Experiment 11: SQLite Databases

1. Objective

Enhance the to-do list application by integrating an SQLite database to store the tasks persistently. This will cover basic database operations in Android.

2. Steps to Complete the Experiment

1. Design the Database Schema:

   Determine the structure of your database, including the tables and columns you will need. For a simple application, you might start with a single table. For example, a Tasks table for a to-do list app might include columns for ID, TaskName, and Completed.

2. Create a SQLite Helper Class:

   Extend SQLiteOpenHelper in a new class, e.g., DatabaseHelper. Implement the onCreate and onUpgrade methods to manage database creation and version management.

DatabaseHelper Class

```
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "todoList.db";
    private static final String TABLE_NAME = "tasks";
    private static final String COL_1 = "ID";
    private static final String COL_2 = "TASKNAME";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE " + TABLE_NAME + " (ID INTEGER
PRIMARY KEY AUTOINCREMENT, TASKNAME TEXT)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
    }

    public boolean insertData(String taskName) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(COL_2, taskName);
        long    result    =    db.insert(TABLE_NAME,    null,
contentValues);
        return result != -1; // Returns 'true' if insertion is
successful
    }

    public Cursor getAllData() {
        SQLiteDatabase db = this.getWritableDatabase();
        return db.rawQuery("SELECT * FROM " + TABLE_NAME, null);
    }
}
```

3. Implement Data Operations:

   Within your DatabaseHelper class, add methods to insert, update, delete, and query data. For example, add a method to insert new tasks, mark a task as completed, delete a task, and fetch all tasks.

4. Integrate Database with UI:

   In your activity, use the DatabaseHelper class to interact with the database. For example, when a new task is added through the UI, call the insertion method from your DatabaseHelper.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
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">

<EditText
    android:id="@+id/etNewTask"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Enter a new task"
    android:inputType="text"
    android:layout_margin="16dp"
    android:layout_above="@+id/btnAddTask"/>

<Button
    android:id="@+id/btnAddTask"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Add Task"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="16dp"/>

<ListView
    android:id="@+id/lvTasks"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_above="@id/etNewTask"
```

```
        android:divider="@android:color/darker_gray"
        android:dividerHeight="1dp"/>

</RelativeLayout>
```

5. Display Data:

   Fetch data from the database and display it in your app's UI, such as in a ListView or RecyclerView. You might use a CursorAdapter or a custom adapter to bind your data to the UI components.

6. Handle User Interactions:

   Implement UI elements that allow users to interact with the data. For example, add buttons or gestures to complete or delete tasks, and update the database accordingly.

MainActivity.java

```java
import androidx.appcompat.app.AppCompatActivity;

import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {
    DatabaseHelper myDb;
    EditText editTextNewTask;
    Button buttonAddTask;
    ListView listViewTasks;

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

        myDb = new DatabaseHelper(this);
        editTextNewTask = findViewById(R.id.etNewTask);
        buttonAddTask = findViewById(R.id.btnAddTask);
        listViewTasks = findViewById(R.id.lvTasks);

        buttonAddTask.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String                    newTask                    =
editTextNewTask.getText().toString();
                if (!newTask.isEmpty()) {
                    myDb.insertData(newTask);
                    populateListView();
                    editTextNewTask.setText("");
                }
            }
        });

        populateListView();
    }

    private void populateListView() {
        ArrayList<String> taskList = new ArrayList<>();
        Cursor data = myDb.getAllData();

        while (data.moveToNext()) {
            // The 1 here is the index of the column in your
database table
            taskList.add(data.getString(1));
        }

        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1, taskList);
        listViewTasks.setAdapter(adapter);
    }
}
```

7. Test the Application:

   Thoroughly test all database operations, including inserting, updating, deleting, and querying data. Ensure the UI correctly reflects the data changes.

3. Explanation

EditText (etNewTask): Allows users to input a new task. Positioned at the bottom of the layout with a margin for clarity.

Button (btnAddTask): When clicked, the app will add the content of the EditText as a new task into the SQLite database and refresh the task list displayed in the ListView.

ListView (lvTasks): Displays the list of tasks retrieved from the SQLite database. It fills the majority of the screen space, showing tasks above the EditText.

Database Helper (DatabaseHelper): Manages database creation and version control. It provides methods to insert new tasks (insertData) and fetch all tasks (getAllData).

Adding Tasks (buttonAddTask OnClickListener): When the "Add Task" button is clicked, the text from the EditText is inserted as a new task in the database, and the ListView is updated.

Displaying Tasks (populateListView): Fetches all tasks from the database and displays them in the ListView using an ArrayAdapter.