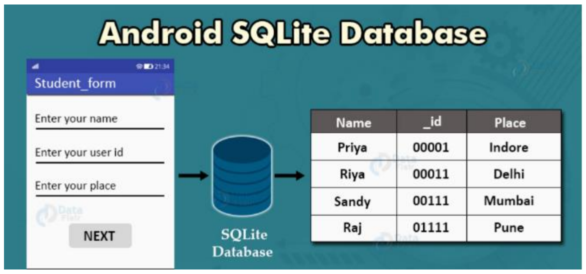
Lab V– Android Programming: Data Storage using SQLite

SQLite

SQLite is an open source SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation. SQLite supports all the relational database features.



Name	_id	Place
Priya	00001	Indore
Riya	00011	Delhi
Sandy	00111	Mumbai
Raj	01111	Pune

Database - Helper class

For managing all the operations related to the database, a helper class has been given and is called **SQLiteOpenHelper**. It automatically manages the creation and update of the database.

Follow the below steps in order to view and locate SQLite database in Android Studio using device file explorer.

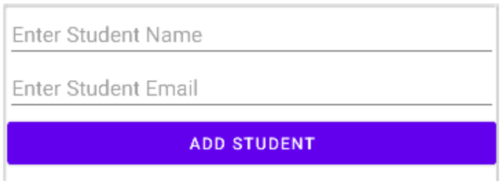
- Search for Device File Explorer in android studio
  Device file explorer can be found in the bottom-right corner of the android studio screen. Click on Device file explorer.
- Search application package name
  To search your package name, go to data > data> package name. Click on package name.
- Download the database
  Now, select database and download database whose extension will be .sqlite, for that right-click on the database name and save file at any desired location but remember the location then click on ok in Save As dialog box.
- Download SQLite browser
  Now to view the database we required SQLite browser, you can download SQLite browser from https://sqlitebrowser.org/dl/ Download a suitable SQLite browser for your device from the above link and open it.

Lab V – Android Programming: Data Storage using SQLite

Create and Add Data to SQLite Database in Android

Step 1: Create a New Project

Step 2: Create the following layout:



Step 3: Creating a new Java class for performing SQLite operations

Navigate to the app > java > your app's package name > Right-click on it > New > Java class and name it as **DBHandler** and add the below code to it.

```java
public class DBHandler extends SQLiteOpenHelper {

        private static final String DB_NAME = "studentdb";
        private static final int DB_VERSION = 1;
        private static final String TABLE_NAME = "student";
        private static final String ID_COL = "id";
        private static final String NAME_COL = "studentname";
        private static final String EMAIL_COL = "studentemail";

    // creating a constructor for our database handler.
 public DBHandler(@Nullable Context context) {
        super(context, DB_NAME, null, DB_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        String query = "CREATE TABLE " + TABLE_NAME + " ("
                + ID_COL + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + NAME_COL + " TEXT,"
                + EMAIL_COL + " TEXT)";

            db.execSQL(query);
    }
    public void addNewStudent(String studentName, String studentEmail) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(NAME_COL, studentName);
        values.put(EMAIL_COL, studentEmail);
        db.insert(TABLE_NAME, null, values);
        db.close();
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int i, int i1) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);}}
```

Step 4: Working with the MainActivity.java file

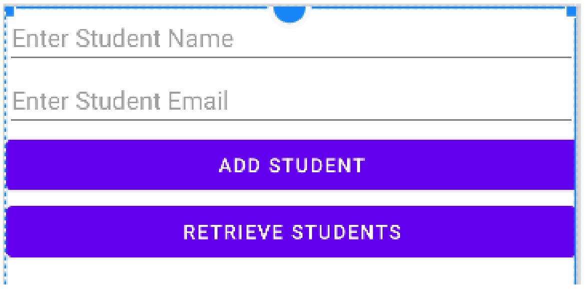
Go to the MainActivity.java file and refer to the following code.

```
public class MainActivity extends AppCompatActivity {

    EditText sname, semail;
    Button addb;
    DBHandler db;

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

        sname = (EditText) findViewById(R.id.sname);
        semail = (EditText) findViewById(R.id.semail);
        addb = (Button) findViewById(R.id.addbtn);

        // creating a new dbhandler class
        // and passing our context to it.
        db = new DBHandler(MainActivity.this);

        addb.setOnClickListener(new addbtn());
    }
    private class addbtn implements View.OnClickListener {
        @Override
        public void onClick(View view) {

            // below line is to get data from all edit text fields.
            String sn = sname.getText().toString();
            String se = semail.getText().toString();

            // validating if the text fields are empty or not.
            if (sn.isEmpty() && se.isEmpty()) {
                Toast.makeText(MainActivity.this, "Please enter all the
data..", Toast.LENGTH_SHORT).show();
            }
            // on below line we are calling a method to add new
            // student to sqlite data and pass all our values to it.
           else{ db.addNewStudent(sn, se);

            // after adding the data we are displaying a toast message.

 Toast.makeText(MainActivity.this, "Student has been added.",
Toast.LENGTH_SHORT).show();
            sname.setText("");
            semail.setText("");
}

        }
    }
}
```

Lab V – Android Programming: Data Storage using SQLite

Read Data from SQLite Database

Step 1: Go to the activity_main.xml file and add a new Button to open a new activity for displaying our list of **students**.



Step 2: Creating a modal class for storing our data

- Navigate to the app > java > your app's package name > Right-click on it > New > Java class and name it as **StudentModal** and add the below code to it.
- Add the following variables:

```
private String StudentName;
private String StudentEmail;
private int id;
```

- Generate constructors for StudentName and StudentEmail, and generate setters and getters for the three variables.

Step 3: Go to **DBHandler** and add the below code to it. In this, we are simply adding a new method for reading all the students from the SQLite database.

```
public ArrayList<StudentModal> readStudents() {
    SQLiteDatabase db = this.getReadableDatabase();
    // on below line we are creating a cursor with query to read data
from database.
    Cursor cursorStudents = db.rawQuery("SELECT * FROM " + TABLE_NAME,
null);

    // on below line we are creating a new array list.
    ArrayList<StudentModal> StudentModalArrayList = new
ArrayList<StudentModal>();
    // moving our cursor to first position.
    if (cursorStudents.moveToFirst()) {
        do {
            // on below line we are adding the data from cursor to our
array list.
StudentModalArrayList.add(new
StudentModal(cursorStudents.getString(1),cursorStudents.getString(2)));

      }
while (cursorStudents.moveToNext());
        // moving our cursor to next.
    }
    // at last closing our cursor
    // and returning our array list.
    cursorStudents.close();
    return StudentModalArrayList;
}
```

Step 4: Create a new Activity for displaying our list of Students called **ViewStudents.**

Note: To retrieve and display data from an SQLite database in Android Studio, you can use a combination of the SQLiteDatabase object, query method, and a suitable UI component (such as TextView, ListView, or RecyclerView) to display the retrieved data. In this lab, we will use **RecyclerView** to display data retrieved from the database.

Step 5: As we have added a new button to our activity_main.xml file so we have to add setOnClickListener() to that button in our MainActivity.java file.

```
viewb.setOnClickListener( new viewdata());

private class viewdata implements View.OnClickListener {
    @Override
    public void onClick(View view) {
        // opening a new activity via a intent.
        Intent i = new Intent(MainActivity.this, ViewStudents.class);
        startActivity(i);
    }
}
```

Step 6: Creating a new layout file for our item of RecyclerView

- Navigate to the app > res > layout > Right-click on it > New > Layout resource file and name it as **std_rv_item**.
- Drag a cardview →drag a linear layout→add two textviews one for the student name and one for the student email.

Step 7: Creating an Adapter class for setting data to our items of RecyclerView
Navigate to the app > java > your app's package name > Right-click on it > New > Java class and name it as **StdRVAdapter** and add the below code to it.

```

public class SRVAdapter extends
RecyclerView.Adapter<SRVAdapter.SViewHolder> {


    private ArrayList<StudentModal> sModalArrayList;
    private Context context;

     // constructor
    public SRVAdapter(ArrayList<StudentModal> sModalArrayList, Context
context) {
        this.sModalArrayList = sModalArrayList;
        this.context = context;
    }

    @NonNull
    @Override
    public SViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        // on below line we are inflating our layout
        // file for our recycler view items.
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.std_rv_item,
parent, false);
        return new SViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull SViewHolder holder, int
position) {
        // on below line we are setting data
        // to our views of recycler view item.

       StudentModal modal = sModalArrayList.get(position);
        holder.sNameTV.setText(modal.getStudentName());
        holder.sEmailTV.setText(modal.getStudentEmail());
      }

    @Override
    public int getItemCount() {
        return sModalArrayList.size();
    }

    public class SViewHolder extends RecyclerView.ViewHolder {
        private TextView sNameTV,sEmailTV;

        public SViewHolder(@NonNull View itemView1) {
            super(itemView1);
            sNameTV = itemView1.findViewById(R.id.stdname);
            sEmailTV = itemView1.findViewById(R.id.stdemail);
        }
    }
}
```

Step 8: Working with the **activity_view_student.xml** file
Navigate to the app > res > layout > activity_view_student.xml and add the below code to that file.

```
<androidx.recyclerview.widget.RecyclerView
       android:id="@+id/idRVs"
       android:layout_width="match_parent"
       android:layout_height="match_parent" />
```

Step 9: Working with the **ViewStudents**.java file

```
public class ViewStudents extends AppCompatActivity {

    private ArrayList<StudentModal> sModalArrayList;
    private DBHandler dbHandler;
    private SRVAdapter sAdapter;
    private RecyclerView sRV;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_students);

        sModalArrayList = new ArrayList<>();
        dbHandler = new DBHandler(ViewStudents.this);
        sModalArrayList = dbHandler.readStudents();

        // on below line passing our array list to our adapter class.
        sAdapter = new SRVAdapter(sModalArrayList, ViewStudents.this);
        sRV = (RecyclerView) findViewById(R.id.idRVs);

        // setting layout manager for our recycler view.
        LinearLayoutManager linearLayoutManager = new
LinearLayoutManager(ViewStudents.this, RecyclerView.VERTICAL, false);
        sRV.setLayoutManager(linearLayoutManager);

        // setting our adapter to recycler view.
        sRV.setAdapter(sAdapter);
    }
}
```

Lab V – Android Programming: Data Storage using SQLite

Update Data to SQLite Database

Step 1: Create a new empty Activity called **UpdateStudent**.

Step 2: Working with the activity_update_student.xml file.

You can copy the same layout of the activity_main.xml.

Step 3: Updating our **DBHandler** class

```
public void updateStudent(String originalStudentName, String
StudentName, String StudentEmail) {

    // calling a method to get writable database.
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();

    // on below line we are passing all values along with its key and
value pair.
    values.put(NAME_COL, StudentName);
    values.put(EMAIL_COL, StudentEmail);

    // on below line we are calling a update method to update our
database and passing our values.
    // and we are comparing it with name of our student which is stored
in original name variable.
    db.update(TABLE_NAME, values, "studentname=?", new
String[]{originalStudentName});
    db.close();
}
```

Step 4: Updating our **StdRVAdapter.java** class

As we will be opening a new activity to update our student. We have to add on click listener for the items of our RecycleView. Add the below code to your adapter class.

```
public void onBindViewHolder(@NonNull SViewHolder holder, int position)
{
   StudentModal modal = sModalArrayList.get(position);
    holder.sNameTV.setText(modal.getStudentName());
    holder.sEmailTV.setText(modal.getStudentEmail());

    holder.itemView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent i= new Intent(context,UpdateStudent.class);
            i.putExtra("name", modal.getStudentName());
            i.putExtra("email", modal.getStudentEmail());
            context.startActivity(i);
        }
    });
}
```

Lab V – Android Programming: Data Storage using SQLite

Step 5: Working with the **UpdateStudent**.java file.

```
public class UpdateStudent extends AppCompatActivity {
    private EditText sNameEdt, sEmailEdt;
    private Button updateStdBtn, deleteStdBtn;
    private DBHandler dbHandler;
    String sName, sEmail;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_update_student);

        sNameEdt = findViewById(R.id.usname);
        sEmailEdt = findViewById(R.id.usemail);

        // on below line we are initialing our dbhandler class.
        dbHandler = new DBHandler(UpdateStudent.this);

        // on below lines we are getting data which
        // we passed in our adapter class.
        sName = getIntent().getStringExtra("name");
        sEmail = getIntent().getStringExtra("email");

        // setting data to edit text
        // of our update activity.
        sNameEdt.setText(sName);
        sEmailEdt.setText(sEmail);
        deleteStdBtn=(Button)findViewById(R.id.deletebtn) ;
        updateStdBtn=(Button) findViewById(R.id.updatebtn);
// adding on click listener for delete button to delete our student.
        deleteStdBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // calling a method to delete our student.
                dbHandler.deleteStudent(sName);
                Toast.makeText(UpdateStudent.this, "Deleted the
student", Toast.LENGTH_SHORT).show();
                Intent i = new Intent(UpdateStudent.this,
MainActivity.class);
                startActivity(i);
            }
        });

        // adding on click listener to our update student button.
        updateStdBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                // inside this method we are calling an update student
                // method and passing all our edit text values.
                dbHandler.updateStudent(sName,
sNameEdt.getText().toString(), sEmailEdt.getText().toString());
```

```
                    // displaying a toast message that our student has been
updated.
                    Toast.makeText(UpdateStudent.this, "Student Updated..",
Toast.LENGTH_SHORT).show();

                    // launching our main activity.
                    Intent i = new Intent(UpdateStudent.this,
MainActivity.class);
                    startActivity(i);
               }
          });
     }
}
```

Delete Data in SQLite Database

Step 1: Updating our DBHandler class

```
public void deleteStudent(String StudentName) {

     SQLiteDatabase db = this.getWritableDatabase();
     db.delete(TABLE_NAME, "studentname=?", new String[]{StudentName});
     db.close();
}
```

Step 2: Adding a button to delete our student
Navigate to the app > res > layout > **activity_update_student.xml** file and add a Button inside this layout for deleting a student.

Step 3: Initializing our button to delete our student
Navigate to the app > java > your app's package name > **UpdateStudent.java** file and add the below code to it.

```
deleteStdBtn.setOnClickListener(new View.OnClickListener() {
     @Override
     public void onClick(View v) {
          // calling a method to delete our course.
          dbHandler.deleteStudent(sName);
          Toast.makeText(UpdateStudent.this, "Deleted the student",
Toast.LENGTH_SHORT).show();
          Intent i = new Intent(UpdateStudent.this, MainActivity.class);
          startActivity(i);
     }
});
```