```
package com.internshala.echo.databases
import android.content.ContentValues
import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
import com.internshala.echo.Songs
/**
 * Created by Harsh Deep Singh on 2/20/2018.
 */
class EchoDatabase : SQLiteOpenHelper {

    /*List for storing the favorite songs*/
    var _songList = ArrayList<Songs>()

    val DB_NAME = "FavoriteDatabase"
    val TABLE_NAME = "FavoriteTable"
    val COLUMN_ID = "SongID"
    val COLUMN_SONG_TITLE = "SongTitle"
    val COLUMN_SONG_ARTIST = "SongArtist"
    val COLUMN_SONG_PATH = "SongPath"

    override fun onCreate(db: SQLiteDatabase?) {
        db?.execSQL("CREATE TABLE " + TABLE_NAME + "( " + COLUMN_ID +
                " INTEGER," + COLUMN_SONG_ARTIST + " STRING," + COLUMN_SONG_TITLE + "
STRING,"
                + COLUMN_SONG_PATH + " STRING);")
    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {
    }

    constructor(context: Context?, name: String?, factory: SQLiteDatabase.CursorFactory?,
version: Int) : super(context, name, factory, version) {}

    fun storeAsFavorite(id: Int?, artist: String?, songTitle: String?, path: String?) {
        val db = this.writableDatabase
        val contentValues = ContentValues()
        contentValues.put(COLUMN_ID, id)
        contentValues.put(COLUMN_SONG_ARTIST, artist)
        contentValues.put(COLUMN_SONG_TITLE, songTitle)
        contentValues.put(COLUMN_SONG_PATH, path)
        db.insert(TABLE_NAME, null, contentValues)
        db.close()
    }

    /*This method asks the database for the list of Songs stored as favorite*/
    fun queryDBList(): ArrayList<Songs>? {

        /*Here a try-catch block is used to handle the exception as no songs in the
database can result in null-pointer exception*/

        try {
            val db = this.readableDatabase

            /*The SQL query used for obtaining the songs is :
            * SELECT * FROM FavoriteTable
            * The query returns all the items present in the table*/
            val query_params = "SELECT * FROM " + TABLE_NAME
            var cSor = db.rawQuery(query_params, null)
```

```
            /*The cSor stores the result obtained from the database
            * The function moveToFirst() checks if there are any entries or not*/
            if (cSor.moveToFirst()) {

                /*If 1 or more rows are returned then we store all the entries into the
array list _songList*/
                do {

                    var _id = cSor.getInt(cSor.getColumnIndexOrThrow(COLUMN_ID))
                    var _artist =
cSor.getString(cSor.getColumnIndexOrThrow(COLUMN_SONG_ARTIST))
                    var _title =
cSor.getString(cSor.getColumnIndexOrThrow(COLUMN_SONG_TITLE))
                    var _songPath =
cSor.getString(cSor.getColumnIndexOrThrow(COLUMN_SONG_PATH))
                    _songList.add(Songs(_id.toLong(), _title, _artist, _songPath, 0))
                }

                    /*This task is performed till there are items present*/
                while (cSor.moveToNext())
            }

            /*Otherwise null is returned*/
            else {
                return null
            }
        }

        /*If there was any exception then it is handled by this*/
        catch (e: Exception) {
            e.printStackTrace()
        }

        /*Finally we return the songList which contains the songs present inside the
database*/
        return _songList
    }

    /*This function is created for checking whether a particular song is a favorite or
not*/
    fun checkifIdExists(_id: Int): Boolean {

        /*Random id which does not exist
        * We know that this id can never exist as the song id cannot be less than 0*/
        var storeId = -1090
        val db = this.readableDatabase

        /*The query for checking the if id is present or not is
        * SELECT * FROM FavoriteTable WHERE SongID = <id_of_our_song>*/
        val query_params = "SELECT * FROM " + TABLE_NAME + " WHERE SongID = '$_id'"
        val cSor = db.rawQuery(query_params, null)
        if (cSor.moveToFirst()) {
            do {

                /*Storing the song id into the variable storeId*/
                storeId = cSor.getInt(cSor.getColumnIndexOrThrow(COLUMN_ID))
            } while (cSor.moveToNext())
        } else {
            return false
        }
```

```
        /*Here we need to return a boolean value i.e. true or false
        * Hence we check if the store id is not equal to -1090 then we return true, else we
return false*/
        return storeId != -1090
    }

    /*This function is used to delete the songs from the favorite if the user the user
removes any song from the favorite list*/
    fun deleteFavourite(_id: Int) {
        val db = this.writableDatabase

        /*The delete query is used to perform the delete function*/
        db.delete(TABLE_NAME, COLUMN_ID + " = " + _id, null)

        /*Here is also we close the database connection
        * Note that we only close the database whenever we open in writable mode*/
        db.close()
    }
}
```