**2.0 CREATE A RECORD IN ANDROID SQLITE DATABASE**

This will answer your question about how to create a record in Android SQLite database.

**2.1 Place a “Create Student” button**

2.1.1 Put a “Create Student” button on your res/layout/activity\_main.xml

2.1.2 Remove the “Hello World!” TextView there

2.1.3 Make button text to “Create Student”

2.1.4 Make button id value to “@+id/buttonCreateStudent”

Code should look like the following.

|  |
| --- |
| <Button      android:id="@+id/buttonCreateStudent"      android:layout\_width="wrap\_content"      android:layout\_height="wrap\_content"      android:layout\_alignParentLeft="true"      android:layout\_alignParentTop="true"  android:onClick:”CreateRecord”      android:text="Create Student" /> |
|  |

**2.2 Put an OnClickListener for the** button

2.2.1 We will set up the OnClickr of the “Create Student” button.

2.2.2 We can identify the button by its ID “buttonCreateStudent”

|  |
| --- |
| @Override      public void CreateRecord(View view) {        }  } |

**2.4 Prepare your student input form**

2.4.1 Right click on your res/layout/ directory > Click “New” > Click “File” > Name it “student\_input\_form.xml”

2.4.2 Put the following code inside **student\_input\_form.xml**

|  |
| --- |
| <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"      xmlns:tools="http://schemas.android.com/tools"      android:layout\_width="match\_parent"      android:layout\_height="match\_parent" >        <EditText          android:id="@+id/editTextStudentFirstname"          android:layout\_width="match\_parent"          android:layout\_height="wrap\_content"          android:layout\_alignParentLeft="true"          android:layout\_alignParentTop="true"          android:hint="Student Firstname"          android:singleLine="true" >            <requestFocus />      </EditText>        <EditText          android:id="@+id/editTextStudentEmail"          android:layout\_width="match\_parent"          android:layout\_height="wrap\_content"          android:layout\_alignParentLeft="true"          android:layout\_below="@+id/editTextStudentFirstname"          android:hint="Student Email"          android:singleLine="true" />    </RelativeLayout> |

**2.5 Show the “create form” to user**

2.5.1 Go back and open your “OnClickListenerCreateStudent.java”

2.5.2 Get the application context, this is needed to inflate an XML layout file. Put the following code inside the **onClick()** method.

|  |
| --- |
| Context context = view.getRootView().getContext(); |

2.5.3 Inflate the student\_input\_form.xml, this will make UI elements or widgets accessible using code. Put the following code after the code in 2.5.2

|  |
| --- |
| LayoutInflater inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);  final View formElementsView = inflater.inflate(R.layout.student\_input\_form, null, false); |

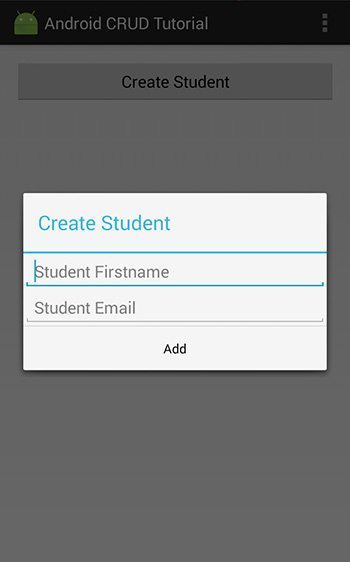
2.5.4 List down form widgets inside student\_input\_form.xml as “final” variables. This is because we will use them inside an AlertDialog. Put the following code after the code in 2.5.3

|  |
| --- |
| final EditText editTextStudentFirstname = (EditText) formElementsView.findViewById(R.id.editTextStudentFirstname);  final EditText editTextStudentEmail = (EditText) formElementsView.findViewById(R.id.editTextStudentEmail); |

2.5.5 Create an AlertDialog with the inflated student\_input\_form.xml and an “Add” button. Put the following code after the code in 2.5.4

|  |
| --- |
| new AlertDialog.Builder(context)      .setView(formElementsView)      .setTitle("Create Student")      .setPositiveButton("Add",          new DialogInterface.OnClickListener() {              public void onClick(DialogInterface dialog, int id) {                    dialog.cancel();              }            }).show(); |

2.5.5 Try to run. The following image should be the output when you clicked the “Create Student” button.

[](https://i1.wp.com/www.androidcode.ninja/wp-content/uploads/2013/02/android-crud-tutorial-create.jpg)

**2.6 Save user input**

2.6.1 Inside the AlertDialog’s “Add” onClick() method, we will save the record.

2.6.2 Get the user inputted values using the following code. Put them inside the AlertDialog’s “Add” onClick() method

|  |
| --- |
| String studentFirstname = editTextStudentFirstname.getText().toString();  String studentEmail = editTextStudentEmail.getText().toString(); |

2.6.3 We’re actually using an OOP approach here. Create a new “ObjectStudent.java” file with properties “firstname” and “email”. Put the following code inside ObjectStudent.java file.

|  |
| --- |
| public class ObjectStudent {        int id;      String firstname;      String email;        public ObjectStudent(){        }  } |

2.6.4 Going back to AlertDialog’s “Add” onClick() method, set the input values as a object, so we can save it to the database.

|  |
| --- |
| ObjectStudent objectStudent = new ObjectStudent();  objectStudent.firstname= studentFirstname;  objectStudent.email= studentEmail; |

2.6.5 Create a new “DatabaseHandler.java” file. This file will handle all the SQLite database connection. It will have the following code.

|  |
| --- |
| public class DatabaseHandler extends SQLiteOpenHelper {        private static final int DATABASE\_VERSION = 1;      protected static final String DATABASE\_NAME = "StudentDatabase";        public DatabaseHandler(Context context) {          super(context, DATABASE\_NAME, null, DATABASE\_VERSION);      }        @Override      public void onCreate(SQLiteDatabase db) {            String sql = "CREATE TABLE students " +                  "( id INTEGER PRIMARY KEY AUTOINCREMENT, " +                  "firstname TEXT, " +                  "email TEXT ) ";            db.execSQL(sql);        }        @Override      public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {            String sql = "DROP TABLE IF EXISTS students";          db.execSQL(sql);            onCreate(db);      }    } |

2.6.6 Create new “TableControllerStudent.java” file, extending the DatabaseHandler. This file will control all the operations related to the student’s table. It will have the following code.

|  |
| --- |
| public class TableControllerStudent extends DatabaseHandler {        public TableControllerStudent(Context context) {          super(context);      }    } |

2.6.7 Do the create() method for creating new record. The following method is inside TableControllerStudent class.

|  |
| --- |
| public boolean create(ObjectStudent objectStudent) {        ContentValues values = new ContentValues();        values.put("firstname", objectStudent.firstname);      values.put("email", objectStudent.email);        SQLiteDatabase db = this.getWritableDatabase();        boolean createSuccessful = db.insert("students", null, values) > 0;      db.close();        return createSuccessful;  } |

2.6.8 Go back inside the AlertDialog of OnClickListenerCreateStudent.java file and call the create() method of TableControllerStudent class.

|  |
| --- |
| boolean createSuccessful = new TableControllerStudent(context).create(objectStudent); |

2.6.9 Tell the user whether insert was a success or failure. We will use Android Toast to do this task.

|  |
| --- |
| if(createSuccessful){      Toast.makeText(context, "Student information was saved.", Toast.LENGTH\_SHORT).show();  }else{      Toast.makeText(context, "Unable to save student information.", Toast.LENGTH\_SHORT).show();  } |

2.6.10 Try to run. You should be able to see a toast message and create a record.

**3.0 COUNT RECORDS FROM ANDROID SQLITE DATABASE**

3.1 On your res/layout/activity\_main.xml, place a TextView under your “Create Student” button.

|  |
| --- |
| <TextView      android:id="@+id/textViewRecordCount"      android:gravity="center"      android:layout\_width="fill\_parent"      android:layout\_height="wrap\_content"      android:layout\_below="@+id/buttonCreateStudent"      android:text="0 records found"      android:padding="1dp" /> |

3.2 On your MainActivity.java file, create a countRecords() method

|  |
| --- |
| public void countRecords() {    } |

3.3 On your TableControllerStudent.java, create a count() method.

|  |
| --- |
| public int count() {        SQLiteDatabase db = this.getWritableDatabase();        String sql = "SELECT \* FROM students";      int recordCount = db.rawQuery(sql, null).getCount();      db.close();        return recordCount;    } |

3.4 Go back to your MainActivity.java > countRecords() method and call the count() method you just created.

|  |
| --- |
| int recordCount = new TableControllerStudent(this).count(); |

3.5 Display the count to the text view.

|  |
| --- |
| TextView textViewRecordCount = (TextView) findViewById(R.id.textViewRecordCount);  textViewRecordCount.setText(recordCount + " records found."); |

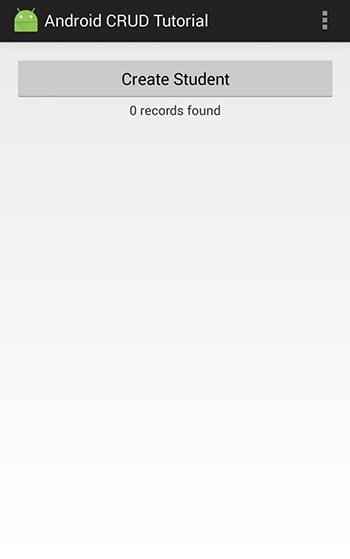
3.6 On your MainActivity.java > onCreate() method, call the countRecords() method.

|  |
| --- |
| countRecords(); |

3.7 On your OnClickListenerCreateStudent.java > inside AlertDialog, also call the countRecords() method. This will change the count value everytime a new record was created.

|  |
| --- |
| countRecords(); |

3.8 Try to run. Output should look like the following.

[](https://i2.wp.com/www.androidcode.ninja/wp-content/uploads/2013/02/android-crud-tutorial-count-records.jpg)

**4.0 READ RECORDS FROM ANDROID SQLITE DATABASE**

4.1 On your activity\_main.xml, put a ScrollView with LinearLayout inside. Put the following code under the TextView textViewRecordCount.

|  |
| --- |
| <ScrollView      android:id="@+id/scrollViewRecords"      android:layout\_width="match\_parent"      android:layout\_height="wrap\_content"      android:layout\_below="@+id/textViewRecordCount" >        <LinearLayout          android:id="@+id/linearLayoutRecords"          android:layout\_width="match\_parent"          android:layout\_height="wrap\_content"          android:orientation="vertical" >        </LinearLayout>  </ScrollView> |

4.2 On your TableControllerStudent.java, create a read() method.

|  |
| --- |
| public List<ObjectStudent> read() {        List<ObjectStudent> recordsList = new ArrayList<ObjectStudent>();        String sql = "SELECT \* FROM students ORDER BY id DESC";        SQLiteDatabase db = this.getWritableDatabase();      Cursor cursor = db.rawQuery(sql, null);        if (cursor.moveToFirst()) {          do {                int id = Integer.parseInt(cursor.getString(cursor.getColumnIndex("id")));              String studentFirstname = cursor.getString(cursor.getColumnIndex("firstname"));              String studentEmail = cursor.getString(cursor.getColumnIndex("email"));                ObjectStudent objectStudent = new ObjectStudent();              objectStudent.id = id;              objectStudent.firstname = studentFirstname;              objectStudent.email = studentEmail;                recordsList.add(objectStudent);            } while (cursor.moveToNext());      }        cursor.close();      db.close();        return recordsList;  } |

4.3 On your MainActivity.java, create the readRecords() method. This will display database records to user interface.

|  |
| --- |
| public void readRecords() {        LinearLayout linearLayoutRecords = (LinearLayout) findViewById(R.id.linearLayoutRecords);      linearLayoutRecords.removeAllViews();        List<ObjectStudent> students = new TableControllerStudent(this).read();        if (students.size() > 0) {            for (ObjectStudent obj : students) {                int id = obj.id;              String studentFirstname = obj.firstname;              String studentEmail = obj.email;                String textViewContents = studentFirstname + " - " + studentEmail;                TextView textViewStudentItem= new TextView(this);              textViewStudentItem.setPadding(0, 10, 0, 10);              textViewStudentItem.setText(textViewContents);              textViewStudentItem.setTag(Integer.toString(id));                linearLayoutRecords.addView(textViewStudentItem);          }        }        else {            TextView locationItem = new TextView(this);          locationItem.setPadding(8, 8, 8, 8);          locationItem.setText("No records yet.");            linearLayoutRecords.addView(locationItem);      }    } |

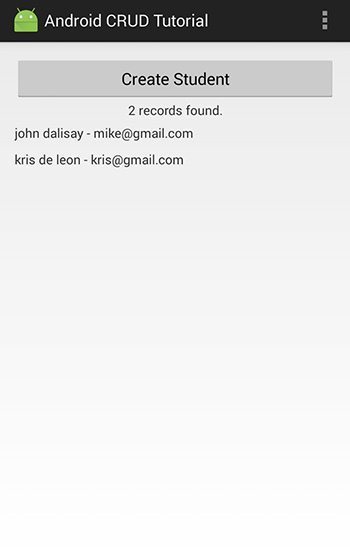
4.4 Call readRecords() method on your MainActivity.java > onCreate() method

|  |
| --- |
| readRecords(); |

4.5 On your OnClickListenerCreateStudent.java > inside AlertDialog, call the readRecords() method as well, so it will refresh the list every time we add new record.

|  |
| --- |
| ((MainActivity) context).readRecords(); |

4.6 Try to run. Output should look like the following.

[](https://i0.wp.com/www.androidcode.ninja/wp-content/uploads/2013/02/android-crud-tutorial-read-records.jpg)

**5.0 UPDATE A RECORD IN ANDROID SQLITE DATABASE**

5.1 Create new OnLongClickListenerStudentRecord.java file. We will use long click to give user an update option.

|  |
| --- |
| public class OnLongClickListenerStudentRecord implements OnLongClickListener {        @Override      public boolean onLongClick(View view) {            return false;      }    } |

5.2 Set the OnLongClickListener for each of the display records. Go to your MainActivity.java > readRecords() method, inside the ‘for’ loop, put the following code under the **textViewStudentItem.setTag(Integer.toString(id));** code.

|  |
| --- |
| textViewStudentItem.setOnLongClickListener(new OnLongClickListenerStudentRecord()); |

5.3 Go back to your OnLongClickListenerStudentRecord.java file, set the following as class variables

|  |
| --- |
| Context context;  String id; |

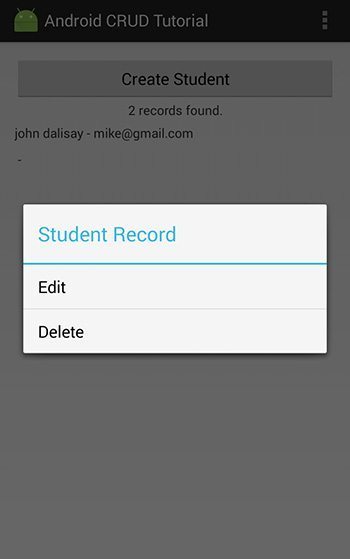
5.4 Put the following code inside the onLongClick() method.

|  |
| --- |
| context = view.getContext();  id = view.getTag().toString(); |

5.5 Add an AlertDialog with simple list view for ‘Edit’ and ‘Delete’ options. Put the following code below 5.4

|  |
| --- |
| final CharSequence[] items = { "Edit", "Delete" };    new AlertDialog.Builder(context).setTitle("Student Record")      .setItems(items, new DialogInterface.OnClickListener() {          public void onClick(DialogInterface dialog, int item) {                dialog.dismiss();            }      }).show(); |

5.6 Try to run. Output should look like the following.

[](https://i0.wp.com/www.androidcode.ninja/wp-content/uploads/2013/02/android-crud-tutorial-edit-delete.jpg)

5.7 Inside the **onClick()** method of AlertDialog in section 5.5, put the following code. “Edit” has an item index of 0.

|  |
| --- |
| if (item == 0) {      editRecord(Integer.parseInt(id));  } |

5.8 On your OnLongClickListenerStudentRecord.java, add the following editRecord() method.

|  |
| --- |
| public void editRecord(final int studentId) {    } |

5.9 Inside the editRecord() method, we will use the following code to read single record. Data will be used to fill up the student form for updating it.

|  |
| --- |
| final TableControllerStudent tableControllerStudent = new TableControllerStudent(context);  ObjectStudent objectStudent = tableControllerStudent.readSingleRecord(studentId); |

5.10 On your TableControllerStudent.java, add the following method readSingleRecord() code.

|  |
| --- |
| public ObjectStudent readSingleRecord(int studentId) {        ObjectStudent objectStudent = null;        String sql = "SELECT \* FROM students WHERE id = " + studentId;        SQLiteDatabase db = this.getWritableDatabase();        Cursor cursor = db.rawQuery(sql, null);        if (cursor.moveToFirst()) {            int id = Integer.parseInt(cursor.getString(cursor.getColumnIndex("id")));          String firstname = cursor.getString(cursor.getColumnIndex("firstname"));          String email = cursor.getString(cursor.getColumnIndex("email"));            objectStudent = new ObjectStudent();          objectStudent.id = id;          objectStudent.firstname = firstname;          objectStudent.email = email;        }        cursor.close();      db.close();        return objectStudent;    } |

5.11 Going back to OnLongClickListenerStudentRecord.java > editRecrod() method, inflate student\_input\_form.xml, this time we will use it for updating a record.

|  |
| --- |
| LayoutInflater inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);  final View formElementsView = inflater.inflate(R.layout.student\_input\_form, null, false); |

5.12 List down form elements. Put the following code under the code in section 5.11

|  |
| --- |
| final EditText editTextStudentFirstname = (EditText) formElementsView.findViewById(R.id.editTextStudentFirstname);  final EditText editTextStudentEmail = (EditText) formElementsView.findViewById(R.id.editTextStudentEmail); |

5.13 Set single record values to the EditText form elements. Put the following code under the code in section 5.12

|  |
| --- |
| editTextStudentFirstname.setText(objectStudent.firstname);  editTextStudentEmail.setText(objectStudent.email); |

5.14 Show an AlertDialog with the form and single record filling it up. Put the following code under the code in section 5.13

|  |
| --- |
| new AlertDialog.Builder(context)      .setView(formElementsView)      .setTitle("Edit Record")      .setPositiveButton("Save Changes",          new DialogInterface.OnClickListener() {              public void onClick(DialogInterface dialog, int id) {                    dialog.cancel();              }            }).show(); |

5.15 Inside the AlertDialog > onClick() method, create the object with the updated value. We are still in the editRecord() method.

|  |
| --- |
| ObjectStudent objectStudent = new ObjectStudent();  objectStudent.id = studentId;  objectStudent.firstname = editTextStudentFirstname.getText().toString();  objectStudent.email = editTextStudentEmail.getText().toString(); |

5.16 Update the record and tell the user whether it was updated or not. Put the following code under the code in section 5.15

|  |
| --- |
| boolean updateSuccessful = tableControllerStudent.update(objectStudent);    if(updateSuccessful){      Toast.makeText(context, "Student record was updated.", Toast.LENGTH\_SHORT).show();  }else{      Toast.makeText(context, "Unable to update student record.", Toast.LENGTH\_SHORT).show();  } |

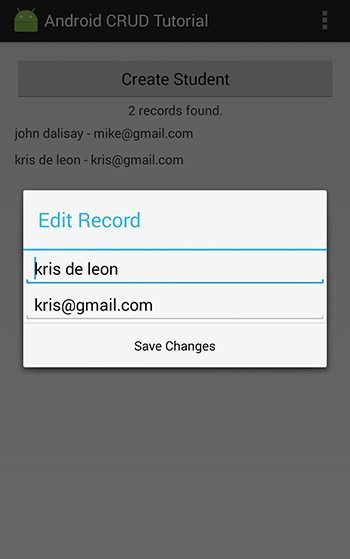
5.17 On your TableControllerStudent.java, add the update() method.

|  |
| --- |
| public boolean update(ObjectStudent objectStudent) {        ContentValues values = new ContentValues();        values.put("firstname", objectStudent.firstname);      values.put("email", objectStudent.email);        String where = "id = ?";        String[] whereArgs = { Integer.toString(objectStudent.id) };        SQLiteDatabase db = this.getWritableDatabase();        boolean updateSuccessful = db.update("students", values, where, whereArgs) > 0;      db.close();        return updateSuccessful;    } |

5.18 Refresh the count and record list. Put the following code under the code in section 5.16

|  |
| --- |
| ((MainActivity) context).countRecords();  ((MainActivity) context).readRecords(); |

5.19 Try to run. The output should look like the following.

[](https://i0.wp.com/www.androidcode.ninja/wp-content/uploads/2013/02/android-crud-tutorial-update.jpg)

**6.0 DELETE A RECORD IN ANDROID SQLITE DATABASE**

6.1 Go to your OnLongClickListenerStudentRecord.java > onLongClick() method > inside the AlertDialog onClick() method. Put the following code after the first if statement.

|  |
| --- |
| else if (item == 1) {        boolean deleteSuccessful = new TableControllerStudent(context).delete(Integer.parseInt(id));        if (deleteSuccessful){          Toast.makeText(context, "Student record was deleted.", Toast.LENGTH\_SHORT).show();      }else{          Toast.makeText(context, "Unable to delete student record.", Toast.LENGTH\_SHORT).show();      }        ((MainActivity) context).countRecords();      ((MainActivity) context).readRecords();    } |

6.2 Go back to your TableControllerStudent.java and add the following delete code.

|  |
| --- |
| public boolean delete(int id) {      boolean deleteSuccessful = false;        SQLiteDatabase db = this.getWritableDatabase();      deleteSuccessful = db.delete("students", "id ='" + id + "'", null) > 0;      db.close();        return deleteSuccessful;    } |