Android SQLite Tutorial

Introduction:

The objective of this tutorial is to demonstrate and explain how to manipulate an SQLite database in the android development environment. It involves the presentation of the database in the UI as well as the functions of CRUD (create, read, update and delete) records in the app demo. The audiences are expected to know how to manage an SQLite database and present the desired output in the android development after studying this tutorial. A simple birth weight database is used as the example.

Assumption:

The audience should know the following knowledge in prior to this tutorial:

- Java programming language,
- Xml language,
- SQL query, and
- Some basic knowledge in android development like layout, event handlers, manipulate the elements in UI, intent, context and so on.

The flow:

We will go through the following steps for this tutorial.

Step 1 – Create the layout of MainActivity with elements

Step 2 – Create a database subclass which inherits SQLiteOpenHelper class

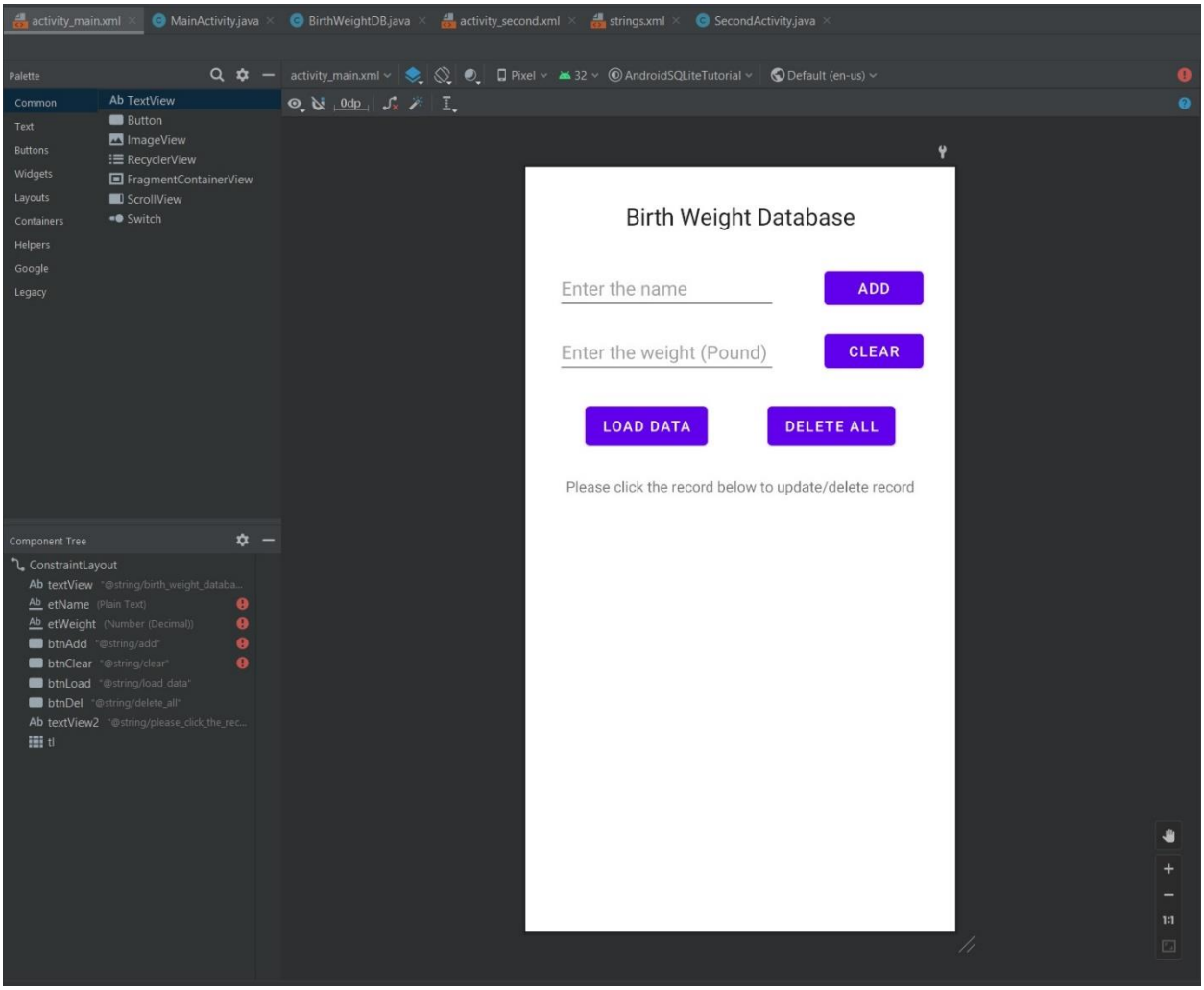
Step 3 – Declare and call the variables and methods in MainActivity.java for creating the database, adding records, and reading/displaying the records

Step 4 – Create the SecondActivity.java and its layout

Step 5 – Declare and call the variables and methods in SecondActivity.java for updating and deleting the records

Step 1 – Create the layout of MainActivity with elements

First of all, we design the layout of the MainActivity. There are several elements including 2 textviews for displaying messages, 2 edittexts for user input, 4 buttons for handling events and 1 tablelayout for displaying the records from the database. Here is the screenshot of the layout code and design.



BirthWeightDB.java | activity_main.xml | MainActivity.java | activity_second.xml | SecondActivity.java

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:text="@string/birth_weight_database"
        android:textAppearance="@style/TextAppearance.AppCompat.Large"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/etName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:layout_marginEnd="8dp"
        android:ems="10"
        android:hint="@string/enter_the_name"
        android:inputType="textPersonName"
        app:layout_constraintEnd_toStartOf="@+id/btnAdd"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView" />

    <EditText
        android:id="@+id/etWeight"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="8dp"
        android:ems="10"
        android:hint="@string/enter_the_weight_pound"
        android:inputType="numberDecimal"
        app:layout_constraintEnd_toStartOf="@+id/btnClear"
```

BirthWeightDB.java | activity_main.xml | MainActivity.java | activity_second.xml | SecondActivity.java

```
        app:layout_constraintEnd_toStartOf="@+id/btnClear"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etName" />

    <Button
        android:id="@+id/btnAdd"
        android:layout_width="95dp"
        android:layout_height="44dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="32dp"
        android:onClick="addData"
        android:text="@string/add"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/etName"
        app:layout_constraintTop_toBottomOf="@+id/textView" />

    <Button
        android:id="@+id/btnClear"
        android:layout_width="95dp"
        android:layout_height="44dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        android:onClick="clear"
        android:text="@string/clear"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/etWeight"
        app:layout_constraintTop_toBottomOf="@+id/btnAdd" />

    <Button
        android:id="@+id/btnLoad"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:onClick="loadData"
        android:text="@string/load_data"
        app:layout_constraintEnd_toStartOf="@+id/btnDel"
        app:layout_constraintHorizontal_chainStyle="spread"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etWeight" />

```

BirthWeightDB.java | activity_main.xml | MainActivity.java | activity_second.xml | SecondActivity.java

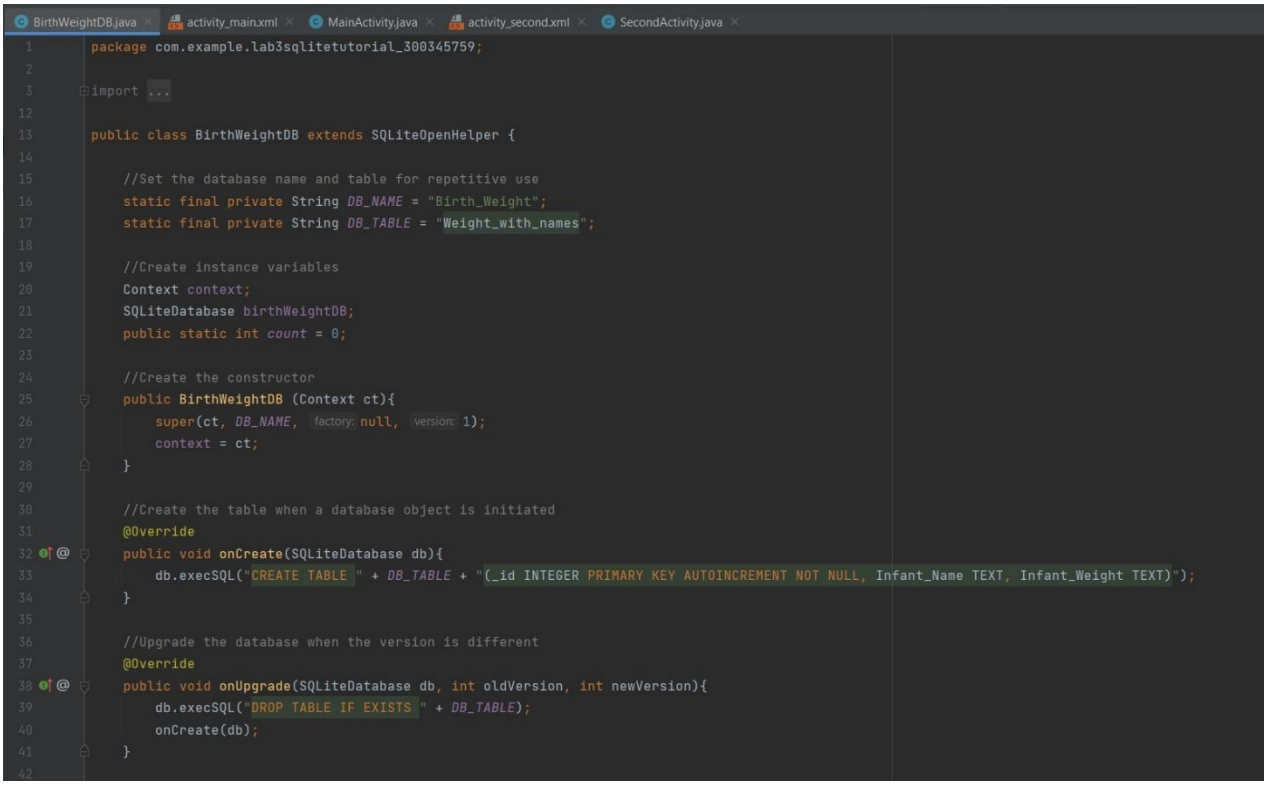
```
78          app:layout_constraintHorizontal_chainStyle="spread"
79          app:layout_constraintStart_toStartOf="parent"
80          app:layout_constraintTop_toBottomOf="@+id/etWeight" />
81
82      <Button
83          android:id="@+id/btnDel"
84          android:layout_width="wrap_content"
85          android:layout_height="wrap_content"
86          android:layout_marginTop="24dp"
87          android:onClick="delDB"
88          android:text="@string/delete_all"
89          app:layout_constraintEnd_toEndOf="parent"
90          app:layout_constraintStart_toEndOf="@+id/btnLoad"
91          app:layout_constraintTop_toBottomOf="@+id/etWeight" />
92
93      <TextView
94          android:id="@+id/textView2"
95          android:layout_width="wrap_content"
96          android:layout_height="wrap_content"
97          android:layout_marginTop="24dp"
98          android:text="@string/please_click_the_record_below_to_update_delete_record"
99          app:layout_constraintEnd_toEndOf="parent"
100         app:layout_constraintStart_toStartOf="parent"
101         app:layout_constraintTop_toBottomOf="@+id/btnLoad" />
102
103     <TableLayout
104         android:id="@+id/tl"
105         android:layout_width="match_parent"
106         android:layout_height="wrap_content"
107         android:layout_marginStart="32dp"
108         android:layout_marginTop="16dp"
109         android:layout_marginEnd="32dp"
110         app:layout_constraintEnd_toEndOf="parent"
111         app:layout_constraintHorizontal_bias="1.0"
112         app:layout_constraintStart_toStartOf="parent"
113         app:layout_constraintTop_toBottomOf="@+id/textView2" />
114
115 </androidx.constraintlayout.widget.ConstraintLayout>
```

Step 2 – Create a database subclass which inherits SQLiteOpenHelper class

A database subclass of SQLiteOpenHelper class is required to allow the app to execute database-related methods from SQLiteOpenHelper class. We name “BirthWeightDB” as an example of creating the database subclass.

After creating the class, we add several variables which will be used in the methods. They include some constant strings which represent the database table and column names, context, an int for saving the number of records, and the database variable itself, SQLiteDatabase variable.

Next, we create a constructor to initiate the database object in the MainActivity. Super is used due to the inheritance from the super class. There are two override methods which are onCreate() and onUpgrade(). We use db.execSQL() with SQL “CREATE TABLE” query in onCreate() to create the table when the database object is initiated. onUpgrade() is used when the developer wants to upgrade the database. Here is the code.



The last part of this class is creating corresponding methods to manage the database. There are five methods which are named as insertData(), loadData(), delAll(), update(), delData(). Besides loadData(), other methods use getWritableDatabase() which does not accept SQL query with "SELECT" allows us to create/update/delete the records in the database. For loadData(), getReadableDatabase(), which accepts SQL query with "SELECT", is used to read the database.

Similar to onCreate(), we use BirthWeightDB.execSQL() with SQL queries to execute the queries in the methods besides loadData(). There are some alternative methods which does not require SQL query for update(), delAll() and delData(). These methods, which are included in the code as comments, are safer for developers to prevent SQL syntax mistakes.

For loadData(), we use ArrayList<String[]> return method to return the database records into an ArrayList. We firstly create an empty ArrayList and make BirthWeightDB readable by using getReadableDatabase(). We next create a cursor, cr, onto a view/table with desired data by using rawQuery() with SQL "SELECT" query. The numeric value of number of records is assigned to the variable count by using cr.getCount(). Then we use while loop with cr.moveToNext() to screen the data from the selected view/table. By using cr.getInt() and cr.getString(), we can assign the values of each field from records to String variable and the ArrayList. At last but not least, we need to close the cursor and return the ArrayList. Here is the code of the 5 methods.

BirthWeightDB.java | activity_main.xml | MainActivity.java | activity_second.xml | SecondActivity.java

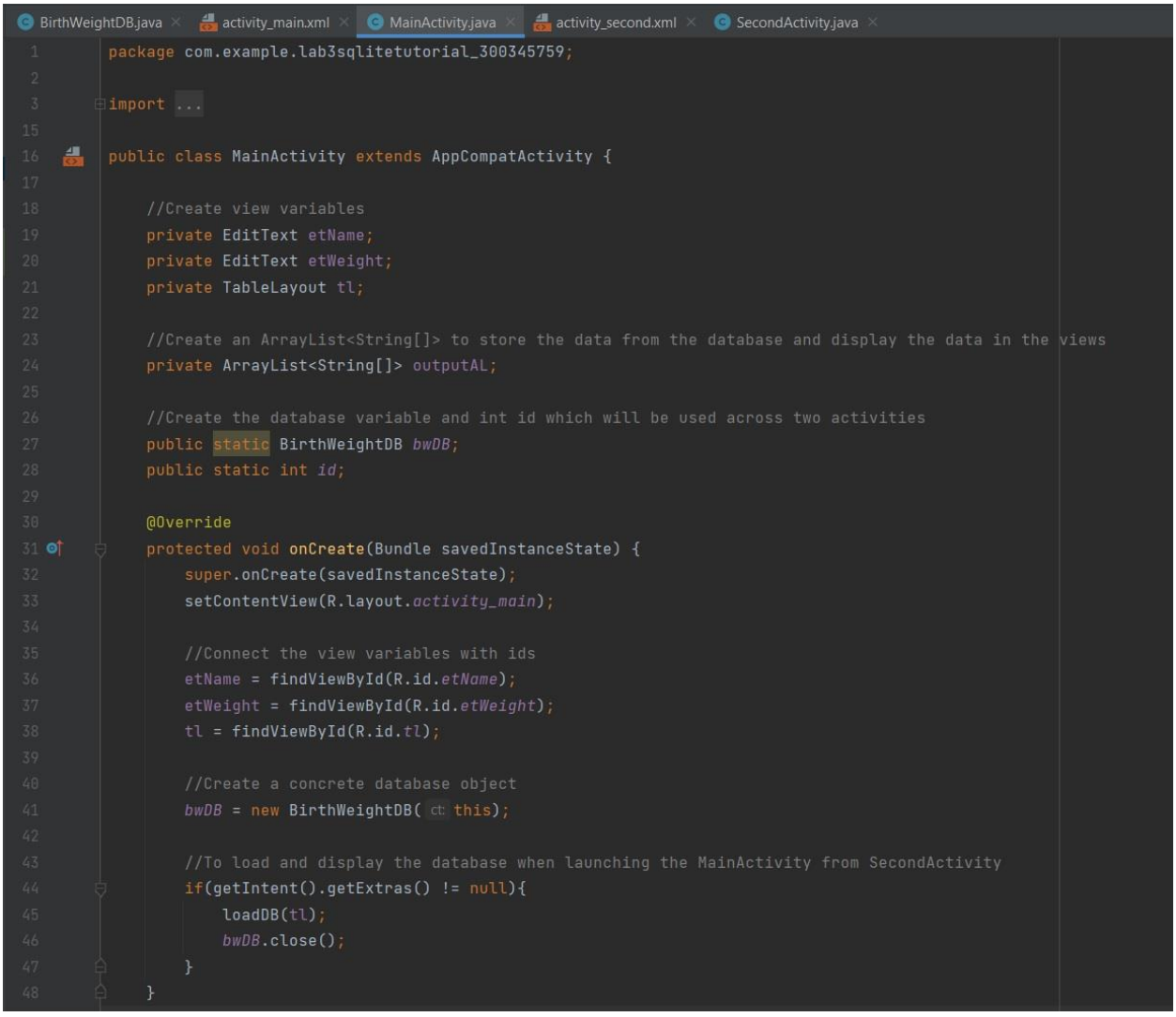
```
42
43      //Create a method to insert/add a record with the values from parameters
44      public void insertData(String name, String weight){
45          //Make the database writable
46          birthWeightDB = getWritableDatabase();
47          //Add the record into the database by executing the SQL INSERT query
48          birthWeightDB.execSQL("INSERT INTO " + DB_TABLE + " (Infant_Name, Infant_Weight) VALUES ('" + name + "','" + weight + "');");
49      }
50
51      //Create a method to return an ArrayList<String[]> with the records from the database
52      public ArrayList<String[]> loadData(){
53
54          ArrayList<String[]> record = new ArrayList<>();
55
56          //Make the database readable
57          birthWeightDB = getReadableDatabase();
58
59          //Create a cursor to read the table created by the SQL SELECT query
60          Cursor cr = birthWeightDB.rawQuery( sql: "SELECT * FROM " + DB_TABLE,  selectionArgs: null);
61          count = cr.getCount();
62
63          //Read every record and store the corresponding values to the arrayList
64          while (cr.moveToNext()){
65              int id = cr.getInt( columnIndex: 0);
66              String idxStr = String.valueOf(id);
67              String nameStr = cr.getString( columnIndex: 1);
68              String weightStr = cr.getString( columnIndex: 2);
69              record.add(new String[]{idxStr, nameStr, weightStr});
70          }
71
72          //Close the cursor
73          cr.close();
74
75          //Return the arraylist
76          return record;
77      }
78
```

BirthWeightDB.java | activity_main.xml | MainActivity.java | activity_second.xml | SecondActivity.java

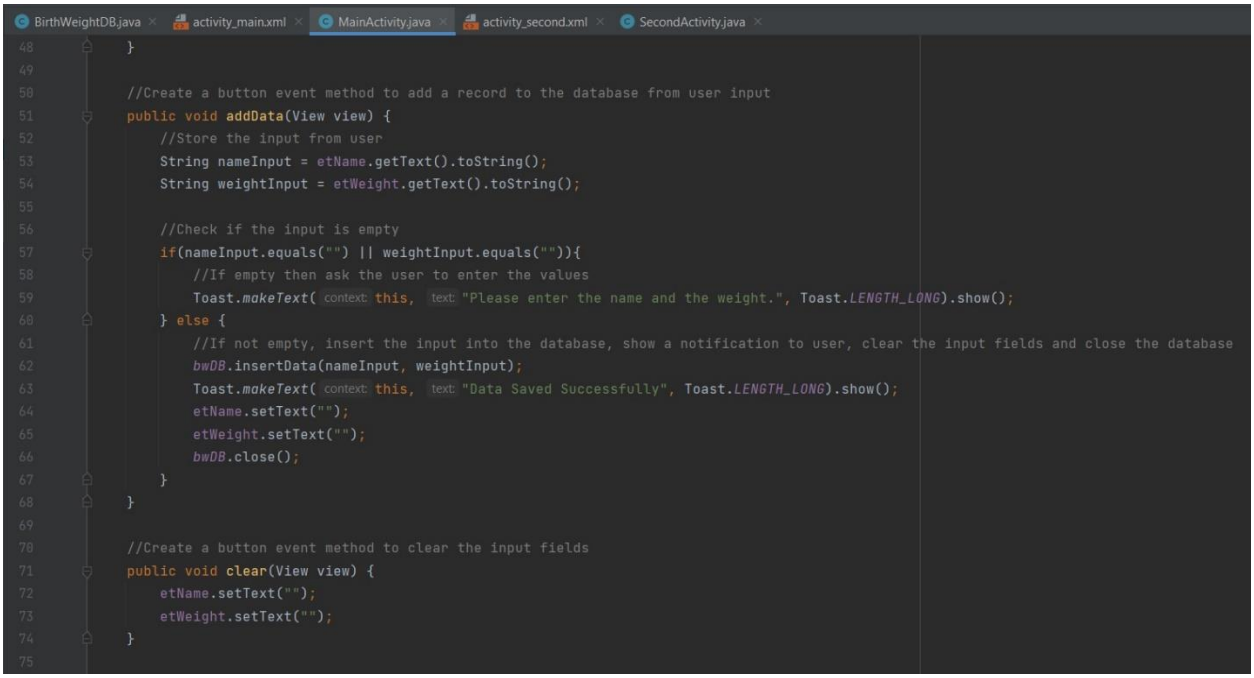
```
        //Return the arraylist
        return record;
    }

    //Create a method to delete the table/database
    public void delAll(){
        //Make the database writable
        birthWeightDB = getWritableDatabase();
        //Delete the table by executing the SQL DELETE query
        birthWeightDB.execSQL("DELETE FROM " + DB_TABLE);

        //Here is another non-SQL way to delete the database (not a table)

        //context.deleteDatabase(DB_NAME);
    }

    //Create a method to update a record in a database with the values of parameters
    public void update(String newName, String newWeight, int id){
        //Make the database writable
        birthWeightDB = getWritableDatabase();
        //Update the record to the database by executing the SQL UPDATE query with WHERE clause
        birthWeightDB.execSQL("UPDATE " + DB_TABLE + " SET Infant_Name = \'" + newName + "\', Infant_Weight = \'"
                + newWeight + "\' WHERE _id=" + id + ";");

        //Here is another non-SQL way to update an record

        //ContentValues values = new ContentValues();
        //values.put("Infant_Name", newName);
        //values.put("Infant_Weight", newWeight);
        //birthWeightDB.update(DB_TABLE, values, "_id=?", new String[]{String.valueOf(id)});
    }

    //Create a method to delete a record with id
    public void delData(int id){
        //Make the database writable
        birthWeightDB = getWritableDatabase();
        //Delete the record to the database by executing the SQL DELETE query with WHERE clause
        birthWeightDB.execSQL("DELETE FROM " + DB_TABLE + " WHERE _id=" + id + ";");

        //birthWeightDB.delete(DB_TABLE, "_id=?", new String[]{String.valueOf(id)});
    }
}
```

Step 3 – Declare and call the variables and methods in MainActivity.java for creating the database, adding records, and reading/displaying the records

Now we develop the methods in the MainActivity.java to create and manipulate a concrete database. We create a number of variables including two edittexts, one tablelayout, one public static BirthWeightDB (it is accessible in the SecondActivity), one ArrayList (for storing the records from the database), and a public static int id (the id from the first column of our database). We assign the values to the edittexts, the tablelayout and create the concrete database, bwDB, in onCreate(). Since we want to read and load the database directly after jumping back from SecondActivity, the if statement in onCreate() will automatically read and load the database if the intent is not empty. Here is the code.



Next we create several methods for button event handlers. They include addData(), clear(), loadDB(), and delDB(). After getting the input from the edittexts, we use if statement to check if the input is empty. If not, we call the insertData() from BirthWeightDB class with input as parameters to insert the input into the database. We empty the edittext and popup a message to confirm the user that the record is added successfully. We close the database at the end. It is simple for clear() which just clear the edittext fields for better user experience. Here is the code



The loadDB() method involves the longest codes in the app. We firstly call the loadData() from the BirthWeightDB class to read the database and store the records to our ArrayList. This method serves for several purposes beside just reading the records from database. It not only checks if the database is empty (the first part if statement) but also dynamically to display the records in the tablelayout. We remove all views from the tablelayout to prevent overlapping display. Then we create a tablerow with two textviews to display the table header and add them to the tablelayout. We use for-loop to create number of tablerows and textviews to display each record. We also use setId(), setClickable(), and setOnClickListener() in the loop to allow user to click each tablerow for modifying/deleting the record in the SecondActivity. Do not forget to add each tablerow into the tablelayout. At last, we clear the edittexts.

To activate the on click event on each table row, an onClickListener variable is needed in the setOnClickListener() aforementioned. We create an intent which allows the user to jump to the SecondActivity. Then we use the Id from the selected tablerow to refer the record from the ArrayList. The fields of the selected record will be stored into the intent by using putExtra(). Finally, we start the intent to jump to SecondActivity.

The last method, delDB(), is simple. We call the delAll() from the BirthWeightDB class to delete the database. Then we remove all the views from the tablelayout and pop up the message about the database deletion. Close the database and this is done. Here is the code.

BirthWeightDB.java | activity_main.xml | MainActivity.java | activity_second.xml | SecondActivity.java

```java
    //Create a button event method to read the database and display the records to user
    public void loadDB(View view) {
        //Read the database and store the values to the ArrayList
        outputAL = bwDB.loadData();

        //Check if the database is empty
        if(bwDB.count == 0) {
            //Clear the tablelayout
            tl.removeAllViews();

            //Create a textview to display a message about no data
            TextView txvNoData = new TextView( context: this);
            txvNoData.setText("There are no data yet.");
            txvNoData.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);
            txvNoData.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.MATCH_PARENT, TableRow.LayoutParams.WRAP_CONTENT));

            //Insert the textview into the tablelayout
            tl.addView(txvNoData);
        } else {
            //If the database is not empty
            //Clear the tablelayout
            tl.removeAllViews();

            //Create a table row to show the headers of records
            TableRow tr_head = new TableRow( context: this);

            //Create two textviews for the two headers
            TextView txvNameHeader = new TextView( context: this);
            txvNameHeader.setText("Infant Name");
            txvNameHeader.setTextSize(16);
            txvNameHeader.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.MATCH_PARENT, TableRow.LayoutParams.WRAP_CONTENT,  initWeight: 4f));

            TextView txvWeightHeader = new TextView( context: this);
            txvWeightHeader.setText("Weight (lbs)");
            txvWeightHeader.setTextSize(16);
            txvWeightHeader.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.MATCH_PARENT, TableRow.LayoutParams.WRAP_CONTENT,  initWeight: 1f));

            //Insert the two textviews into the table row
            tr_head.addView(txvNameHeader);
            tr_head.addView(txvWeightHeader);

            //Insert the table row to the tablelayout
            tl.addView(tr_head);
```

BirthWeightDB.java | activity_main.xml | MainActivity.java | activity_second.xml | SecondActivity.java

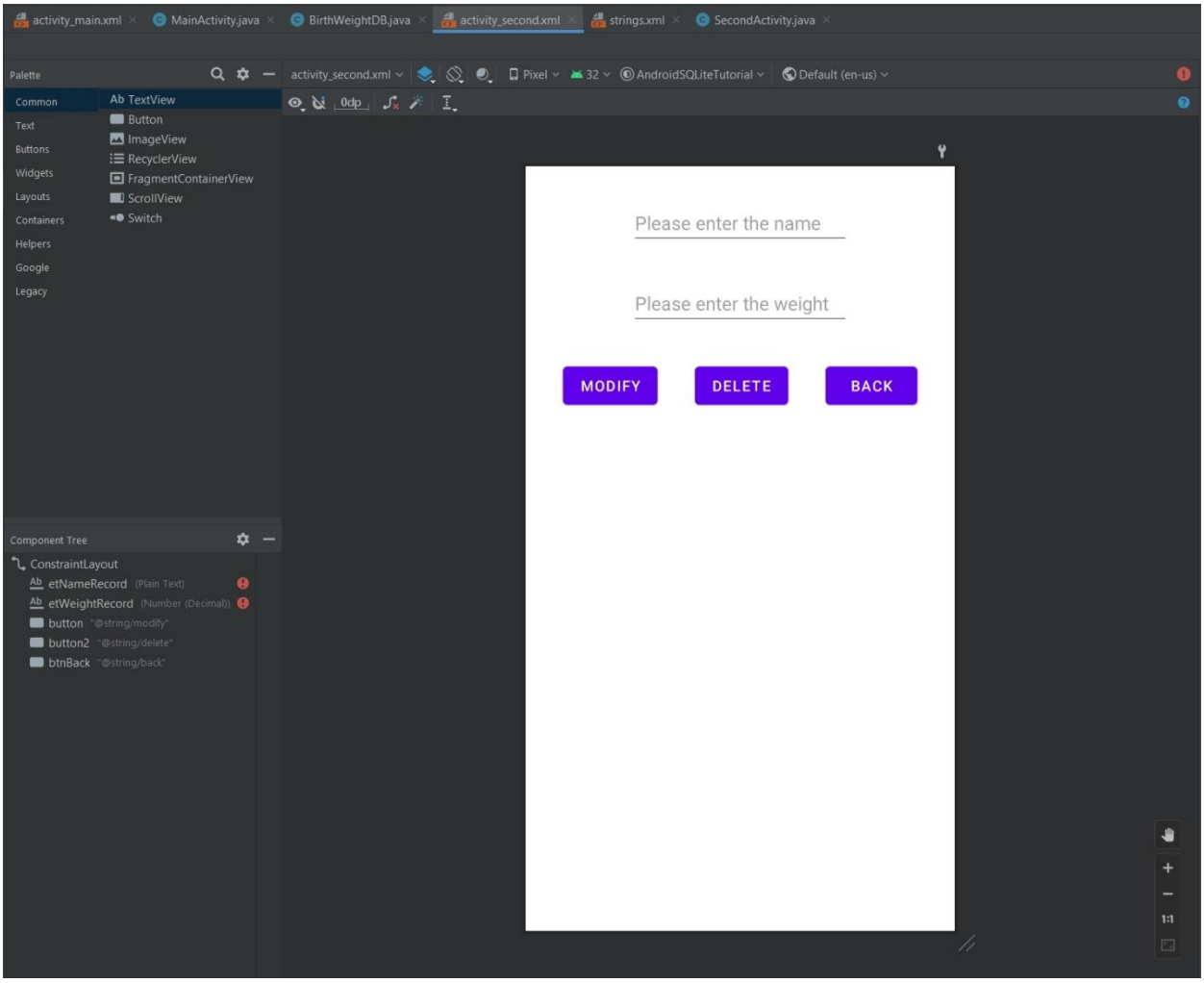
```java
            //Using for-loop to display every records from the database
            for(int i = 0; i < bwDB.count; i++){
                //Create a table tow to show the data of each record
                TableRow tr = new TableRow( context: this);

                //Create two textviews for the fields of each record
                TextView txv1 = new TextView( context: this);
                //Read the first field from the ArrayList. ([0] is the id column)
                String s1 = outputAL.get(i)[1];
                txv1.setText(s1);
                txv1.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.MATCH_PARENT, TableRow.LayoutParams.WRAP_CONTENT,  initWeight: 4f));

                TextView txv2 = new TextView( context: this);
                //Read the second field from the ArrayList
                String s2 = outputAL.get(i)[2];
                txv2.setText(s2);
                txv2.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.MATCH_PARENT, TableRow.LayoutParams.WRAP_CONTENT,  initWeight: 1f));

                //Insert the two textviews into each table row
                tr.addView(txv1);
                tr.addView(txv2);

                //Set Id and make each row to be clickable with events
                tr.setId(i);
                tr.setClickable(true);
                tr.setOnClickListener(trOnClickListener);

                //Insert each table row to the tablelayout
                tl.addView(tr);
            }

            //Clear the input fields
            etName.setText("");
            etWeight.setText("");
        }
    }
```

BirthWeightDB.java | activity_main.xml | MainActivity.java | activity_second.xml | SecondActivity.java

```
156
157     //Create an onClickListener to execute the followings when the user clicks the table row
158     public View.OnClickListener trOnClickListener = new View.OnClickListener(){
159         @Override
160 @       public void onClick(View v) {
161
162             //Create an intent of SecondActivity
163             Intent intent1 = new Intent(v.getContext(), SecondActivity.class);
164
165             //Refer which table row is clicked
166             int idx = v.getId();
167             //Get the id from the ArrayList for modifying/deleting a record in the database
168             id = Integer.parseInt(outputAL.get(idx)[0]);
169
170             //Get the field values of the selected table row for user to recognize which row is selected
171             String s1 = outputAL.get(idx)[1];
172             String s2 = outputAL.get(idx)[2];
173
174             //Store the values into the intent
175             intent1.putExtra( name: "id", id);
176             intent1.putExtra( name: "name", s1);
177             intent1.putExtra( name: "weight", s2);
178
179             //Jump to SecondActivity
180             startActivity(intent1);
181         }
182     };
183
184     //Create a button event method to delete the table/database
185     public void delDB(View view) {
186         //Delete the table/database
187         bwDB.delAll();
188         //Clear the tablelayout
189         tl.removeAllViews();
190         //Notify the user that the table/database is deleted
191         Toast.makeText( context: this,  text: "All data is erased.", Toast.LENGTH_LONG).show();
192         //Close the database
193         bwDB.close();
194     }
195 }
```

Step 4 – Create the SecondActivity.java and its layout

We create a new empty activity and called it SecondActivity for allowing user to modify/delete a certain record. The layout of the SecondActivity is much simpler. There are 2 edittexts and 3 buttons only. Here is the screenshot of the layout code and design.



BirthWeightDB.java | activity_main.xml | MainActivity.java | activity_second.xml | SecondActivity.java

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SecondActivity">

    <EditText
        android:id="@+id/etNameRecord"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:ems="10"
        android:hint="@string/please_enter_the_name"
        android:inputType="textPersonName"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/etWeightRecord"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:ems="10"
        android:hint="@string/please_enter_the_weight"
        android:inputType="numberDecimal"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etNameRecord" />

```

BirthWeightDB.java | activity_main.xml | MainActivity.java | activity_second.xml | SecondActivity.java

```
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:text="@string/modify"
        android:onClick="updateRecord"
        app:layout_constraintEnd_toStartOf="@+id/button2"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintHorizontal_chainStyle="spread"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etWeightRecord" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:text="@string/delete"
        android:onClick="delRecord"
        app:layout_constraintEnd_toStartOf="@+id/btnBack"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toEndOf="@+id/button"
        app:layout_constraintTop_toBottomOf="@+id/etWeightRecord" />

    <Button
        android:id="@+id/btnBack"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:text="@string/back"
        android:onClick="back"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toEndOf="@+id/button2"
        app:layout_constraintTop_toBottomOf="@+id/etWeightRecord" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 5 – Declare and call the variables and methods in SecondActivity.java for updating and deleting the records

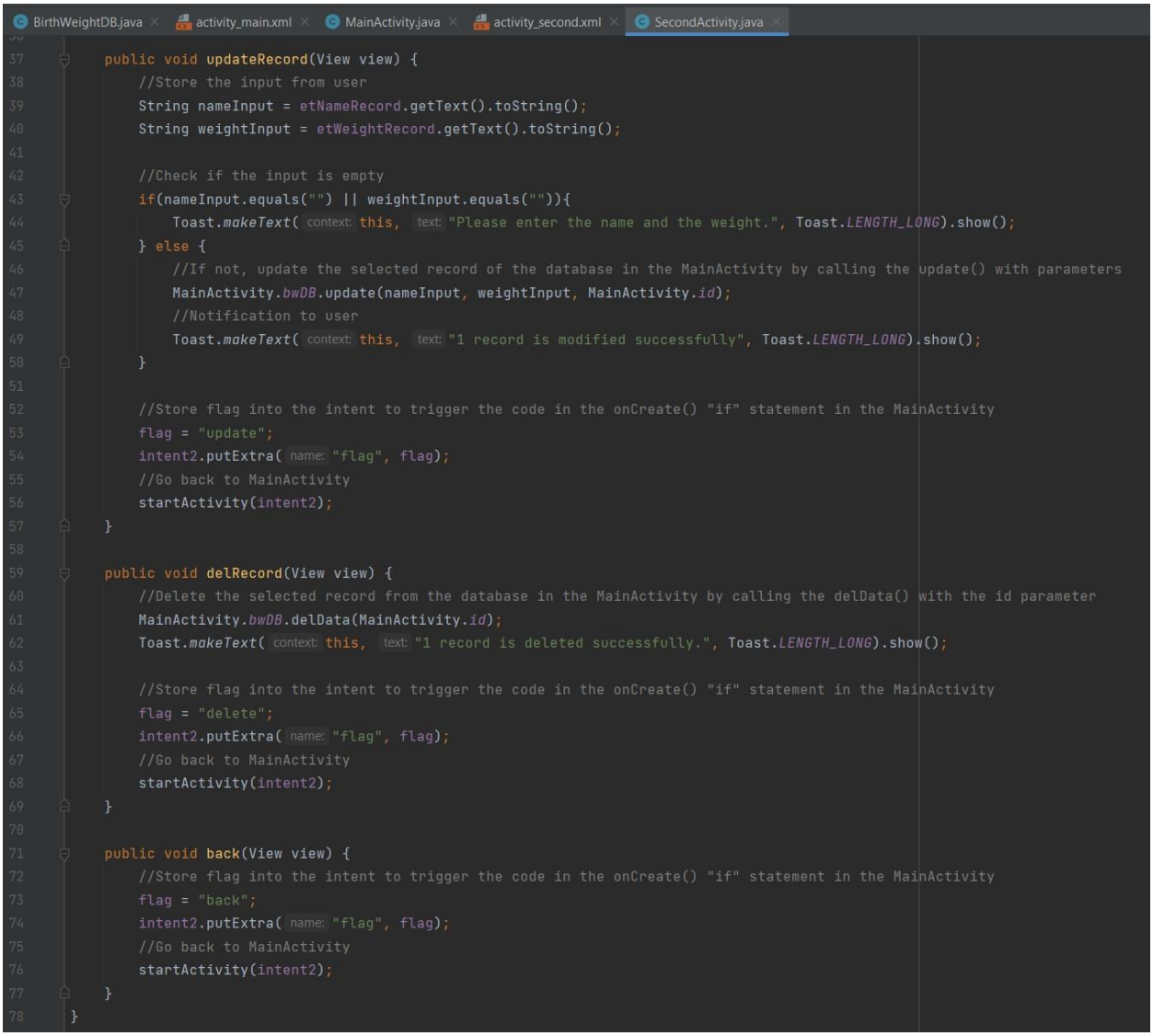
Firstly, we declare a String, two editext variables and another intent to jump back MainActivity. We then assign the corresponding values to the intent and edittext variables and set the text from the MainActivity intent by using getIntent().getStringExtra() in onCreate(). The String flag is used to store some data to show that the intent is not empty. It will trigger the loadData() in the onCreate() of MainActivity to load and display the data automatically. Here is the code.

BirthWeightDB.java | activity_main.xml | MainActivity.java | activity_second.xml | SecondActivity.java

```java
package com.example.lab3sqlitetutorial_300345759;

import ...

public class SecondActivity extends AppCompatActivity {

    //Create view variables
    private EditText etNameRecord;
    private EditText etWeightRecord;
    //Create an intent and a String variable for directing to MainActivity
    private Intent intent2;
    private String flag;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);

        //Connect the view variables with id
        etNameRecord = findViewById(R.id.etNameRecord);
        etWeightRecord = findViewById(R.id.etWeightRecord);

        //Get the field values from MainActivity
        etNameRecord.setText(getIntent().getStringExtra( name: "name"));
        etWeightRecord.setText(getIntent().getStringExtra( name: "weight"));

        //Create an intent object of MainActivity
        intent2 = new Intent( packageContext: this, MainActivity.class);
    }
```

Next we create 3 methods for 3 button event handlers which are updateRecord(), delRecord(), and back(). After getting the input from edittext and checking if the input is empty, we call the update() from BirthWeightDB class of the concrete database created in the MainAcitvity to update the record within the same database. We use the MainActivity id to indicate which record is needed to be updated. Set and assign the flag to the intent then start the intent to

jump back to MainActivity. The delRecord() is similar to updateRecord. It just calls another method, delData() from the BirthWeightDB(). The back() is just used to implement the intent to mainActivity without manipulating the database. Here is the code.




```
    public void updateRecord(View view) {
        //Store the input from user
        String nameInput = etNameRecord.getText().toString();
        String weightInput = etWeightRecord.getText().toString();

        //Check if the input is empty
        if(nameInput.equals("") || weightInput.equals("")){
            Toast.makeText( context: this,  text: "Please enter the name and the weight.", Toast.LENGTH_LONG).show();
        } else {
            //If not, update the selected record of the database in the MainActivity by calling the update() with parameters
            MainActivity.bwDB.update(nameInput, weightInput, MainActivity.id);
            //Notification to user
            Toast.makeText( context: this,  text: "1 record is modified successfully", Toast.LENGTH_LONG).show();
        }

        //Store flag into the intent to trigger the code in the onCreate() "if" statement in the MainActivity
        flag = "update";
        intent2.putExtra( name: "flag", flag);
        //Go back to MainActivity
        startActivity(intent2);
    }

    public void delRecord(View view) {
        //Delete the selected record from the database in the MainActivity by calling the delData() with the id parameter
        MainActivity.bwDB.delData(MainActivity.id);
        Toast.makeText( context: this,  text: "1 record is deleted successfully.", Toast.LENGTH_LONG).show();

        //Store flag into the intent to trigger the code in the onCreate() "if" statement in the MainActivity
        flag = "delete";
        intent2.putExtra( name: "flag", flag);
        //Go back to MainActivity
        startActivity(intent2);
    }

    public void back(View view) {
        //Store flag into the intent to trigger the code in the onCreate() "if" statement in the MainActivity
        flag = "back";
        intent2.putExtra( name: "flag", flag);
        //Go back to MainActivity
        startActivity(intent2);
    }
}
```