Refer <https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html> and <https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html> for detailed information on SQLite open helper.

**SQLiteDatabase class methods used for this example:**

* method for inserting a row into the database

**long insert (**[**String**](https://developer.android.com/reference/java/lang/String.html) **table,**[**String**](https://developer.android.com/reference/java/lang/String.html) **nullColumnHack,** [**ContentValues**](https://developer.android.com/reference/android/content/ContentValues.html) **values)**

Returns the row ID of the newly inserted row, or -1 if an error occurred

ContentValues: this map contains column values for the row. The keys should be the column names and the values the column values



nullColumnHack parameter provides the name of nullable column name to explicitly insert a NULL into in the case where your values is empty.

* method for deleting rows in the database

**int delete (**[**String**](https://developer.android.com/reference/java/lang/String.html) **table,** [**String**](https://developer.android.com/reference/java/lang/String.html) **whereClause,** [**String[]**](https://developer.android.com/reference/java/lang/String.html) **whereArgs)**

Returns the number of rows affected

whereClause- the optional WHERE clause to apply when deleting. Passing null will delete all rows.

whereArgs - You may include ?s in the where clause, which will be replaced by the values from whereArgs.

* method for updating rows in the database
* **int update (**[**String**](https://developer.android.com/reference/java/lang/String.html) **table,** [**ContentValues**](https://developer.android.com/reference/android/content/ContentValues.html) **values,** [**String**](https://developer.android.com/reference/java/lang/String.html) **whereClause,** [**String[]**](https://developer.android.com/reference/java/lang/String.html) **whereArgs)**

Returns the number of rows affected

ContentValues: a map from column names to new column values. null is a valid value that will be translated to NULL.

whereClause-- the optional WHERE clause to apply when updating. Passing null will update all rows.

whereArgs String: You may include ?s in the where clause, which will be replaced by the values from whereArgs. The values will be bound as Strings.

* Query the given table, returning a [Cursor](https://developer.android.com/reference/android/database/Cursor.html) over the result set.

[**Cursor**](https://developer.android.com/reference/android/database/Cursor.html) **query (**[**String**](https://developer.android.com/reference/java/lang/String.html) **table,** [**String[]**](https://developer.android.com/reference/java/lang/String.html) **columns,** [**String**](https://developer.android.com/reference/java/lang/String.html) **selection,**[**String[]**](https://developer.android.com/reference/java/lang/String.html) **selectionArgs,**

[**String**](https://developer.android.com/reference/java/lang/String.html) **groupBy,** [**String**](https://developer.android.com/reference/java/lang/String.html) **having,** [**String**](https://developer.android.com/reference/java/lang/String.html) **orderBy)**

columns : A list of which columns to return. Passing null will return all columns

selection: A filter declaring which rows to return, formatted as an SQL WHERE clause (excluding the WHERE itself). Passing null will return all rows for the given table.

selectionArgs: You may include ?s in selection, which will be replaced by the values from selectionArgs, in order that they appear in the selection.

groupBy :A filter declaring how to group rows, formatted as an SQL GROUP BY clause (excluding the GROUP BY itself). Passing null will cause the rows to not be grouped.

having: A filter declare which row groups to include in the cursor, if row grouping is being used, formatted as an SQL HAVING clause (excluding the HAVING itself). Passing null will cause all row groups to be included, and is required when row grouping is not being used.

orderBy: How to order the rows, formatted as an SQL ORDER BY clause (excluding the ORDER BY itself). Passing null will use the default sort order, which may be unordered.

**Constructor and methods of SQLiteOpenHelper**

**SQLiteOpenHelper (Context context, String name, SQLiteDatabase.CursorFactory factory, int version)**

Create a helper object to create, open, and/or manage a database. This method always returns very quickly. The database is not actually created or opened until one of getWritableDatabase() or getReadableDatabase() is called.

* **SQLiteDatabase getWritableDatabase ()**

Create and/or open a database that will be used for reading and writing. Once opened successfully, the database is cached, so you can call this method every time you need to write to the database.

* **void onCreate (SQLiteDatabase db)**

Called when the database is created for the first time. This is where the **creation of tables** and the initial population of the tables should happen.

Example:

Example: Create a table ‘Employee’ inside a database “EmpDBase.db” , having columns empid(Integer primary key),Emp\_name(TEXT NOT NULL) and Emp\_address (TEXT NOT NULL)

Perform Insert, Update and delete operations on the created table through frontend using SQLite open helper class.

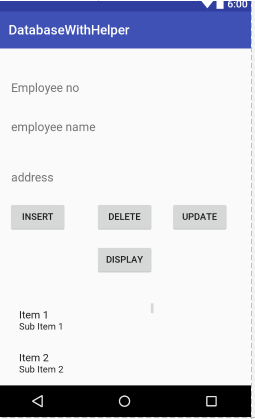
Also Display all the data from table (order by emp\_name) on list view of same activity.

**Steps**

1>create UI for app as per requirements

2>Create a new class (MyDBHelper in this example) that extends SQLiteOpenHelper and all the operations related to database(like open/create database, open/create table, insert update delete data fetch.... are implemented here. write different methods for insert, update, delete, data fetch operations using built-in methods insert(),update(),delete(),query() etc. of SQLiteDatabase class.

3>In MainActivity.java class create an object of above created class and call required methods from this object on the button click.



*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/activity\_main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:paddingBottom="@dimen/activity\_vertical\_margin"  
 android:paddingLeft="@dimen/activity\_horizontal\_margin"  
 android:paddingRight="@dimen/activity\_horizontal\_margin"  
 android:paddingTop="@dimen/activity\_vertical\_margin"  
 tools:context="com.example.student.databasewithhelper.MainActivity"**>  
  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textPersonName"  
 android:ems="10"  
 android:layout\_alignParentTop="true"  
 android:layout\_marginTop="23dp"  
 android:id="@+id/et\_empno"  
 android:layout\_alignParentStart="true"  
 android:hint="Employee no"** />  
  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textPersonName"  
 android:ems="10"  
 android:layout\_below="@+id/et\_empno"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="17dp"  
 android:id="@+id/et\_ename"  
 android:hint="employee name"** />  
  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textPersonName"  
 android:ems="10"  
 android:layout\_below="@+id/et\_ename"  
 android:layout\_alignParentStart="true"  
 android:layout\_marginTop="34dp"  
 android:id="@+id/et\_adds"  
 android:hint="address"** />  
  
 <**Button  
 android:text="DELETE"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/bt\_del"  
 android:layout\_alignBaseline="@+id/bt\_ins"  
 android:layout\_alignBottom="@+id/bt\_ins"  
 android:layout\_centerHorizontal="true"** />  
  
 <**Button  
 android:text="UPDATE"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/bt\_update"  
 android:layout\_marginStart="26dp"  
 android:layout\_alignTop="@+id/bt\_del"  
 android:layout\_toEndOf="@+id/bt\_disp"** />  
  
 <**Button  
 android:text="display"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/bt\_disp"  
 android:layout\_marginTop="17dp"  
 android:layout\_below="@+id/bt\_del"  
 android:layout\_alignStart="@+id/bt\_del"** />  
  
 <**Button  
 android:text="INSERT"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/bt\_ins"  
 android:layout\_centerVertical="true"  
 android:layout\_alignParentStart="true"** />  
  
 <**ListView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_below="@+id/bt\_disp"  
 android:layout\_toStartOf="@+id/bt\_update"  
 android:layout\_marginTop="42dp"  
 android:id="@+id/lv\_getEmp"** />  
  
</**RelativeLayout**>

**MainActivity.java**

**package** com.example.student.databasewithhelper;  
  
**import** android.database.Cursor;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.\*;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** MainActivity **extends** AppCompatActivity {  
 MyDBHelper **dbHelper**;  
EditText **num**,**name**,**ads**;  
 Button **ins**,**disp**,**updt**,**delt**;  
 ListView **disp\_data**;  
 List<String> **res**=**new** ArrayList<String>();  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 **num**=(EditText)findViewById(R.id.***et\_empno***);  
 **name**=(EditText)findViewById(R.id.***et\_ename***);  
 **ads**=(EditText)findViewById(R.id.***et\_adds***);  
 **ins**=(Button)findViewById(R.id.***bt\_ins***);  
 **updt**=(Button)findViewById(R.id.***bt\_update***);  
 **delt**=(Button)findViewById(R.id.***bt\_del***);  
 **disp**=(Button)findViewById(R.id.***bt\_disp***);  
 **disp\_data**=(ListView)findViewById(R.id.***lv\_getEmp***);  
 **dbHelper**=**new** MyDBHelper(MainActivity.**this**);  
 **ins**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **long** insres=**dbHelper**.insert(Integer.*parseInt*(**num**.getText().toString()),**name**.getText().toString().trim(),**ads**.getText().toString().trim());  
 **if**(insres==-1)  
 Toast.*makeText*(MainActivity.**this**,**"insert has some error"**,Toast.***LENGTH\_SHORT***).show();  
 **else** Toast.*makeText*(MainActivity.**this**,**"inserted "**,Toast.***LENGTH\_SHORT***).show();  
  
 }  
 });  
 **updt**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **long** upres=**dbHelper**.update(Integer.*parseInt*(**num**.getText().toString()),**name**.getText().toString().trim(),**ads**.getText().toString().trim());  
 **if**(upres==0)  
 Toast.*makeText*(MainActivity.**this**,**"Update has some error"**,Toast.***LENGTH\_SHORT***).show();  
 **else if**(upres==1)  
 Toast.*makeText*(MainActivity.**this**,**"Updated"**,Toast.***LENGTH\_SHORT***).show();  
 }  
 });  
 **delt**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **long** delres=**dbHelper**.delete(Integer.*parseInt*(**num**.getText().toString()));  
 **if**(delres==0)  
 Toast.*makeText*(MainActivity.**this**,**"Delete has some error"**,Toast.***LENGTH\_SHORT***).show();  
 **else** Toast.*makeText*(MainActivity.**this**,**"Deleted "**,Toast.***LENGTH\_SHORT***).show();  
  
 }  
 });  
 **disp**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Cursor c=**dbHelper**.getAllEmployee();  
 **if**(c!=**null**)  
 {  
 **if**(c.moveToFirst()) {  
 **do** {  
 String eid=c.getString(c.getColumnIndex(**dbHelper**.*ID*));  
 String enam=c.getString(c.getColumnIndex(**dbHelper**.*NAME*));  
 String eads=c.getString(c.getColumnIndex(**dbHelper**.*ADDRESS*));  
 String row=eid +**" "**+enam +**" "**+eads;  
 **res**.add(row);  
 } **while** (c.moveToNext());  
 }  
 }  
 ArrayAdapter<String> adapter=**new** ArrayAdapter<String>(MainActivity.**this**,android.R.layout.***simple\_list\_item\_1***,**res**);  
 **disp\_data**.setAdapter(adapter);  
 }  
 });  
  
 }  
  
 }

**MyDBHelper.java**

**package** com.example.student.databasewithhelper;  
  
**import** android.content.ContentValues;  
**import** android.content.Context;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.database.sqlite.SQLiteOpenHelper;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
**import** android.database.Cursor;  
  
*/\*\*  
 \* Created by student on 16-Feb-18.  
 \*/***public class** MyDBHelper **extends** SQLiteOpenHelper {  
 **private static final** String ***DBNAME*** = **"EmpDBase.db"**;  
 **private static final** Integer ***VERSION*** = 1;  
 **public static** String *TABLE\_NAME* = **"Employee"**;  
 **public static** String *ID* = **"empid"**;  
 **public static** String *NAME* = **"Emp\_name"**;  
 **public static** String *ADDRESS* = **"Emp\_ads"**;  
 **private** SQLiteDatabase **dbs**;  
  
  
 **public** MyDBHelper(Context context) {  
 **super**(context, ***DBNAME***, **null**, ***VERSION***);  
 }  
  
 @Override  
 **public void** onCreate(SQLiteDatabase db) {  
 String qry\_table = **"CREATE TABLE "** + *TABLE\_NAME* + **"("** + *ID* + **" INTEGER PRIMARY KEY,"** + *NAME* + **" TEXT NOT NULL,"** + *ADDRESS* + **" TEXT NOT NULL);"**;  
 db.execSQL(qry\_table);  
  
 }  
  
 @Override  
 **public void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) {  
  
 }  
  
  
 **public long** insert(**int** id, String name, String address) {  
 **dbs** = getWritableDatabase();  
 ContentValues values = **new** ContentValues();  
 values.put(*ID*, id);  
 values.put(*NAME*, name);  
 values.put(*ADDRESS*, address);  
 **return dbs**.insert(*TABLE\_NAME*, **null**, values);  
 }  
  
 **public long** delete(**int** id) {  
 String where = *ID* + **" = "** + id;  
  
 **return dbs**.delete(*TABLE\_NAME*, where, **null**);  
 }  
  
 **public long** update(**int** id, String name, String address) {  
 ContentValues values = **new** ContentValues();  
 values.put(*NAME*, name);  
 values.put(*ADDRESS*, address);  
 String where = *ID* + **" = "** + id;  
 **return dbs**.update(*TABLE\_NAME*, values, where, **null**);  
 }  
  
 **public** Cursor getAllEmployee() {  
 *//String query = "SELECT \* FROM " + TABLE\_NAME;*  
 *//return dbs.rawQuery(query, null);* **return dbs**.query(*TABLE\_NAME*,**null**,**null**,**null**,**null**,**null**,*NAME*);  
 }  
}

