**

|  |  |  |
| --- | --- | --- |
| **CRUD** | **REST Method** | **SQL Statement** |
| Create | POST | INSERT |
| Read | GET | SELECT |
| Update | PUT | UPDATE |
| Delete | DELETE | DELETE |

**Testing**

**Test plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case Number | Step  Number | Description | Return  Value | Expected | Actual |
|  | 1 | GetSong | If we want SongId 1,then it returns | songId 1parameters are read such as Name,Genre,Instruments & Directors | Same as Expected |
|  | 2 | GetAllSong | SongName,,Genre, Instruments & Directors | Name,Genre,Instruments & Directors from the DB | Same as Expected |
| ST- 001 | 3 | Insert Song | Song Id | >0 |  |
|  | 4 | Get Song for inserted SongId | Inserted Song | NOT NULL Name = Inserted SongName  Genre = Inserted Genre  Instruments = Inserted Instruments  Directors = Inserted Directors | Same as Expected |
|  | 5 | Update Song | Number of Rows | = 1 |  |
|  | 6 | Get Song for updated Student | Updated Song | NOT NULL Name = Updated name  Genre = Updated Genre  Instruments = Updated Instruments  Directors =Updated Directors | Same as Expected |
|  | 7 | Delete Student | Has Been Deleted | True | Same as Expected |
|  | 8 | Get Deleted Student | Student | NULL |  |

**Unit testing of db**

**DB NAME : MYPROJECT**

**TABLE NAME : SONG**

**1).Method Name : GetSongTest**

Description : Test the song from their respective Id

Parameter : None

Return : Test the datas from Song Table

**2).Method Name : InsertSongTest**

Description : Insert the Parameters to song Table

Parameter : Name,Genre,Instruments & Directors

Return : Test the inserteddatas from Song Table

Check : i). SondId is Greater then 0

ii).Get the inserted song and then validate it with the song name

**2).Method Name : InsertSongTest**

Description : Insert the Parameters to song Table

Parameter : Name,Genre,Instruments & Directors

Return : Test the inserteddatas from Song Table

Check : i). SondId is Greater then 0

ii).Get the inserted song and then validate it with the song name

**3). Update the Song**

Description : Update the Parameters to songId to the Existing songId

Parameter : Name,Genre,Instruments & Directors

Return : Get the updated Song

Check : i). Test the Updateddatas from Song Table

ii). validate it with the song name both the songs are equal

**4). Delete the Song**

Description : Delete the Parameters from song table uding SongId

Parameter : Name,Genre,Instruments & Directors

Return : Get the Deleted Song (NULL)

Check : i). Test the deleted Song from Song Table

**API TESTING USING NUNIT**

**DB Name: MyProject**

**Table Name: Song**

1).**Method Name : CreateSongAsync**

Description : It gives the response the the URL and Deserialize the data from song table

2).**Method Name : GetSongAsync**

Description : It gives the response the the URL and Deserialize the data from song table for the given respective SongId.

Return : Returns the datas from songId

3).**Method Name : GetSongsAsync**

Description : It gives the response the the URL and Deserialize the data from song table

Return : Returns the all datas from songTable

4).**Method Name : UpdateSongAsync**

Description : It gives the response the the URL and Deserialize the data to the song table .

Return : Returns the SongData

5).**Method Name : DeleteSongAsync**

Description : It deletes the data from the request

Return : Returns the number of rows