LESSON – 9

SQLITE DATABASE

Agenda

- Introduction to SQLite Database
- SQLiteOpenHelper
- Content values
- Cursor
- CRUD Queries
- Hands on Example-1
- Hands on Example-1
- SQLite Browser

INTRODUCTION

- SQLite is a lightweight, embedded relational database management system that is included as part of the Android framework and provides a mechanism for implementing organized persistent data storage for Android applications.
- In addition to the SQLite database, the Android framework also includes a range of Java classes that may be used to create and manage SQLite based databases and tables.

Introduction

- SQLite is an open source storage database that manages all database related queries.
- Maintain structures data.
  - Server less and self contained.
  - Full documentation is available at http://www.sqlite.org
- Android provides its own API to deal with database connectivity. This API is provided in android.database and android.database.sqlite packages.
- In apps, we interact with a SQLite database using the SQLiteOpenHelper class and the SQLiteDatabase class.
- For the full syntax of all SQLite comments see https://sqlite.org/lang.html

SQLiteOpenHelper

- Using SQLiteOpenHelper
  - The main functionality of the class is to open the database if it exists, create if it does not, and upgrade the version as required.
  - It provides a constructor to construct a helper class by subclassing this class and overriding the methods named onCreate() an onUpgrade().

  Syntax:

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

  where,

  - context – represent the context to create or open the database
  - name – represents the name of the database
  - factory – represents the factory class used for creating the cursor object ,its an optional parameter passed as null.
  - version – represent the number of the database

SQLiteOpenHelper Example

Create your DBHelper class which extends from SQLiteOpenHelper and provide a constructor for this class and call super class constructor by passing 4 parameters.

```
public class EmployeeDBHelper extends SQLiteOpenHelper {

    private   static final  String DATABASE_NAME  =  "empdb.db";
    private   static final  int    DATABASE_VERSION   =  1;


// Constructor
public EmployeeDBHelper(Context context) {
        super(context, DATABASE_NAME, null, 1);
    }

}
```

- Some of the commonly used methods of this class are as follows:
- onCreate (SQLiteDatabase db)
  - The method is invoked when the database is created for the first time. This is where the table is created and populated. The database name is passed as an argument.
- onUpgrade (SQLiteDatabase db, int oldVersion, int newVersion)
  - The method is invoked when the database needs to be upgraded. The implementation should use this method to drop tables, add tables, or perform any other operations such as the need to upgrade to the new schema version.

Note : Refer SQLite version from

https://developer.android.com/reference/android/database/sqlite/package-summary.html

Employee Table

Column	Data Type
Id	Number
Name	Varchar
Desig	Varchar
Dept	Varchar

Example – Override onCreate() and onUpgrade()

```
public class EmployeeDBHelper extends SQLiteOpenHelper {

    private    static final  String DATABASE_NAME  =  "empdb.db";
    private    static final  int    DATABASE_VERSION   =  1;

                                    // Constructor
    public EmployeeDBHelper(Context context) {
        super(context, DATABASE_NAME, null, 1);
    }
                                  // Create a table
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create table employee(id number, name varchar(50),
                             desig varchar(50), dept varchar(50))");
    }
                                 // Upgrade Table
    @Override
    public void onUpgrade(SQLiteDatabase db, int i, int i1) {
    db.execSQL("drop table if exists employee");
    onCreate(db);
    }
}
```

Notable Methods of the SQLiteOpenHelper Class

- getWritableDatabase() – Opens or creates a database for reading and writing. Returns a reference to the database in the form of a SQLiteDatabase object.
- getReadableDatabase() – Creates or opens a database for reading only. Returns a reference to the database in the form of a SQLiteDatabase object.
- close() – Closes the database.

Content values

- ContentValues is a convenience class that allows key/value pairs to be declared consisting of table column identifiers and the values to be stored in each column.
- This class is of particular use when inserting or updating entries in a database table.
- Example
- ContentValues values=**new** ContentValues();

```
        values.put("id",Integer.parseInt(et1.getText().toString()));
        values.put("name",et2.getText().toString());
        values.put("desig",et3.getText().toString());
        values.put("dept", et4.getText().toString());
```

Cursor Class

- Cursor : A class provided specifically to provide access to the results of a database query. Key methods of this class are as follows:
  - close() – Releases all resources used by the cursor and closes it.
  - getCount() – Returns the number of rows contained within the result set.
  - moveToFirst() – Moves to the first row within the result set.
  - moveToLast() – Moves to the last row in the result set.
    moveToNext() – Moves to the next row in the result set.
    moveToPosition(int position): Moves the cursor to the specified position
  - get<type>() – Returns the value of the specified <type> contained at the specified column index of the row at the current cursor position (variations consist of getString(), getInt(), getShort(), getFloat() and getDouble()).

CRUD Queries

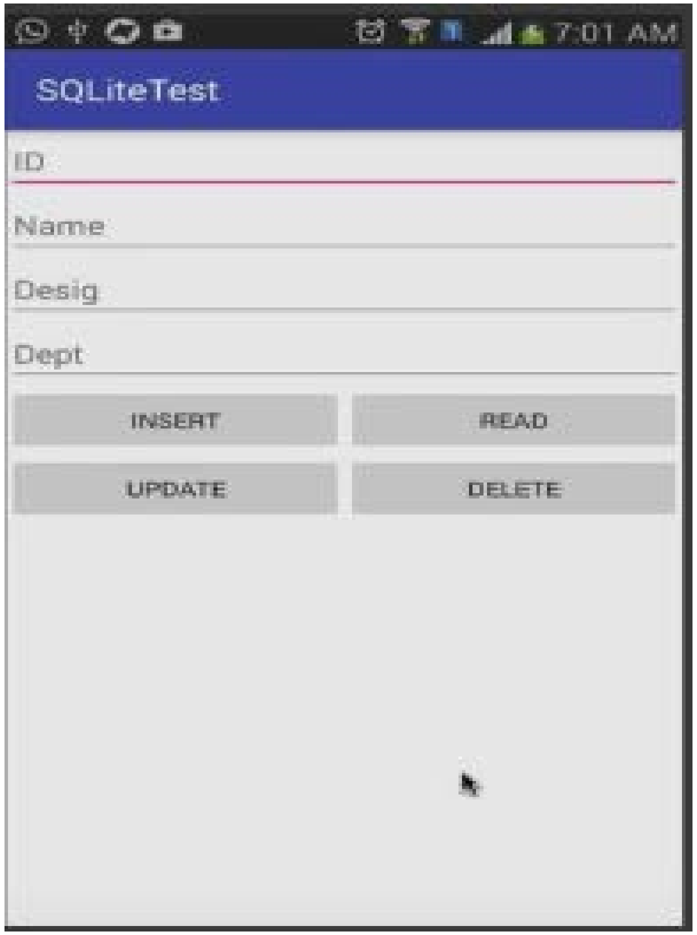
- SQLiteDatabase : The class has methods to create, delete, execute SQL commands, and perform other common database management tasks. Some of the commonly used methods of this class are as follows:
  - execSQL (String sql) - perform Create, Insert, Update and Delete
    - This method is used to execute a single SQL statement that is not a SELECT statement or any other SQL statement that returns data. It is used generally for creating table. The sql statement to be executed is passed as an argument.
  - long insert (String table, String nullColumnHack, ContentValues values)
    - It is a convenience method used for inserting a row into the database. The method returns the row ID of the newly inserted row, or -1 if an error takes place. The method accepts the table name and the initial column values for the row.
    - **long** count=**dBase**.insert(**"employee"**,**null**,values);
    - values argument is the object of ContentValues.

CRUD Queries

- **int update (String table, ContentValues values, String whereClause, String[] whereArgs)**
  - It is a convenience method used for updating rows in the database. The method returns the number of rows affected. It accepts the table name, new column values, and an optional where clause.
- **int delete (String table, String whereClause, String[] whereArgs)**
  - It is a convenience method used for deleting rows in the database. The method returns the number of rows affected if a where clause is passed, otherwise it returns 0.
- **Cursor query (String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy)**
  - The method queries the given table and returns a Cursor over the result set.
  - RawQuery methods can only be used for read queries.

Hands on example - 1

- Work with simple Employee database with single table to perform CRUD operation using SQLiteOpenHelper and SQLiteDatabase. Eg : Lesson9\EmployeeDB



activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/et1"
        android:hint="Enter ID" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/et2"
        android:hint="Enter Name" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/et3"
        android:hint="Enter Desig" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/et4"
        android:hint="Enter Dept" />
```

activity_main.xml

```
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <Button
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.5"
            android:text="Insert"
            android:onClick="insert"/>
        <Button
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.5"
            android:text="Read"
            android:onClick="read"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <Button
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.5"
            android:text="Update"
            android:onClick="update" />
        <Button
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.5"
            android:text="Delete"
            android:onClick="delete" />
    </LinearLayout>
</LinearLayout>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
  // Declare the objects for EditText control
    EditText et1,et2,et3,et4;
 // Declare an object for SQLiteDatabase
    SQLiteDatabase dBase;
    EmployeeDBHelper dbHelper;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
         // Initialize database objects
              dbHelper = new EmployeeDBHelper(this);
              dBase = dbHelper.getWritableDatabase();
        // get all the EditText components into your activity class
        et1=(EditText)findViewById(R.id.et1);
        et2=(EditText)findViewById(R.id.et2);
        et3=(EditText)findViewById(R.id.et3);
        et4=(EditText)findViewById(R.id.et4);
```

```java
// Insert Query
public void insert(View v){
    ContentValues values=new ContentValues();
    // Using put method to add key(Column name) and Value
     values.put("id",Integer.parseInt(et1.getText().toString()));
     values.put("name",et2.getText().toString());
     values.put("desig",et3.getText().toString());
     values.put("dept", et4.getText().toString());
    /* call insert method by passing three parameters which return a long value
    * 1. Table name,
    * 2. nullColumnHack is an optional String type - pass null - helps to insert null value in
the specified column by using this parameter
    * 3. Object of ContentValues - It is used to insert or update values into database tables*/
// How many records are affected with these query is a count
     long count=dBase.insert("employee",null,values);
     if(count!=-1){
         Toast.makeText(MainActivity.this, "Successfully
                 Inserted",Toast.LENGTH_SHORT).show();
         et1.setText("");
         et2.setText("");
         et3.setText("");
         et4.setText("");
     }else{
         Toast.makeText(MainActivity.this, "Unable to Insert",
        Toast.LENGTH_SHORT).show();
     }
}
```

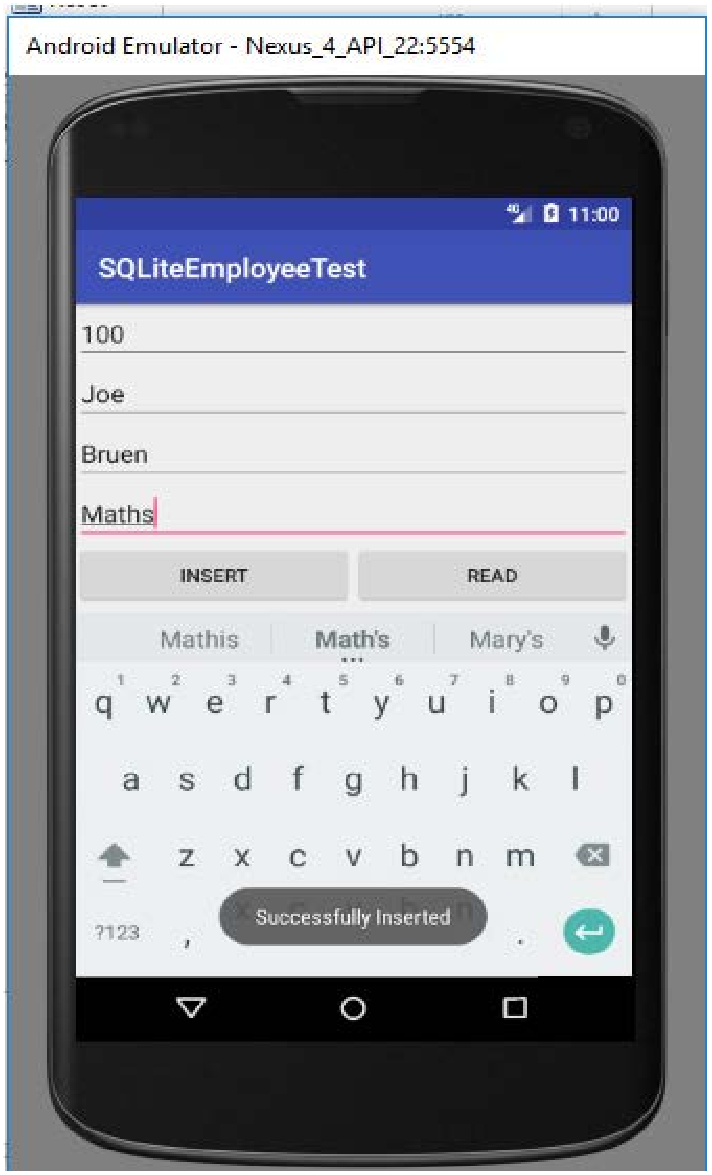
```java
// Read Query
public void read(View v){
    // Use query() method by passing seven parameter according to the requirement
   // Query returns the type of Cursor
      // select * from employee where id = et1.getText();
     Cursor c=dBase.query("employee",null,"id=?",
            new String[]{et1.getText().toString()},null,null,null);
 // This logic retrieve all the records from the employee select * from employee
  //Cursor c = dBase.query
        ("employee",null,null,null,null,null,null);
 // Initially cursor points -1, to set the pointer to 1 using moveToNext
     c.moveToNext();
     if(c.getCount()>0){ //Retrieve name,desig,dept
         et2.setText(c.getString(1));
         et3.setText(c.getString(2));
         et4.setText(c.getString(3));
     }
     else{
         Toast.makeText(getApplicationContext(),
         et1.getText().toString() + " Not found",
                      Toast.LENGTH_LONG).show();
             et1.setText("");
             et2.setText("");
             et3.setText("");
             et4.setText("");
     }
}
```

```
// Update Query
public void update(View v){
    ContentValues values=new ContentValues();
     values.put("name",et2.getText().toString());
     values.put("desig",et3.getText().toString());
     values.put("dept",et4.getText().toString());
    // update employee columns where id = et1.getText()
    int count=dBase.update("employee",values,"id=?",
            new String[]{et1.getText().toString()});
    if(count>0){ // Update happen in the table record
        Toast.makeText(getApplicationContext(),count+
         "row Updated...", Toast.LENGTH_LONG).show();
    }else{ // No Update
        Toast.makeText(getApplicationContext(),
         "Failed to Updated", Toast.LENGTH_LONG).show();
    }
}
```

```
// delete Query
public void delete(View v){
     // delete from employee where id = et1.getText();
   int count=
     dBase.delete("employee","id=?",new
     String[]{et1.getText().toString()});
   if(count>0){
     Toast.makeText(getApplicationContext(),
     count+"row deleted...",
     Toast.LENGTH_LONG).show();
    }
    else{
     Toast.makeText(getApplicationContext(),
     "Failed to delete",
    Toast.LENGTH_LONG).show();
    }
}
}
```

Run the App

Performing Insert Operation


Android Emulator - Nexus_4_API_22:5554

SQLiteEmployeeTest

100

Joe

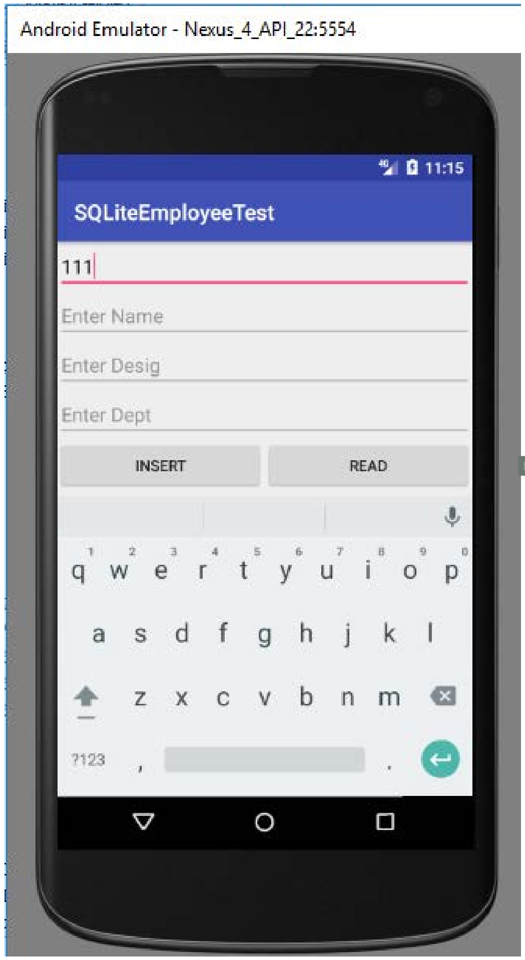
Bruen

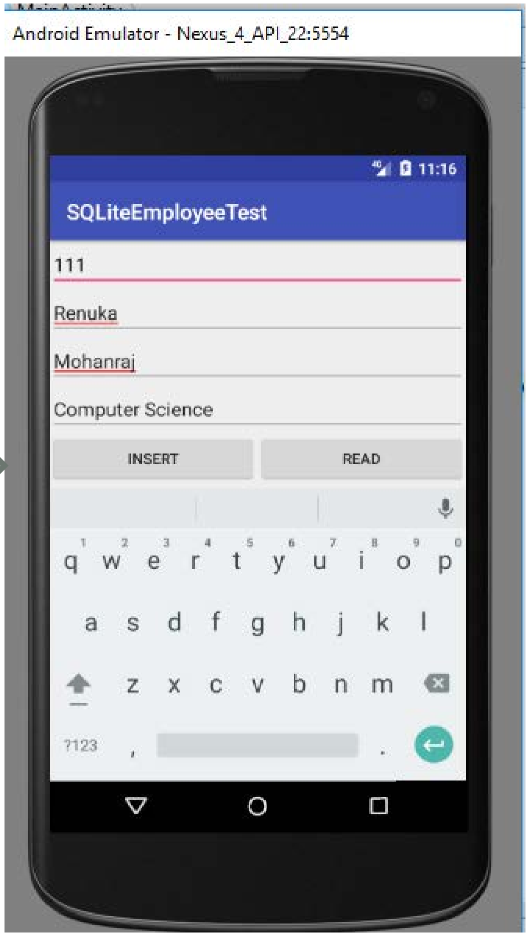
Maths

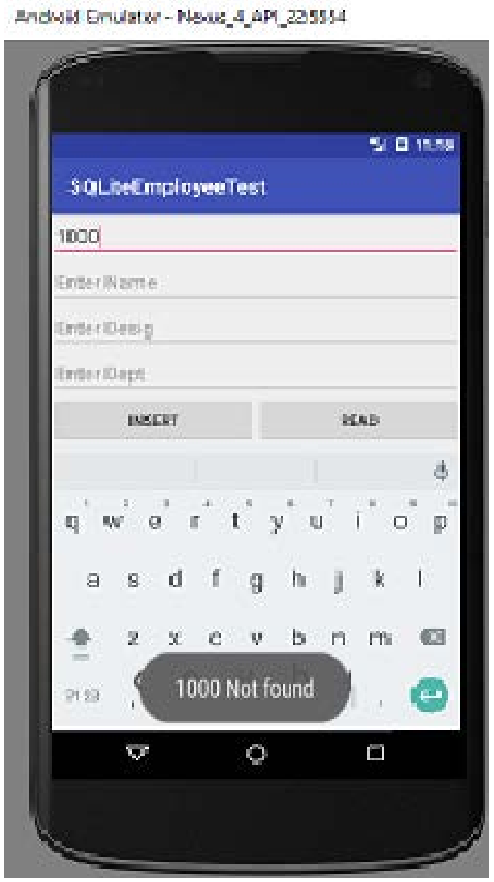
Successfully Inserted

Performing Read Operation

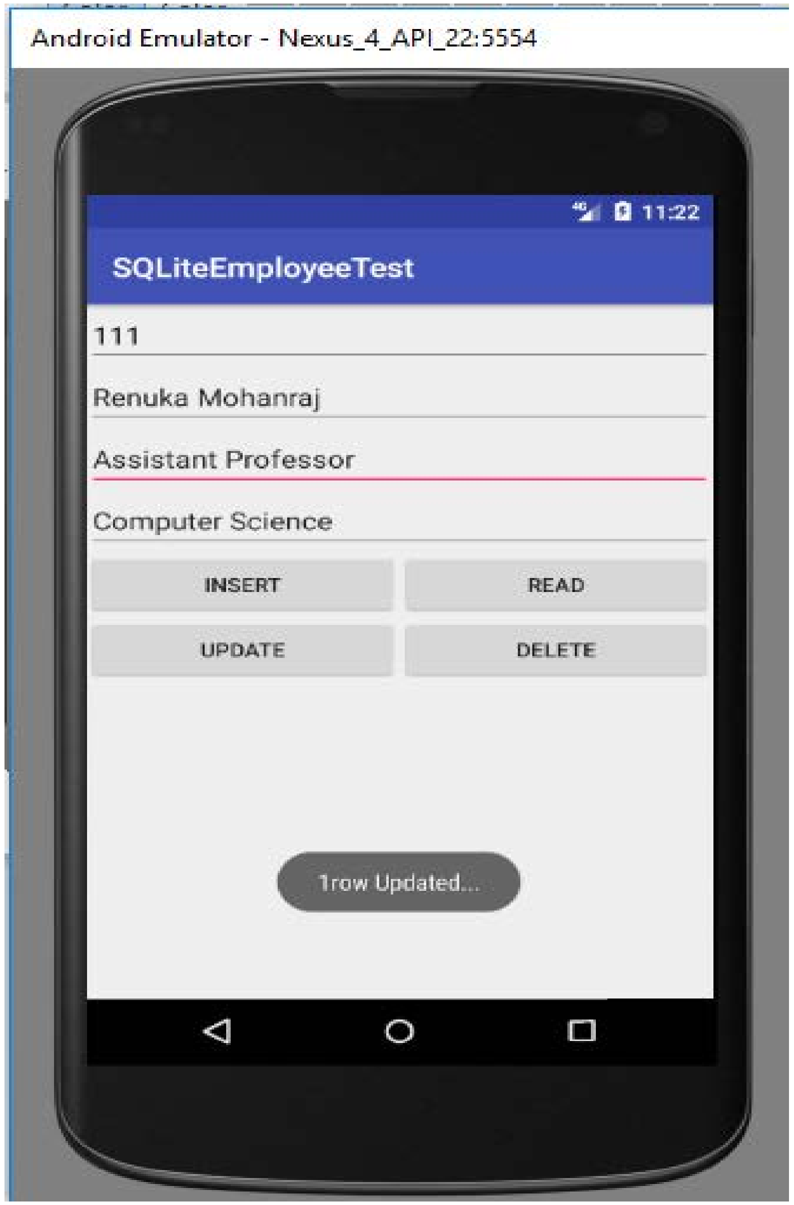
Entering id 111 and click the Read button, you will get the Employee record or else Not Found Toast Message







Performing Update Operation


Android Emulator - Nexus_4_API_22:5554

SQLiteEmployeeTest

111

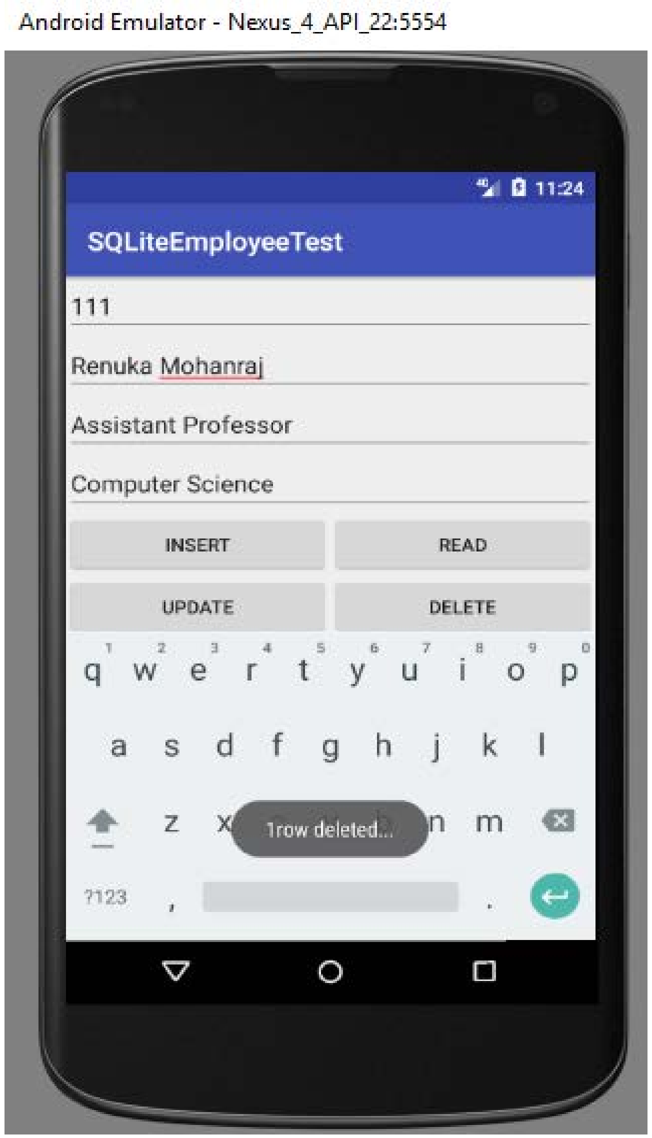
Renuka Mohanraj

Assistant Professor

Computer Science

1row Updated...

Performing Delete Operation


Android Emulator - Nexus_4_API_22:5554

SQLiteEmployeeTest

111

Renuka Mohanraj

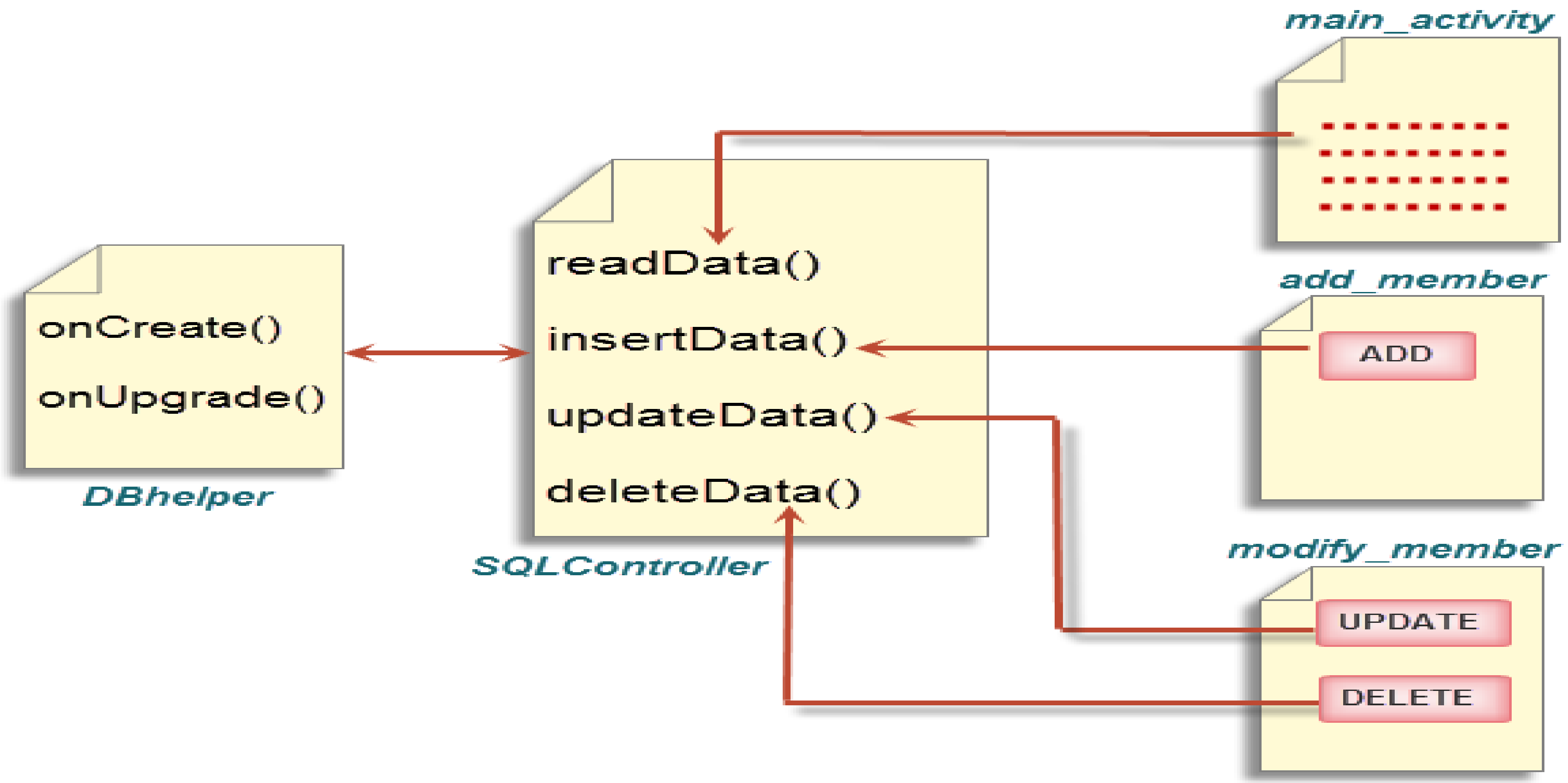
Assistant Professor

Computer Science

1row deleted...

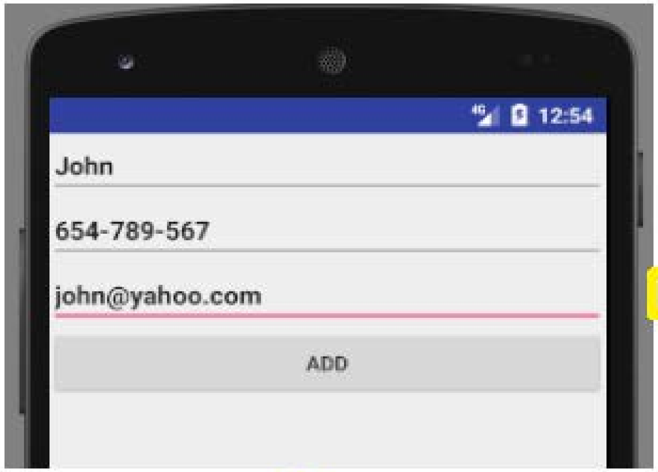
Hands on Example - 2

Here a database called contact with a table test to hold contactid, name, mobile and email and perform the following operation. Refer : Lesson9/CustomerSQLite



Adding Contacts

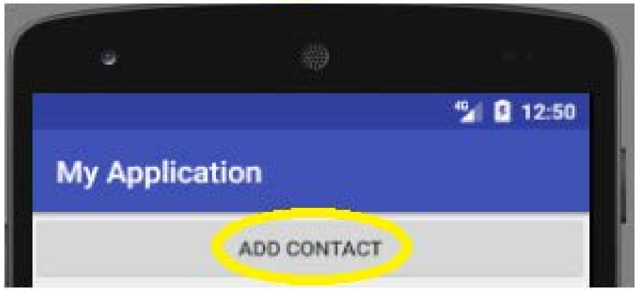
Add_member.java



John

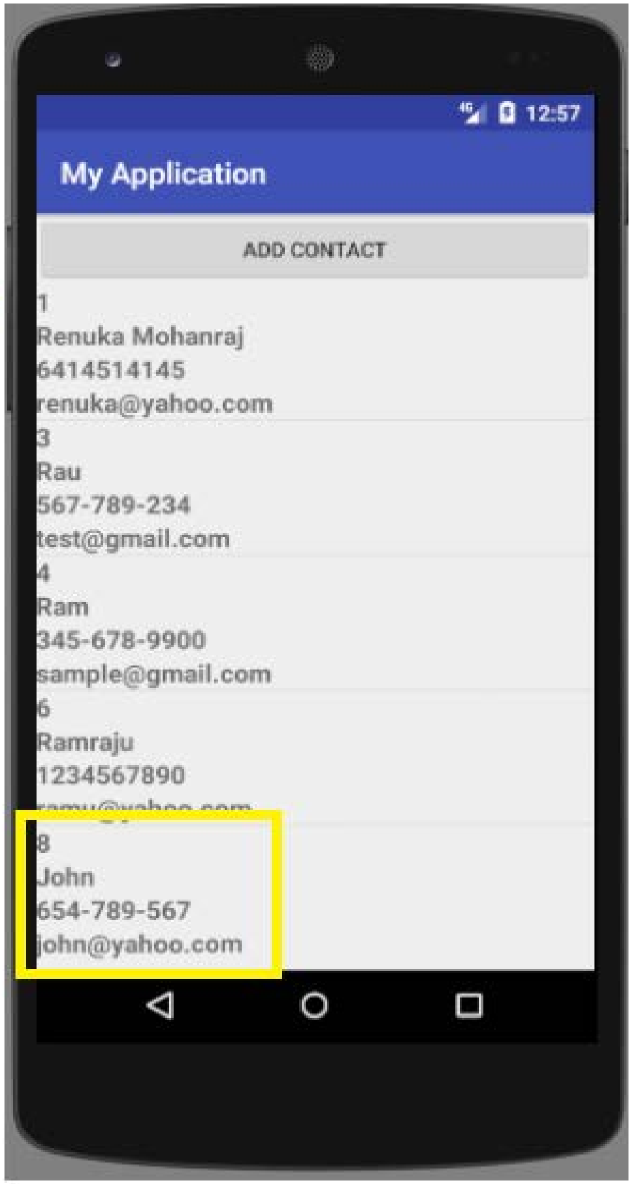
654-789-567

john@yahoo.com



MainActivity.java

MainActivity.java



1
Renuka Mohanraj
6414514145
renuka@yahoo.com

3
Rau
567-789-234
test@gmail.com

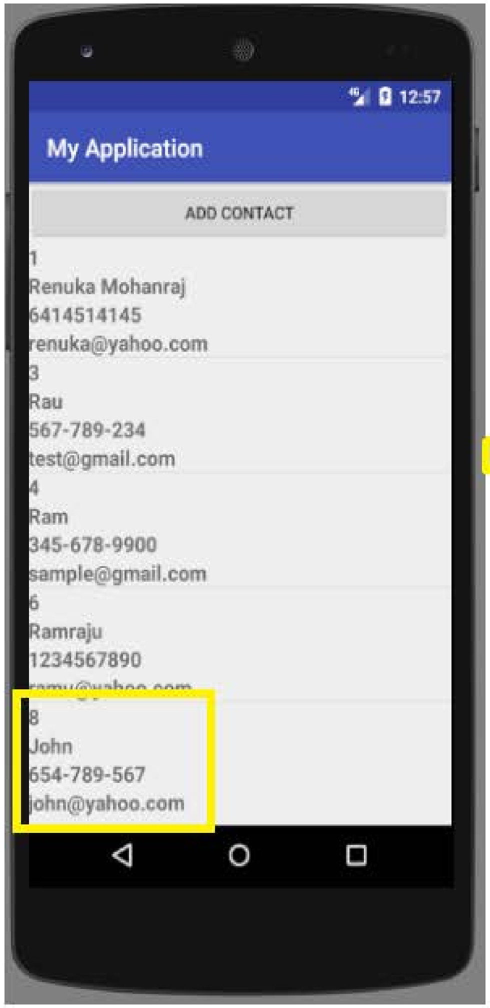
4
Ram
345-678-9900
sample@gmail.com

6
Ramraju
1234567890
ramu@yahoo.com

8
John
654-789-567
john@yahoo.com

Updating Contacts

Once you click on entry from the Listview, then you will get Modify_member activity. You can go for Update, changes will be reflect in the MainActivity ListView.


12:57

My Application

ADD CONTACT

1
Renuka Mohanraj
6414514145
renuka@yahoo.com

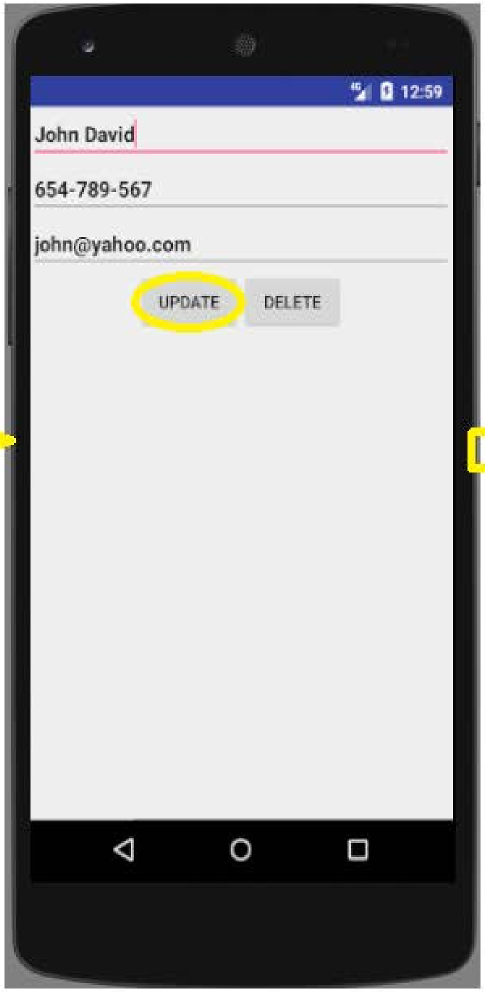
3
Rau
567-789-234
test@gmail.com

4
Ram
345-678-9900
sample@gmail.com

6
Ramraju
1234567890
ramu@yahoo.com

8
John
654-789-567
john@yahoo.com

MainActivity.java

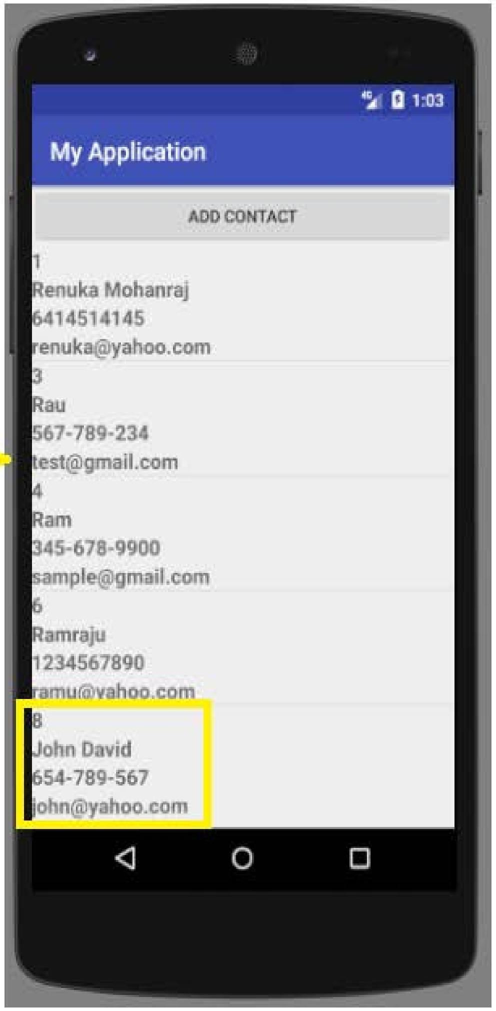

12:59

John David

654-789-567

john@yahoo.com

Modify_member.java


1:03

My Application

ADD CONTACT

1
Renuka Mohanraj
6414514145
renuka@yahoo.com

3
Rau
567-789-234
test@gmail.com

4
Ram
345-678-9900
sample@gmail.com

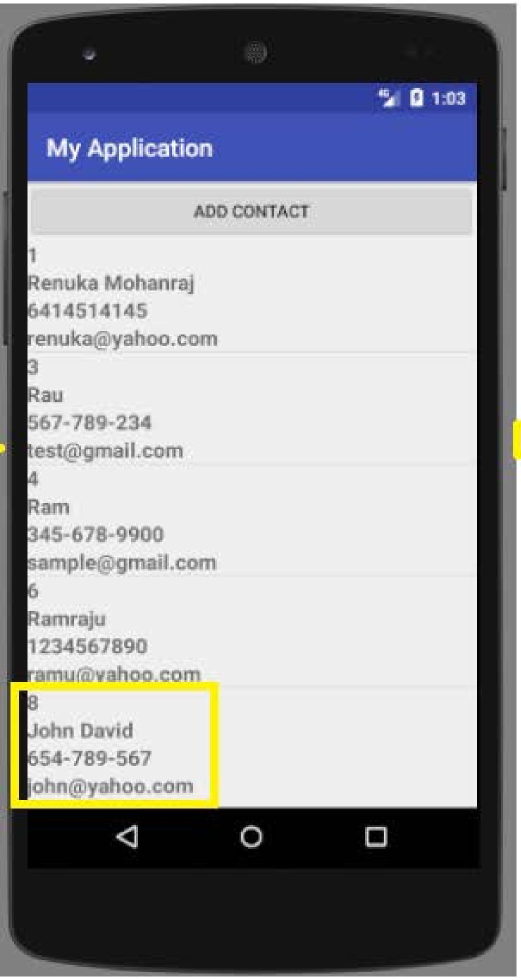
6
Ramraju
1234567890
ramu@yahoo.com

8
John David
654-789-567
john@yahoo.com

MainActivity.java

Deleting Contacts

Once you click on entry from the Listview, then you will get Modify_member activity. You can go for Delete, changes will be reflect in the MainActivity ListView.



1:03

My Application

ADD CONTACT

1
Renuka Mohanraj
6414514145
renuka@yahoo.com

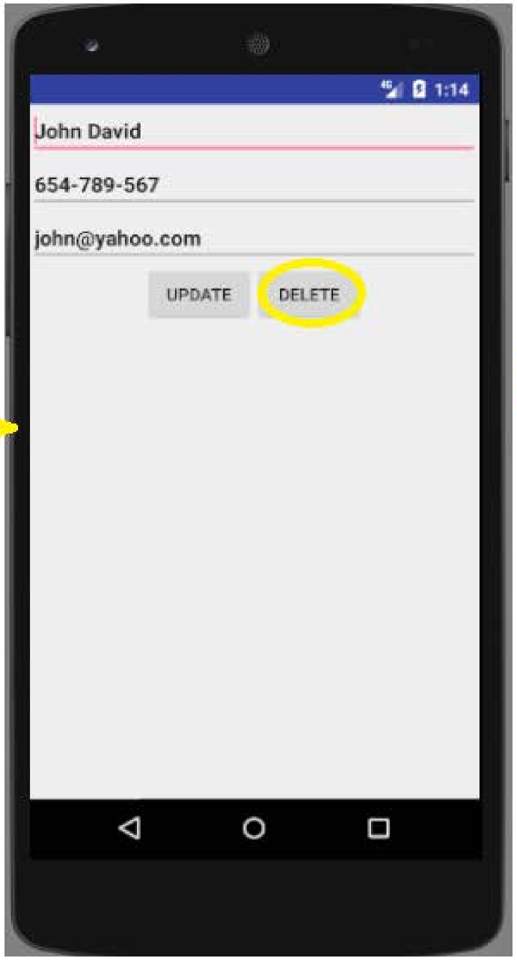
3
Rau
567-789-234
test@gmail.com

4
Ram
345-678-9900
sample@gmail.com

6
Ramraju
1234567890
ramu@yahoo.com

8
John David
654-789-567
john@yahoo.com

MainActivity.java



1:14

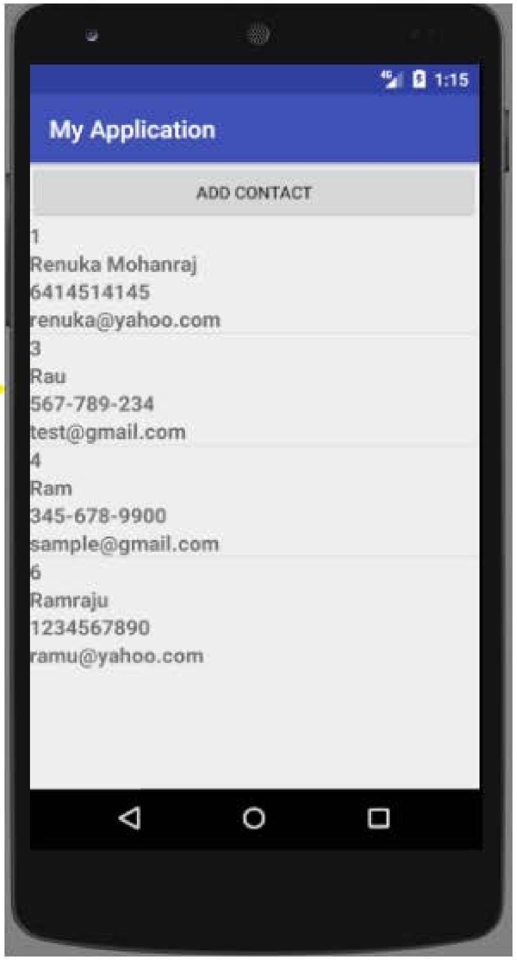
John David

654-789-567

john@yahoo.com

UPDATE DELETE

Modify_member.java



1:15

My Application

ADD CONTACT

1
Renuka Mohanraj
6414514145
renuka@yahoo.com

3
Rau
567-789-234
test@gmail.com

4
Ram
345-678-9900
sample@gmail.com

6
Ramraju
1234567890
ramu@yahoo.com

MainActivity.java

Database and table structure

Database name : Contacts, Table name : test, below is the table structure

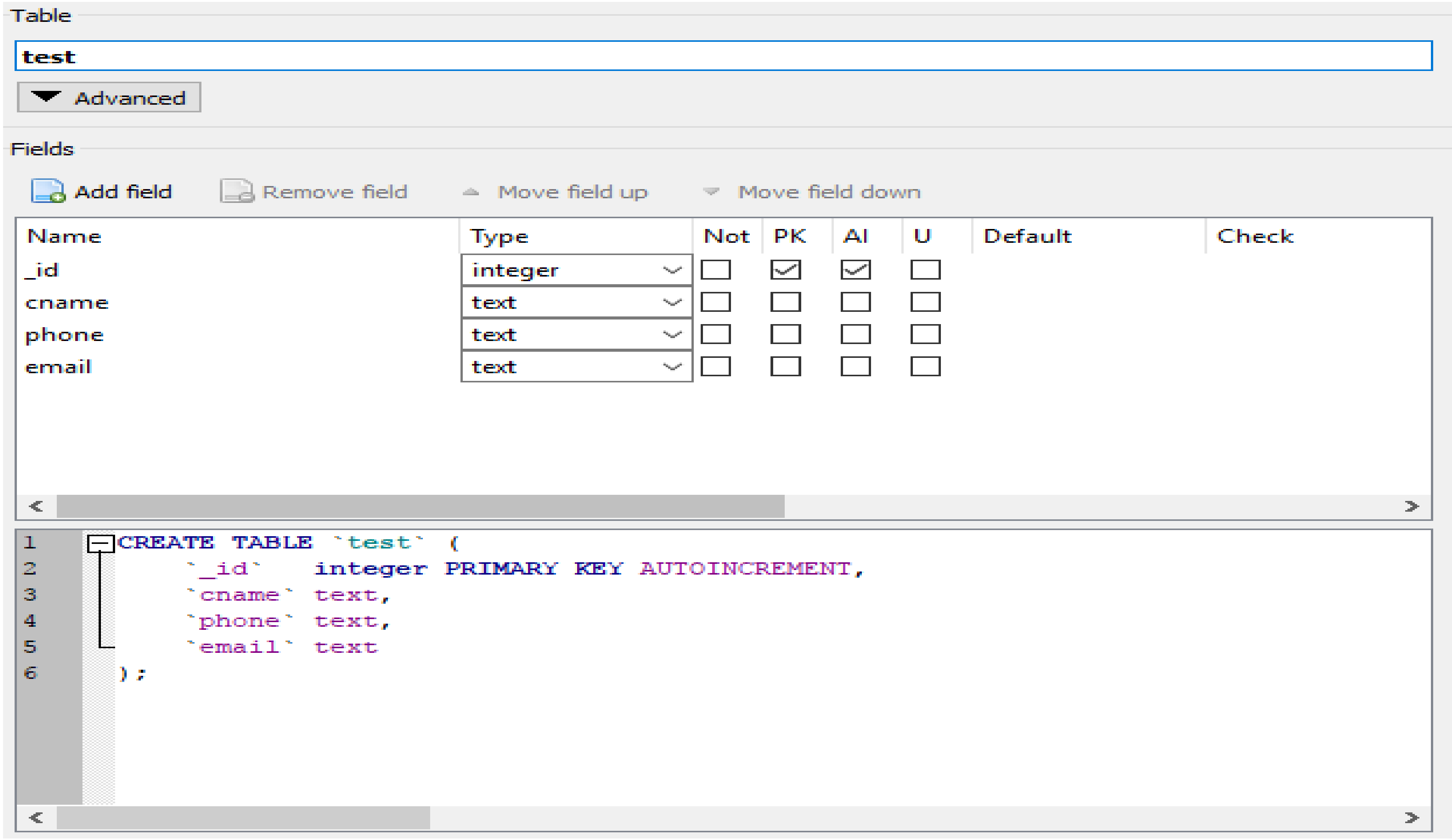


Table: test

Name	Type	Not	PK	AI	U	Default	Check
_id	integer	☐	✓	✓	☐		
cname	text	☐	☐	☐	☐		
phone	text	☐	☐	☐	☐		
email	text	☐	☐	☐	☐		

```
CREATE TABLE `test` (
	`_id`	integer PRIMARY KEY AUTOINCREMENT,
	`cname`	text,
	`phone`	text,
	`email`	text
);
```

SQL Operation	Activity Layout File	Java File
Select	activity_main.xml view_member_entry.xml	MainActivity.java ListView Component Layout
Insert	add_member.xml	Add_member.java
Update and Delete	modify_member.xml	Modify_member.java
Database and Table Creation		DBHelper.java
CRUD operations methods		SQLController.java

Simple Cursor Adapter

- An easy adapter to map columns from a cursor to TextViews or ImageViews defined in an XML file. You can specify which columns you want, which views you want to display the columns, and the XML file that defines the appearance of these views.
- Example:

```
        lv = (ListView) findViewById(R.id.lv1);
        Cursor cursor = dbcon.readData();
// Columns
String[] from = new
String[]{DBHelper.KEY_ROWID,DBHelper.NAME,DBHelper.PHONE,DBHelper.EMAIL};

// Matching ids from view for the specified columns
int[] to = new int[] { R.id.cid,R.id.vname, R.id.vmobile,R.id.vemail };

SimpleCursorAdapter adapter = new SimpleCursorAdapter(
      MainActivity.this, R.layout.view_member_entry, cursor, from, to);
  /*Notifies the attached observers that the underlying data has been
        changed and any View reflecting the data set should refresh itself.  */
   adapter.notifyDataSetChanged();
   lv.setAdapter(adapter);
```

How to view the table using SQLite browser

Step 1 : Install SQLite browser according to your OS.

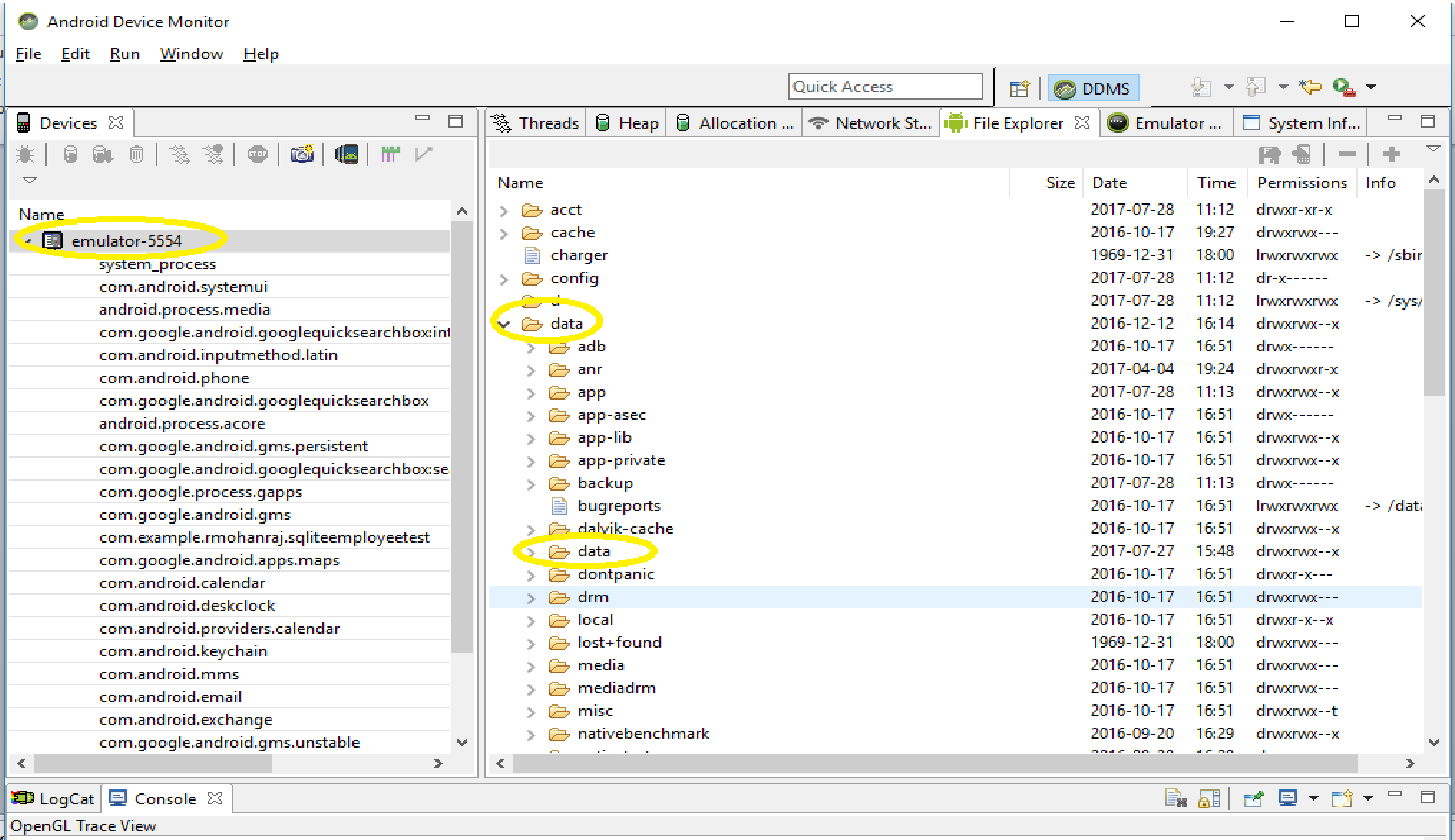
Step 2 : Keep the code in Running mode.

Step 3 : Open Android Device Monitor by clicking

Tools→Android→Android Device Monitor

How to view the table using SQLite browser

Step 4 : Choose your device on the left side, then on right side click data→ data and then choose the correct package

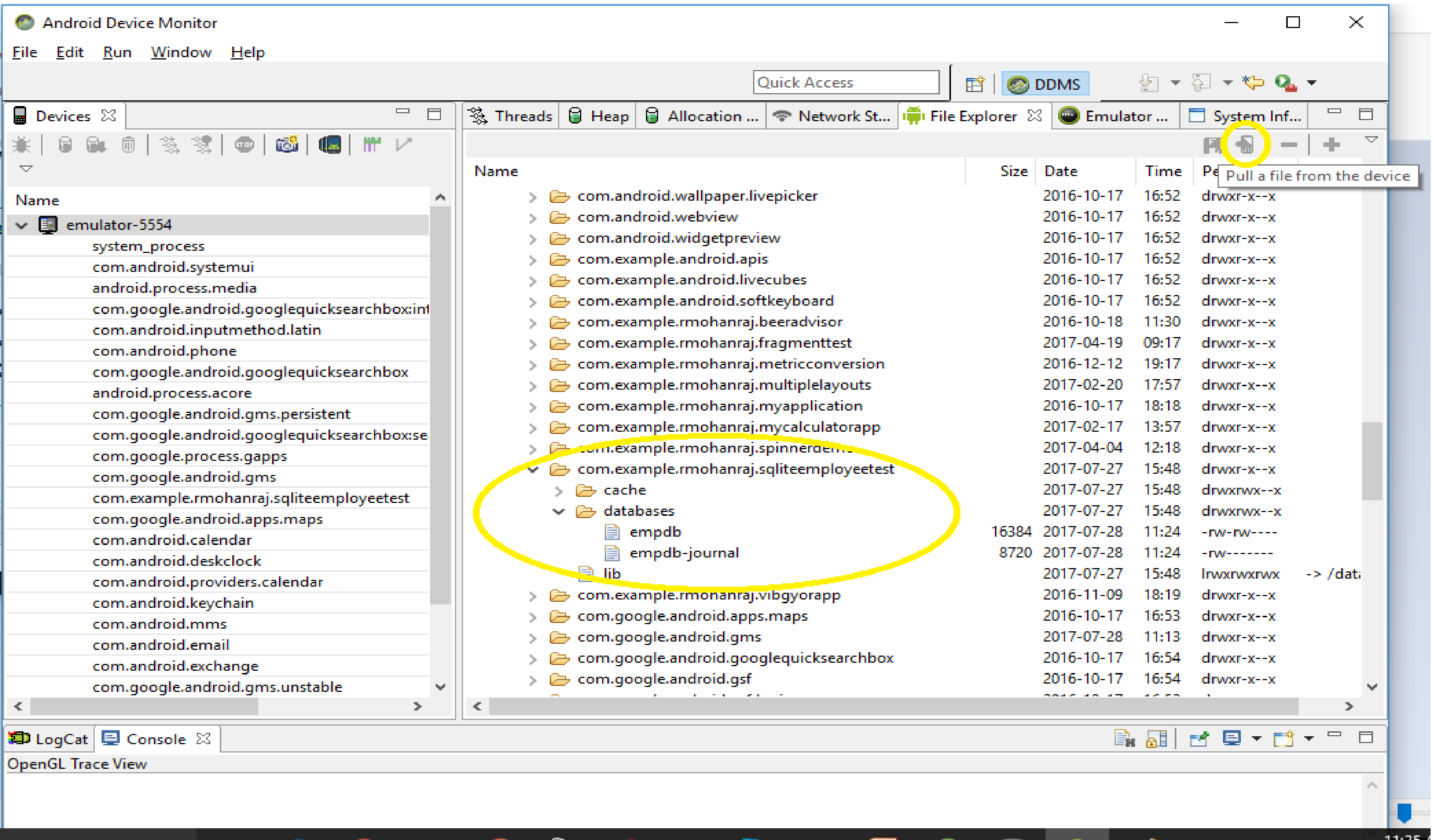


Name
- emulator-5554
  - system_process
  - com.android.systemui
  - android.process.media
  - com.google.android.googlequicksearchbox:int
  - com.android.inputmethod.latin
  - com.android.phone
  - com.google.android.googlequicksearchbox
  - android.process.acore
  - com.google.android.gms.persistent
  - com.google.android.googlequicksearchbox:se
  - com.google.process.gapps
  - com.google.android.gms
  - com.example.rmohanraj.sqliteemployeetest
  - com.google.android.apps.maps
  - com.android.calendar
  - com.android.deskclock
  - com.android.providers.calendar
  - com.android.keychain
  - com.android.mms
  - com.android.email
  - com.android.exchange
  - com.google.android.gms.unstable

Name	Size	Date	Time	Permissions	Info
acct		2017-07-28	11:12	drwxr-xr-x	
cache		2016-10-17	19:27	drwxrwx---	
charger		1969-12-31	18:00	lrwxrwxrwx	-> /sbi
config		2017-07-28	11:12	dr-x------	
d		2017-07-28	11:12	lrwxrwxrwx	-> /sys/
data		2016-12-12	16:14	drwxrwx--x	
adb		2016-10-17	16:51	drwx------	
anr		2017-04-04	19:24	drwxrwxr-x	
app		2017-07-28	11:13	drwxrwx--x	
app-asec		2016-10-17	16:51	drwx------	
app-lib		2016-10-17	16:51	drwxrwx--x	
app-private		2016-10-17	16:51	drwxrwx--x	
backup		2017-07-28	11:13	drwx------	
bugreports		2016-10-17	16:51	lrwxrwxrwx	-> /dat
dalvik-cache		2016-10-17	16:51	drwxrwx--x	
data		2017-07-27	15:48	drwxrwx--x	
dontpanic		2016-10-17	16:51	drwxr-x---	
drm		2016-10-17	16:51	drwxrwx---	
local		2016-10-17	16:51	drwxr-x--x	
lost+found		1969-12-31	18:00	drwxrwx---	
media		2016-10-17	16:51	drwxrwx---	
mediadrm		2016-10-17	16:51	drwxrwx---	
misc		2016-10-17	16:51	drwxrwx--t	
nativebenchmark		2016-09-20	16:29	drwxrwx--x	

How to view the table using SQLite browser

Step 5 : Click databases from the data in the concerned package to view your database then select the empdb and click pull a file from device icon to store in your local drive.

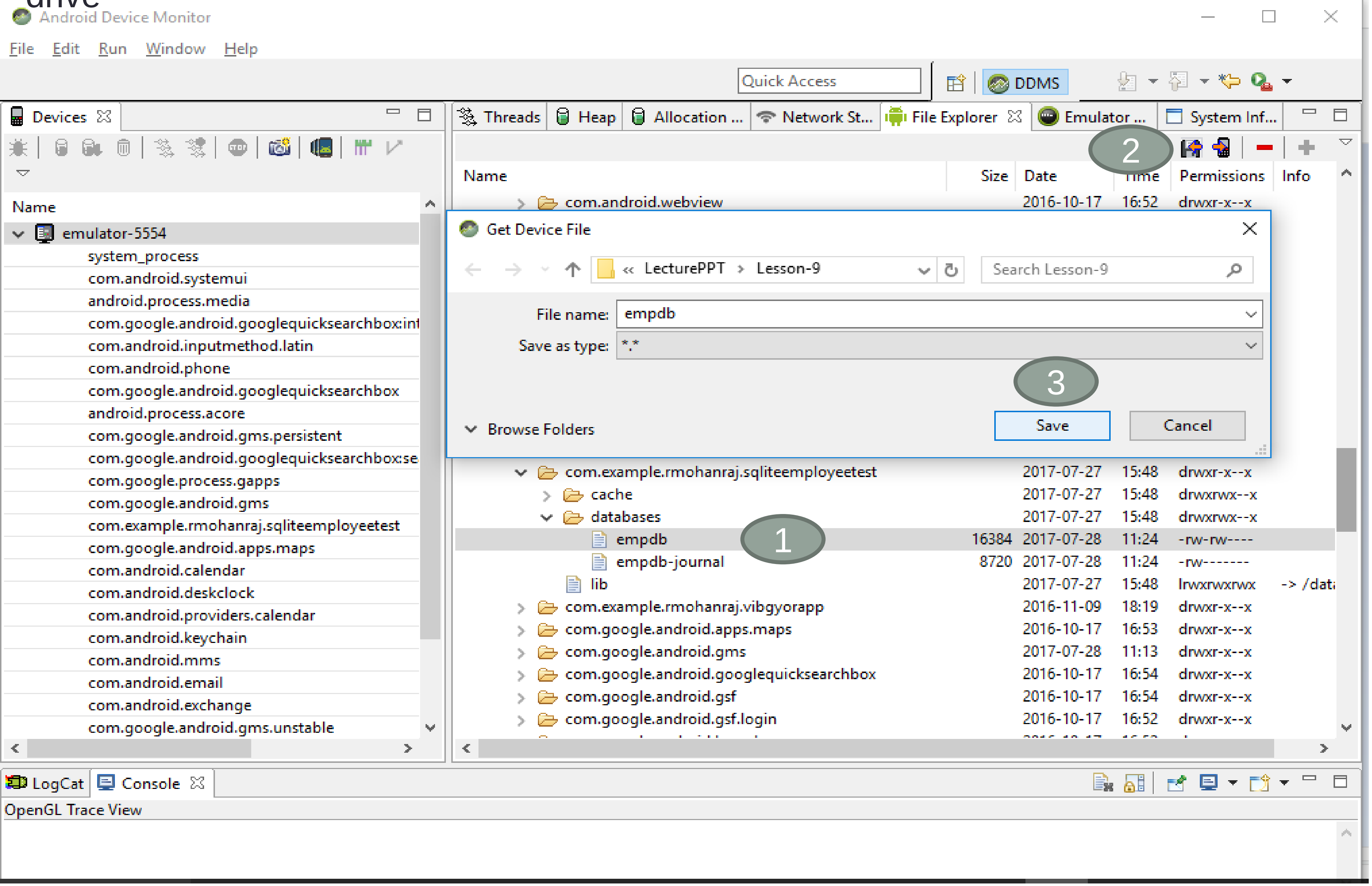


Name
emulator-5554
system_process
com.android.systemui
android.process.media
com.google.android.googlequicksearchbox:int
com.android.inputmethod.latin
com.android.phone
com.google.android.googlequicksearchbox
android.process.acore
com.google.android.gms.persistent
com.google.android.googlequicksearchbox:se
com.google.process.gapps
com.google.android.gms
com.example.rmohanraj.sqliteemployeetest
com.google.android.apps.maps
com.android.calendar
com.android.deskclock
com.android.providers.calendar
com.android.keychain
com.android.mms
com.android.email
com.android.exchange
com.google.android.gms.unstable

Name	Size	Date	Time	Pe	
com.android.wallpaper.livepicker		2016-10-17	16:52	drwxr-x--x	
com.android.webview		2016-10-17	16:52	drwxr-x--x	
com.android.widgetpreview		2016-10-17	16:52	drwxr-x--x	
com.example.android.apis		2016-10-17	16:52	drwxr-x--x	
com.example.android.livecubes		2016-10-17	16:52	drwxr-x--x	
com.example.android.softkeyboard		2016-10-17	16:52	drwxr-x--x	
com.example.rmohanraj.beeradvisor		2016-10-18	11:30	drwxr-x--x	
com.example.rmohanraj.fragmenttest		2017-04-19	09:17	drwxr-x--x	
com.example.rmohanraj.metricconversion		2016-12-12	19:17	drwxr-x--x	
com.example.rmohanraj.multiplelayouts		2017-02-20	17:57	drwxr-x--x	
com.example.rmohanraj.myapplication		2016-10-17	18:18	drwxr-x--x	
com.example.rmohanraj.mycalculatorapp		2017-02-17	13:57	drwxr-x--x	
com.example.rmohanraj.spinnerdem		2017-04-04	12:18	drwxr-x--x	
com.example.rmohanraj.sqliteemployeetest		2017-07-27	15:48	drwxr-x--x	
cache		2017-07-27	15:48	drwxrwx--x	
databases		2017-07-27	15:48	drwxrwx--x	
empdb	16384	2017-07-28	11:24	-rw-rw----	
empdb-journal	8720	2017-07-28	11:24	-rw-------	
lib		2017-07-27	15:48	lrwxrwxrwx	-> /dat
com.example.rmohanraj.vibgyorapp		2016-11-09	18:19	drwxr-x--x	
com.google.android.apps.maps		2016-10-17	16:53	drwxr-x--x	
com.google.android.gms		2017-07-28	11:13	drwxr-x--x	
com.google.android.googlequicksearchbox		2016-10-17	16:54	drwxr-x--x	
com.google.android.gsf		2016-10-17	16:54	drwxr-x--x	

Step 5 : Screen shots to save the database

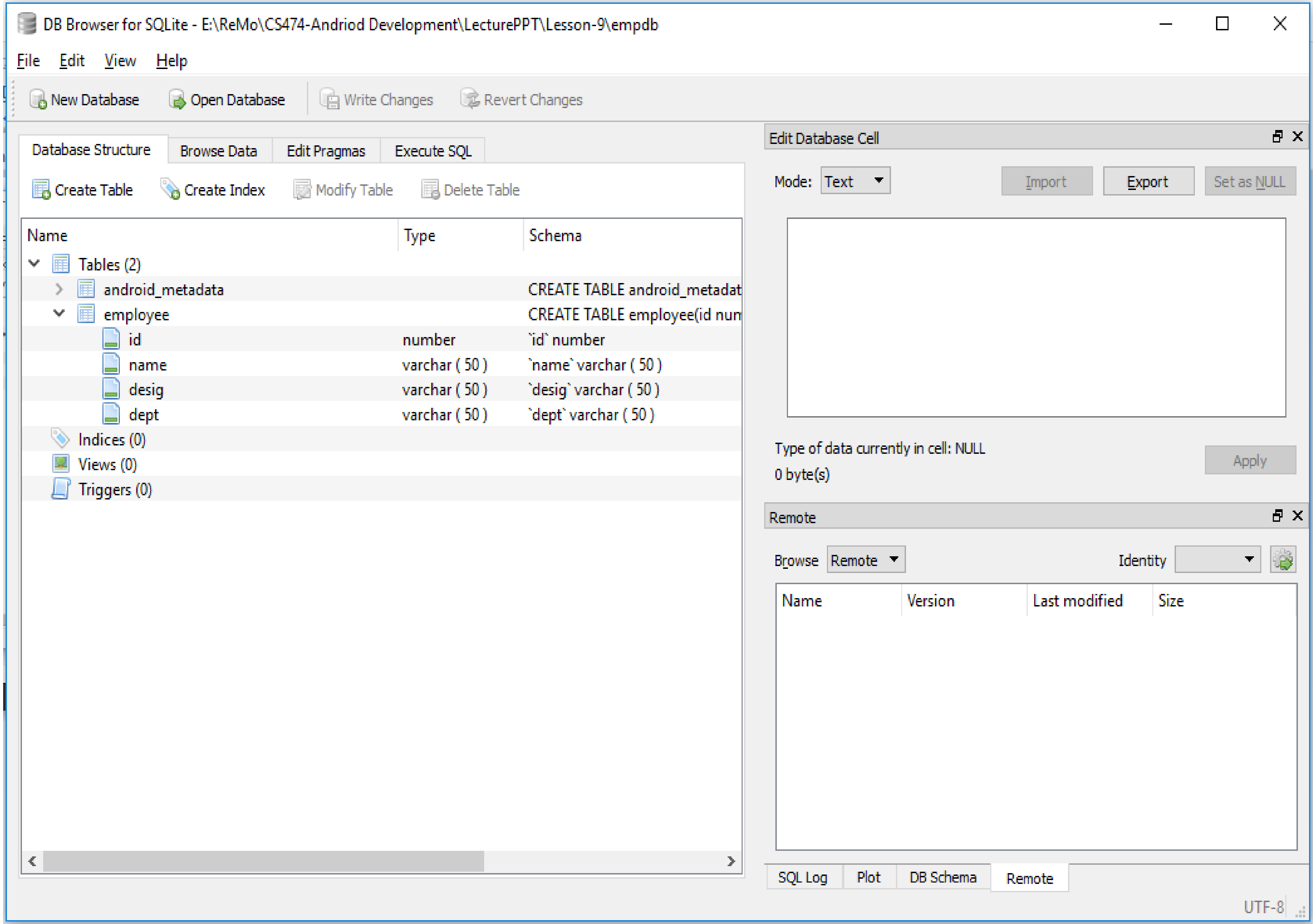
1 → Click your table ; 2→ Click pull a file from the device ; 3→ Save the File in your drive



Name
emulator-5554
system_process
com.android.systemui
android.process.media
com.google.android.googlequicksearchbox:int
com.android.inputmethod.latin
com.android.phone
com.google.android.googlequicksearchbox
android.process.acore
com.google.android.gms.persistent
com.google.android.googlequicksearchbox:se
com.google.process.gapps
com.google.android.gms
com.example.rmohanraj.sqliteemployeetest
com.google.android.apps.maps
com.android.calendar
com.android.deskclock
com.android.providers.calendar
com.android.keychain
com.android.mms
com.android.email
com.android.exchange
com.google.android.gms.unstable

Name	Size	Date	Time	Permissions	Info
com.android.webview		2016-10-17	16:52	drwxr-x--x	
com.example.rmohanraj.sqliteemployeetest		2017-07-27	15:48	drwxr-x--x	
cache		2017-07-27	15:48	drwxrwx--x	
databases		2017-07-27	15:48	drwxrwx--x	
empdb	16384	2017-07-28	11:24	-rw-rw----	
empdb-journal	8720	2017-07-28	11:24	-rw-------	
lib		2017-07-27	15:48	lrwxrwxrwx	-> /dat
com.example.rmohanraj.vibgyorapp		2016-11-09	18:19	drwxr-x--x	
com.google.android.apps.maps		2016-10-17	16:53	drwxr-x--x	
com.google.android.gms		2017-07-28	11:13	drwxr-x--x	
com.google.android.googlequicksearchbox		2016-10-17	16:54	drwxr-x--x	
com.google.android.gsf		2016-10-17	16:54	drwxr-x--x	
com.google.android.gsf.login		2016-10-17	16:52	drwxr-x--x	

Step 6 : Open your saved database file using SQLite Browser.

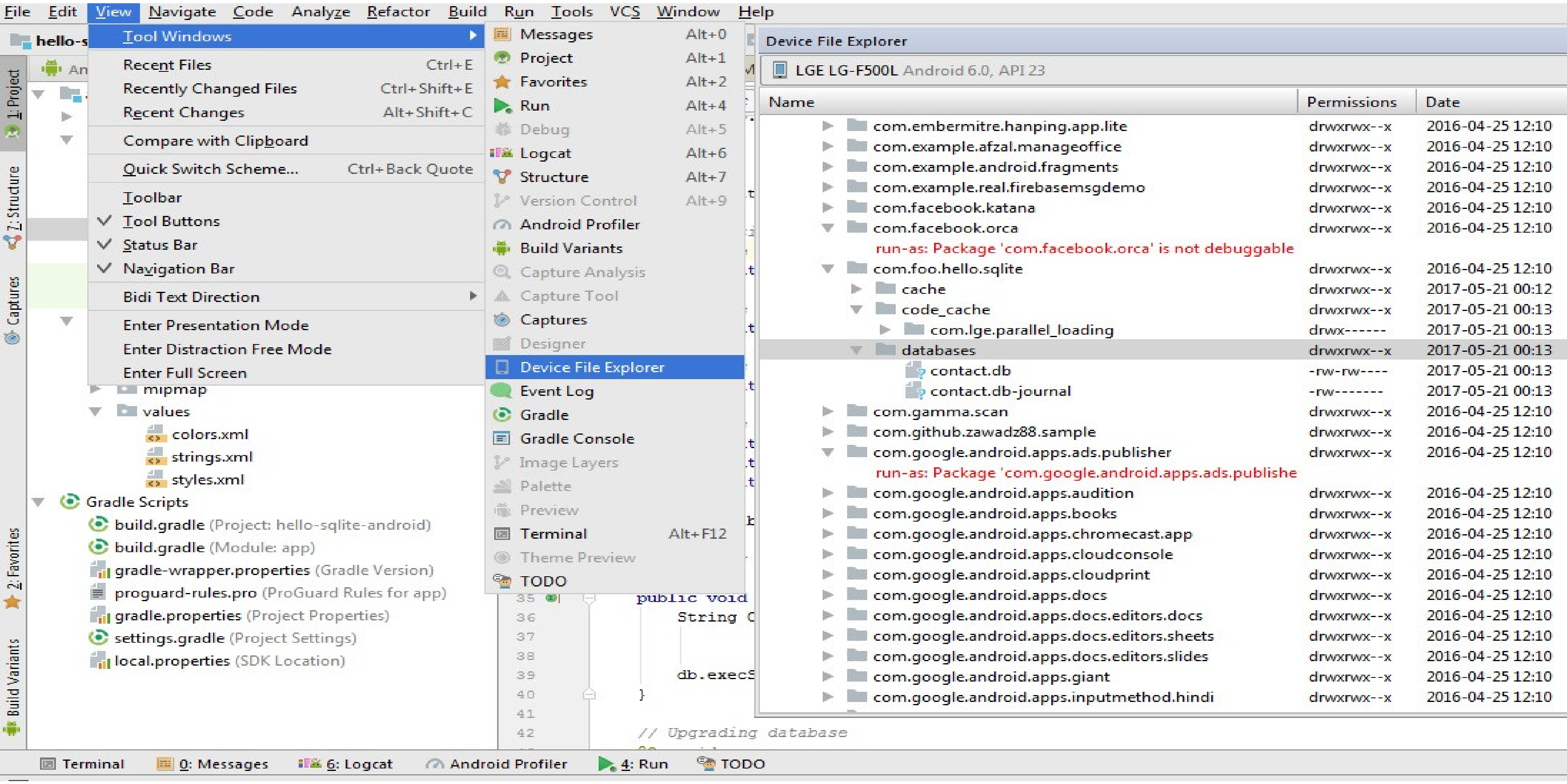

DB Browser for SQLite - E:\ReMo\CS474-Andriod Development\LecturePPT\Lesson-9\empdb

Name	Type	Schema
Tables (2)		
android_metadata		CREATE TABLE android_metadat
employee		CREATE TABLE employee(id nun
id	number	`id` number
name	varchar (50)	`name` varchar (50)
desig	varchar (50)	`desig` varchar (50)
dept	varchar (50)	`dept` varchar (50)
Indices (0)		
Views (0)		
Triggers (0)		

Type of data currently in cell: NULL
0 byte(s)

Name	Version	Last modified	Size

- If you are using Android Studio 3.0 or later version then follow these steps.
- Click **View** > **Tool Windows** > **Device File Explorer**.
- Expand **/data/data/[package-name]** nodes.
- You can only expand packages which runs in debug mode for non rooted device.


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Tool Windows
Recent Files Ctrl+E
Recently Changed Files Ctrl+Shift+E
Recent Changes Alt+Shift+C
Compare with Clipboard
Quick Switch Scheme... Ctrl+Back Quote
Toolbar
Tool Buttons
Status Bar
Navigation Bar
Bidi Text Direction
Enter Presentation Mode
Enter Distraction Free Mode
Enter Full Screen
Messages Alt+0
Project Alt+1
Favorites Alt+2
Run Alt+4
Debug Alt+5
Logcat Alt+6
Structure Alt+7
Version Control Alt+9
Android Profiler
Build Variants
Capture Analysis
Capture Tool
Captures
Designer
Device File Explorer
Event Log
Gradle
Gradle Console
Image Layers
Palette
Preview
Terminal Alt+F12
Theme Preview
TODO
mipmap
values
colors.xml
strings.xml
styles.xml
Gradle Scripts
build.gradle (Project: hello-sqlite-android)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
Terminal 0: Messages 6: Logcat Android Profiler 4: Run TODO

```
public void
    String C

    db.execS
}

// Upgrading database
```

Device File Explorer

LGE LG-F500L Android 6.0, API 23

Name	Permissions	Date
com.embermitre.hanping.app.lite	drwxrwx--x	2016-04-25 12:10
com.example.afzal.manageoffice	drwxrwx--x	2016-04-25 12:10
com.example.android.fragments	drwxrwx--x	2016-04-25 12:10
com.example.real.firebasemsgdemo	drwxrwx--x	2016-04-25 12:10
com.facebook.katana	drwxrwx--x	2016-04-25 12:10
com.facebook.orca	drwxrwx--x	2016-04-25 12:10
run-as: Package 'com.facebook.orca' is not debuggable		
com.foo.hello.sqlite	drwxrwx--x	2016-04-25 12:10
cache	drwxrwx--x	2017-05-21 00:12
code_cache	drwxrwx--x	2017-05-21 00:13
com.lge.parallel_loading	drwx------	2017-05-21 00:13
databases	drwxrwx--x	2017-05-21 00:13
contact.db	-rw-rw----	2017-05-21 00:13
contact.db-journal	-rw-------	2017-05-21 00:13
com.gamma.scan	drwxrwx--x	2016-04-25 12:10
com.github.zawadz88.sample	drwxrwx--x	2016-04-25 12:10
com.google.android.apps.ads.publisher	drwxrwx--x	2016-04-25 12:10
run-as: Package 'com.google.android.apps.ads.publishe		
com.google.android.apps.audition	drwxrwx--x	2016-04-25 12:10
com.google.android.apps.books	drwxrwx--x	2016-04-25 12:10
com.google.android.apps.chromecast.app	drwxrwx--x	2016-04-25 12:10
com.google.android.apps.cloudconsole	drwxrwx--x	2016-04-25 12:10
com.google.android.apps.cloudprint	drwxrwx--x	2016-04-25 12:10
com.google.android.apps.docs	drwxrwx--x	2016-04-25 12:10
com.google.android.apps.docs.editors.docs	drwxrwx--x	2016-04-25 12:10
com.google.android.apps.docs.editors.sheets	drwxrwx--x	2016-04-25 12:10
com.google.android.apps.docs.editors.slides	drwxrwx--x	2016-04-25 12:10
com.google.android.apps.giant	drwxrwx--x	2016-04-25 12:10
com.google.android.apps.inputmethod.hindi	drwxrwx--x	2016-04-25 12:10