모바일프로그래밍
- 안드로이드(Android) -

- 안드로이드 기능/실습 – 17
  - RecyclerView 구성

- 안드로이드 기능/실습 – 18
  - SQLite 사용법
  - SQLite 기본 구현 과정

❑ 안드로이드 기능 / 실습 - 17
RECYCLERVIEW 구성

❑ RecyclerView

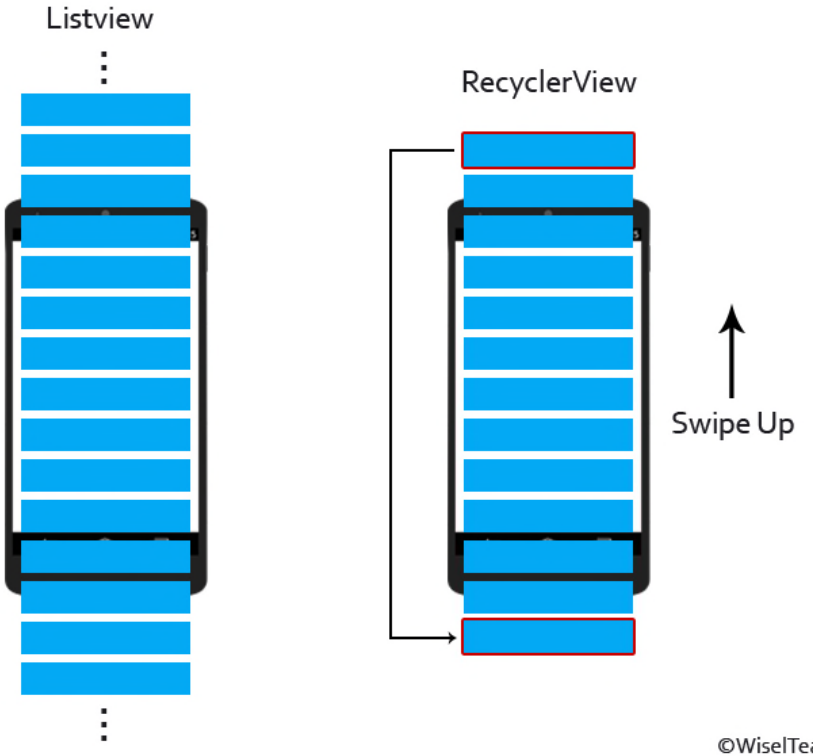
❍목적

- ❑ 리사이클러뷰 이해 및 구성 방법 습득

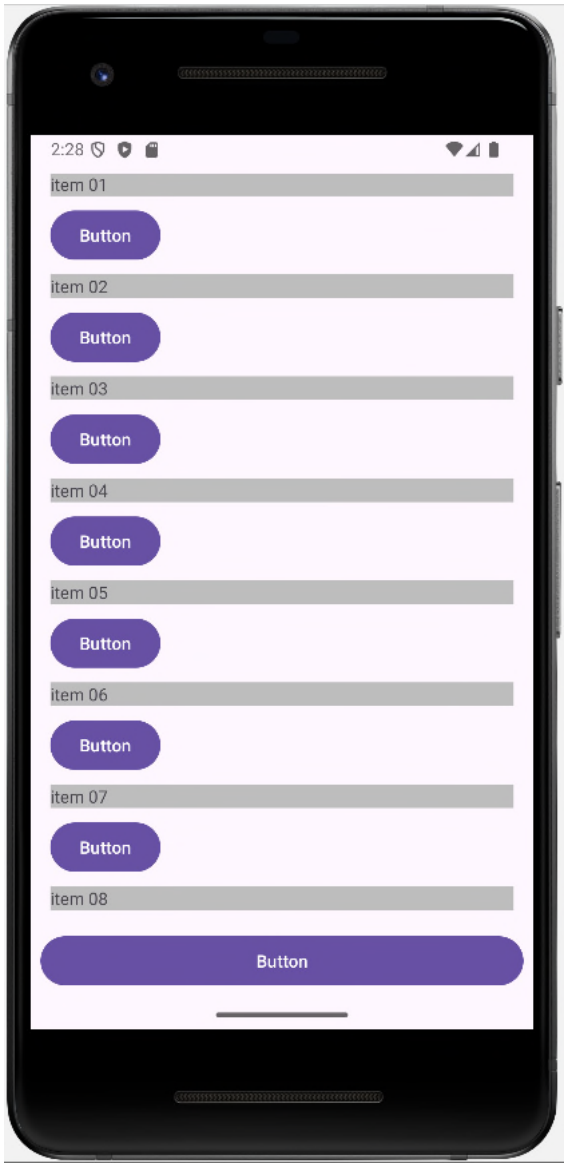
❍RecyclerView

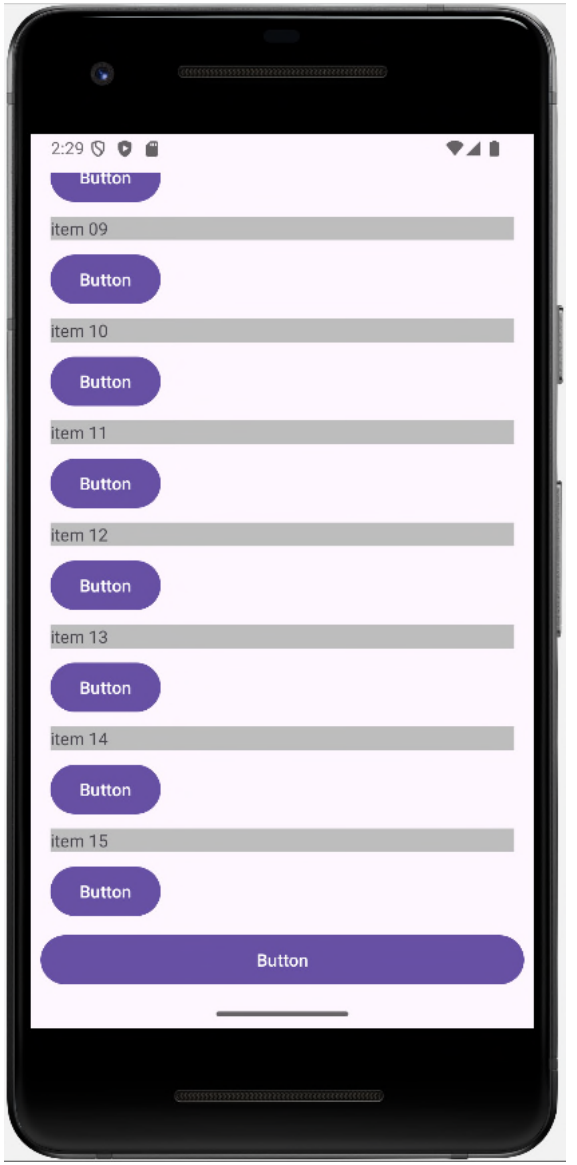
- ❑ 각 아이템을 목록형태로 화면에 나타내는데 사용되며 대량의 데이터를 효율적으로 화면에 표시하기 위해서 사용

❍개념



❑ 실행 모습


2:28
item 01
Button
item 02
Button
item 03
Button
item 04
Button
item 05
Button
item 06
Button
item 07
Button
item 08
Button


2:29
Button
item 09
Button
item 10
Button
item 11
Button
item 12
Button
item 13
Button
item 14
Button
item 15
Button

❑ RecyclerView 구현

❍ 구현 방법

- ❑ 1. RecyclerView를 가진 Layout 구성
- ❑ 2. 각 데이터 아이템을 표현하기 위한 레이아웃 구성
- ❑ 3. RecyclerView 표현(LayoutManager 객체 이용) 정의
- ❑ 4. RecyclerView 어댑터 구현
  - ❖ 생성자를 통해서 데이터를 전달받도록 구현
  - ❖ 사용자 ViewHolder 구현
  - ❖ 3개의 메소드를 구현
    - onCreateViewHolder() : ViewHolder 객체를 생성하고 초기화
    - onBindVlewHolder() : 데이터를 가져와 ViewHolder 안의 내용을 채움
    - getItemCount() : 총 데이터 개수를 반환

❑ RecyclerView 구현

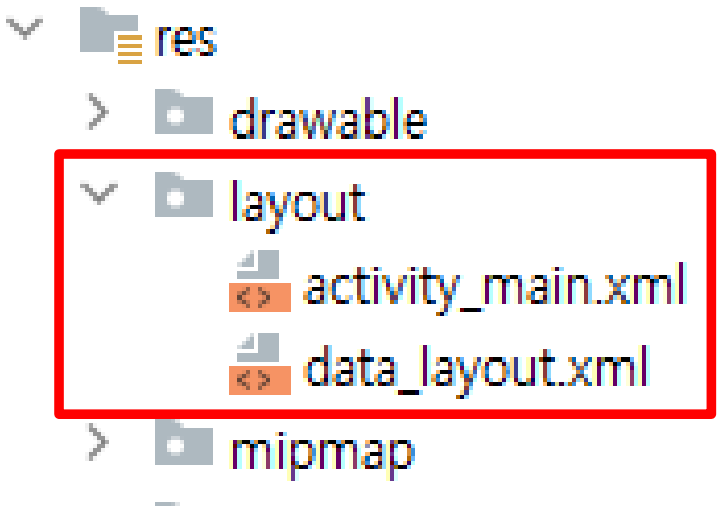
- onCreateViewHolder()
  - ViewHolder 객체를 생성하고 초기화
  - 뷰 객체를 담고 있는 ViewHolder가 생성되는 함수

- onBindViewHolder()
  - onCreateVIewHolder에서 리턴한 ViewHolder에 데이터를 셋팅
  - 생성된 뷰홀더에 데이터를 바인딩 해주는 함수

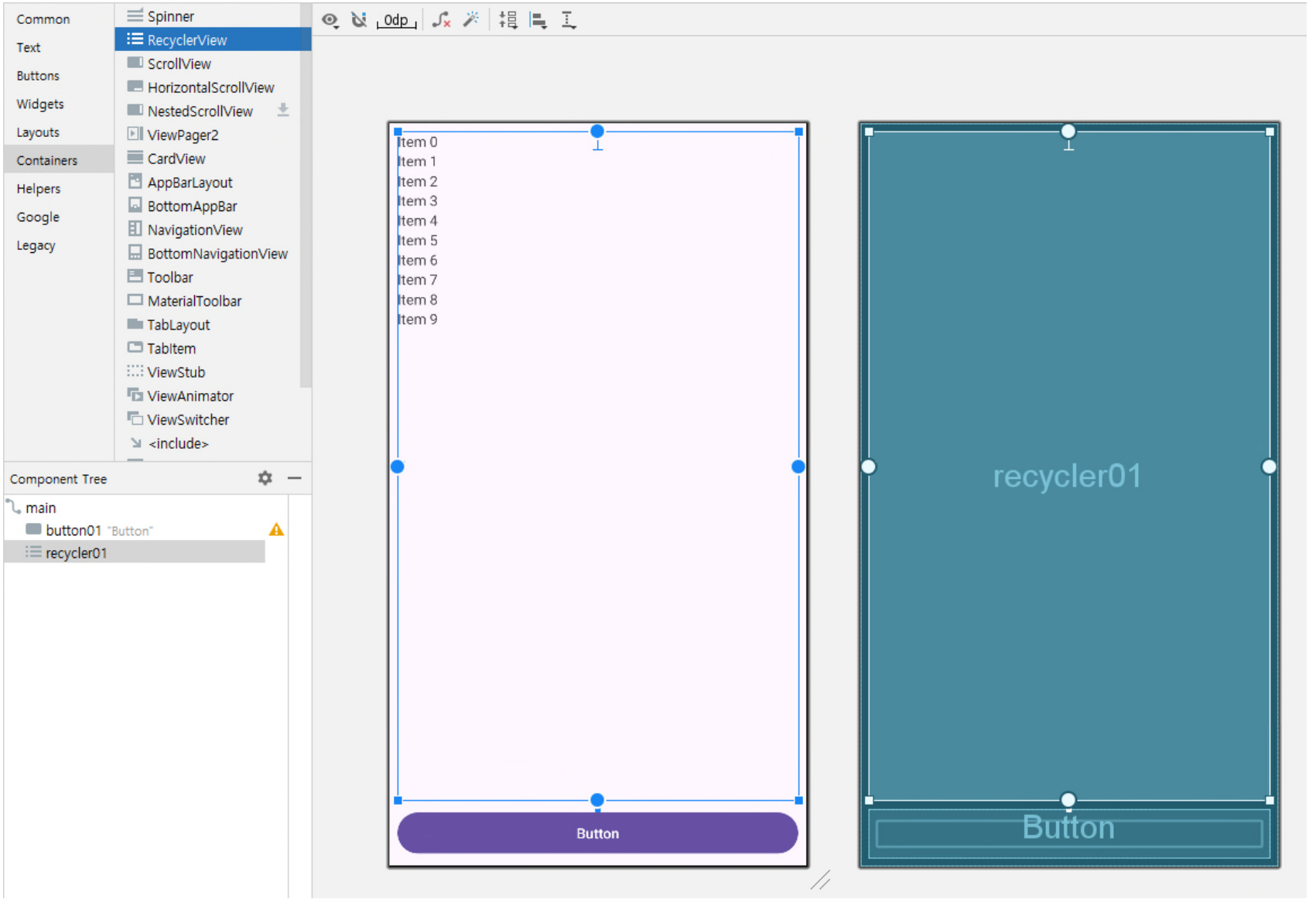
- getItemCount()
  - 연결하고자 하는 아이템의 총 개수

❑ 레이아웃 구성

- RecyclerView를 가진 메인 레이아웃과 데이터를 표현하기 위한 레이아웃으로 구성
  - 데이터를 표현하기 위한 별도의 레이아웃 파일 추가



❑ 레이아웃 구성 - activity_main.xml



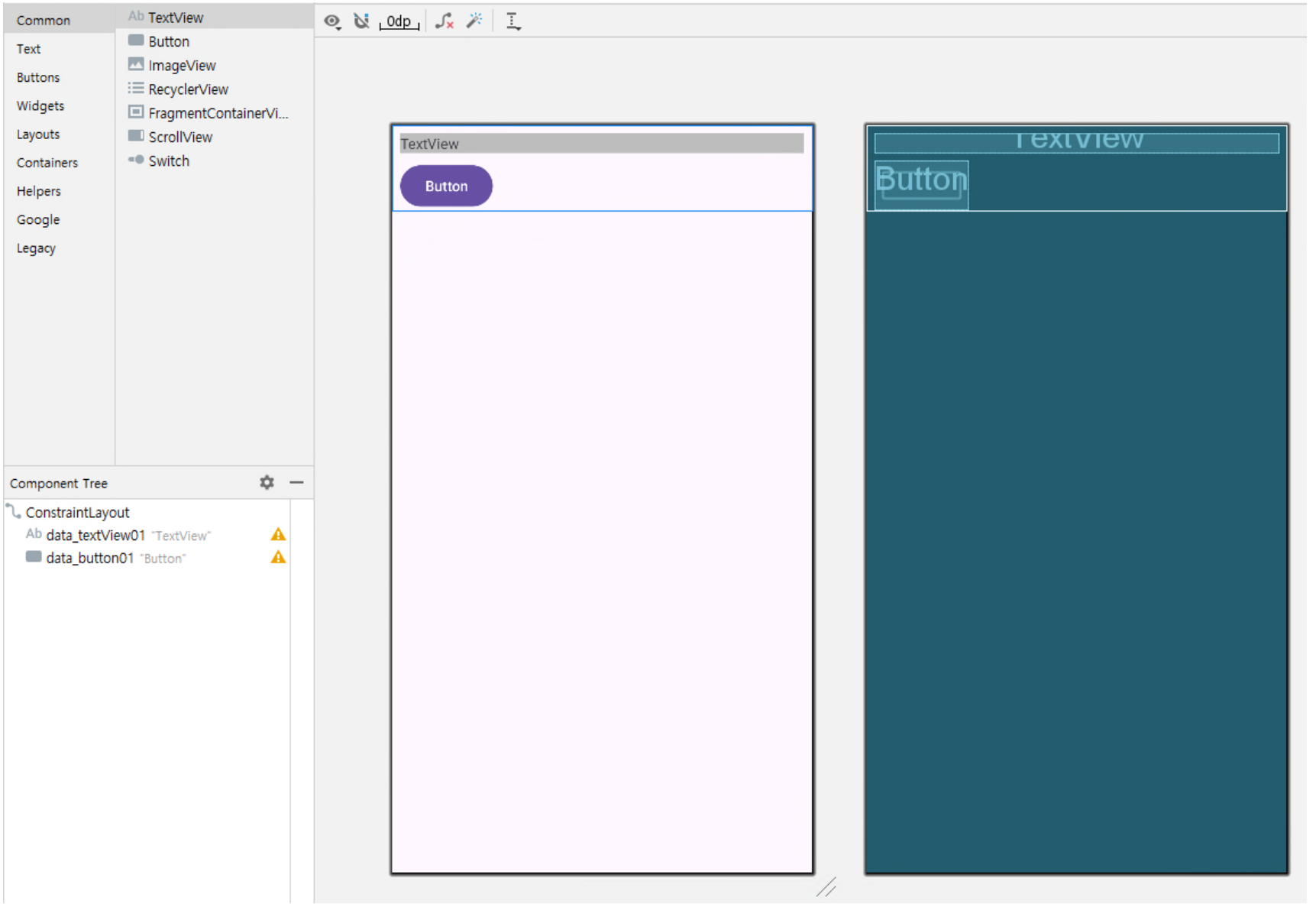
❑ 레이아웃 구성 - activity_main.xml

```
<?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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button01"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:text="Button"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
```

❑ 레이아웃 구성 - activity_main.xml

```
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycler01"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        app:layout_constraintBottom_toTopOf="@+id/button01"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

❑ 레이아웃 구성 - data_layout.xml



❑ 레이아웃 구성 - data_layout.xml

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

    <TextView
        android:id="@+id/data_textView01"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:background="#FFBDBDBD"
        android:text="TextView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

❑ 레이아웃 구성 - data_layout.xml

```
    <Button
        android:id="@+id/data_button01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="Button"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/data_textView01" />

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

❑ 코드 작성

```
package com.practice.ex.recyclerview;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;

import androidx.activity.EdgeToEdge;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

public class MainActivity extends AppCompatActivity {

    Button btn01;  2 usages

    RecyclerView reclyclerView01;  3 usages
    RecyclerView.Adapter adapter;  2 usages

    private final String[] items = { "item 01", "item 02", "item 03", "item 04", "item 05",  1 usage
                                     "item 06", "item 07", "item 08", "item 09", "item 10",
                                     "item 11", "item 12", "item 13", "item 14", "item 15" };
```

여기서는 데이터 관련 클래스 등은 구현하지 않음
기본적인 RecyclerView 구현 과정만을 보임

❑ 코드 작성

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_main);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {...});

    btn01 = (Button) findViewById(R.id.button01);
    btn01.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            finish();
        }
    });

    reclyclerView01 = (RecyclerView) findViewById(R.id.recycler01);

    reclyclerView01.setLayoutManager(new LinearLayoutManager( context: this));

    adapter = new RecyclerAdpater(items);

    reclyclerView01.setAdapter(adapter);
}
```

LayoutManager를 이용하여 보이는 모습을 정의

- LinearLayoutManager
- GridLayoutManager
- StaggeredGridLayoutManager

❑ 코드 작성

```
private class RecyclerAdpater extends RecyclerView.Adapter<RecyclerAdpater.MyViewHolder> {  2 usages

    String[] data_items;  3 usages

    public RecyclerAdpater(String[] items) {  1 usage
        data_items = items;
    }

    public class MyViewHolder extends RecyclerView.ViewHolder {  5 usages

        TextView data_text01;  2 usages
        Button data_btn01;  1 usage

        public MyViewHolder(@NonNull View itemView) {  1 usage
            super(itemView);

            this.data_text01 = itemView.findViewById(R.id.data_textView01);
            this.data_btn01 = itemView.findViewById(R.id.data_button01);
        }
    }
```

RecyclerView 어댑터는 RecyclerView.Adapter를 상속받아서 구현하는데 제네릭으로 구현하고자 하는 어댑터의 뷰홀더를 부여해야 함
따라서 일반적으로 아래 뷰홀더를 먼저 구현한 후 설정

RecyclerView 클래스 내의 추상 클래스인 ViewHolder를 상속받아 사용자 뷰홀더를 구성

❑ 코드 작성

```
    @NonNull
    @Override
    public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View hoolderView = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.data_layout, parent,  attachToRoot: false);
        MyViewHolder myViewHolder = new MyViewHolder(hoolderView);
        return myViewHolder;
    }
```

뷰 객체를 담고 있는 ViewHolder가 생성되는 함수

```
    @Override
    public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
        holder.data_text01.setText(data_items[position]);
    }
```

생성된 ViewHolder에 데이터를 설정

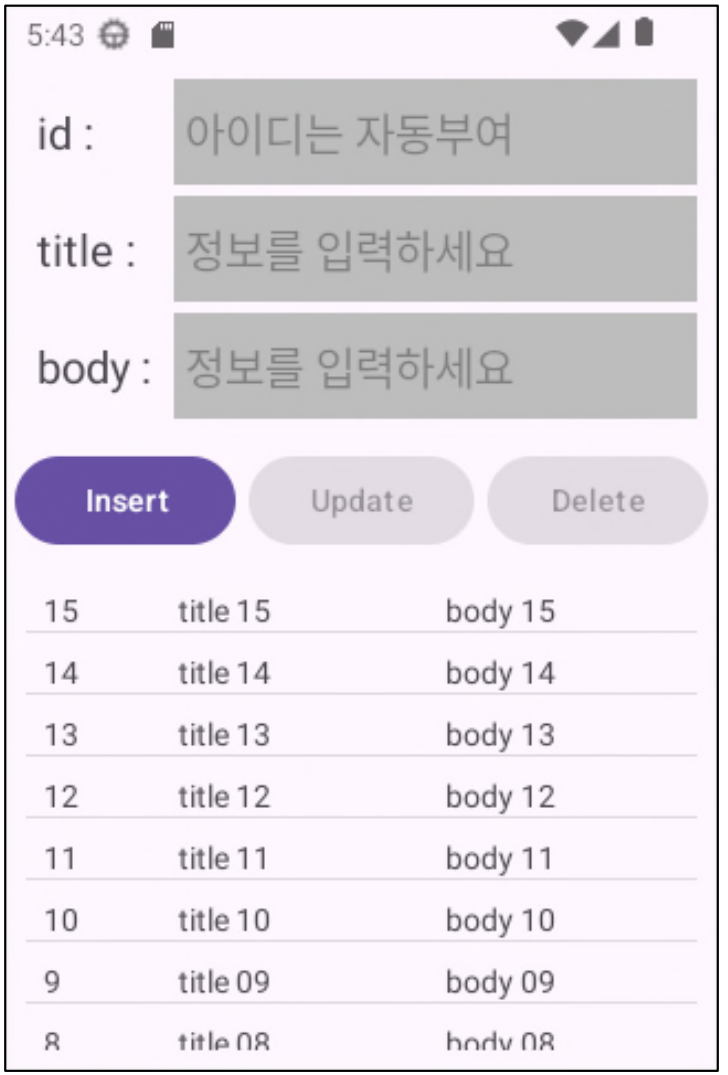
```
    @Override
    public int getItemCount() {
        return data_items.length;
    }
}
}
```

데이터의 전체 개수를 리턴

❑ 안드로이드 기능/실습 - 18 SQLITE

- 테이블레이아웃 기초 사용법 습득
- 데이터베이스 구축을 위한 SQLite 사용 방법 습득
- 입력, 수정, 삭제 등의 기능 확인
- 데이터베이스 연결 및 생성 등의 과정에 대한 세부적 확인 과정이 필요

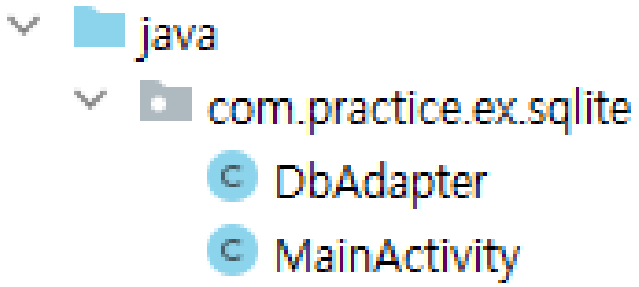
❑ 실행 모습


5:43
id :
아이디는 자동부여
title :
정보를 입력하세요
body :
정보를 입력하세요
Insert
Update
Delete
15 title 15 body 15
14 title 14 body 14
13 title 13 body 13
12 title 12 body 12
11 title 11 body 11
10 title 10 body 10
9 title 09 body 09
8 title 08 body 08

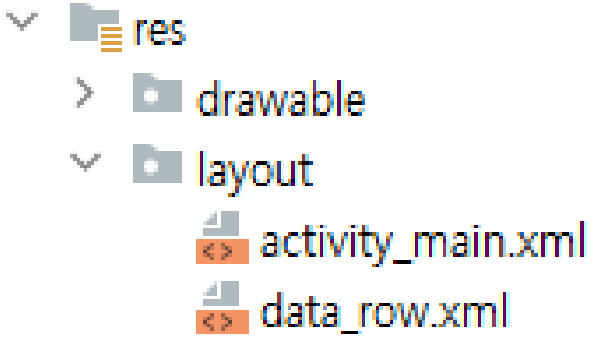

5:44
id :
8
title :
title 08
body :
body 08
Insert
Update
Delete
15 title 15 body 15
14 title 14 body 14
13 title 13 body 13
12 title 12 body 12
11 title 11 body 11
10 title 10 body 10
9 title 09 body 09
8 title 08 body 08

○기본 구조

❑ 코드 부분



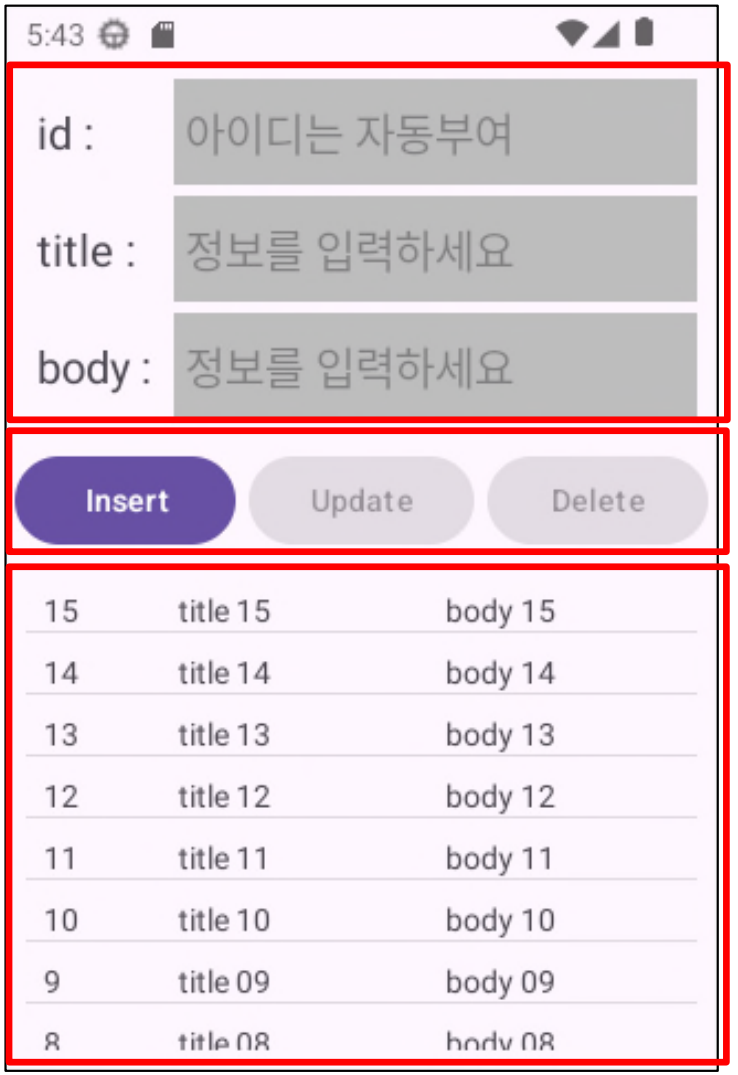
❑ 레이아웃 부분



MainActivity의 레이아웃

MainActivity 내의 ListView의 데이터 표현을 위한 레이아웃

❑ 레이아웃 구성

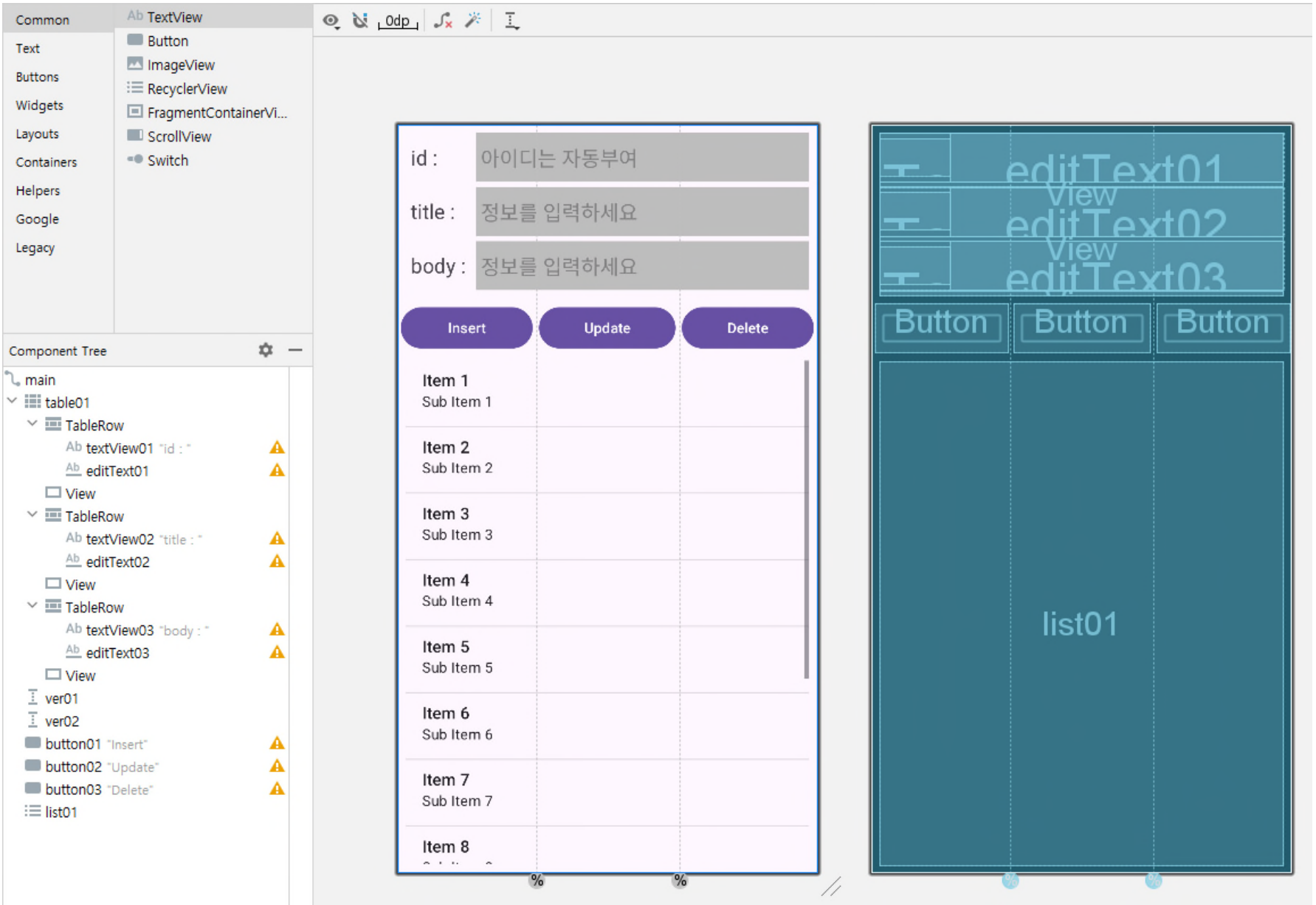


데이터 입력창
- id는 자동 부여, 따라서 비활성화

데이터 처리를 위한 버튼
- 입력, 갱신, 삭제 기능
- 갱신과 삭제는 해당 아이템을 선택했을 경우에만 수행, 따라서 선택한 경우만 활성화

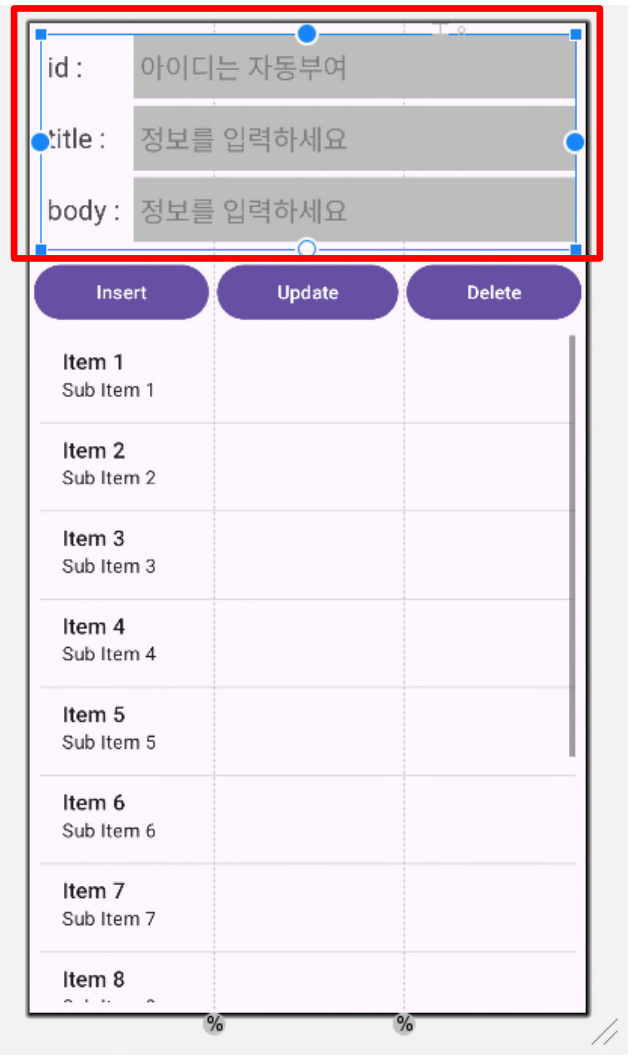
데이터 출력창
- ListView를 사용하여 데이터베이스에 존재하는 현재 내용을 출력
- 데이터 출력을 위한 레이아웃을 별도로 이용

❑ 레이아웃 구성


Common
Text
Buttons
Widgets
Layouts
Containers
Helpers
Google
Legacy
TextView
Button
ImageView
RecyclerView
FragmentContainerVi...
ScrollView
Switch
0dp
Component Tree
main
table01
TableRow
textView01 "id : "
editText01
View
TableRow
textView02 "title : "
editText02
View
TableRow
textView03 "body : "
editText03
View
ver01
ver02
button01 "Insert"
button02 "Update"
button03 "Delete"
list01
id :
아이디는 자동부여
title :
정보를 입력하세요
body :
정보를 입력하세요
Insert
Update
Delete
Item 1
Sub Item 1
Item 2
Sub Item 2
Item 3
Sub Item 3
Item 4
Sub Item 4
Item 5
Sub Item 5
Item 6
Sub Item 6
Item 7
Sub Item 7
Item 8
editText01
View
editText02
View
editText03
Button
Button
Button
list01

❑ 레이아웃 - activity_main.xml

- 여기서는 데이터 입력을 위한 부분인 (id, title, body)에 대해서 테이블 레이아웃을 이용하였음
  - ❑ 기존 레이아웃을 이용해도 상관없음
  - ❑ 코드는 테이블 레이아웃을 이용하는 방법도 함께 습득하는 것으로 진행



테이블 레이아웃으로 작성한 부분

```
<?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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

❑ 레이아웃 - activity_main.xml

```
<TableLayout
    android:id="@+id/table01"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:stretchColumns="1"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <TableRow>

        <TextView
            android:id="@+id/textView01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="id : "
            android:textSize="20sp" />

        <EditText
            android:id="@+id/editText01"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#BDBDBD"
            android:inputType="textPersonName"
            android:padding="5dp"
            android:ems="5"
            android:minHeight="48dp"
            android:hint="아이디는 자동부여"
            android:textSize="20sp" />

    </TableRow>
```

TableLayout
- 전체 디자인을 관리
- <TableRow>를 이용하여 행을 구성
- 위젯이 가장 많은 행을 기준으로 위젯 하나당 열 하나를 생성
- android:stretchColumns를 이용하여 남은 공간을 어느 컬럼에서 차지할 것인지를 결정 위치 값은 0부터 시작

TableRow 아래에 위젯들을 배치
- android:layout_column = “” 를 이용하여 위젯의 배치되는 컬럼 위치를 결정 (여기서는 불필요하여 기재하지 않음)
- android:layout_span = “”을 이용하여 여러 컬럼에 걸쳐서 배치하는 것이 가능 (여기서는 불필요하여 기재하지 않음)

❑ 레이아웃 - activity_main.xml

```
<View android:layout_height="5dp" />

<TableRow>

    <TextView
        android:id="@+id/textView02"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="5dp"
        android:text="title : "
        android:textSize="20sp" />

    <EditText
        android:id="@+id/editText02"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#BDBDBD"
        android:inputType="textPersonName"
        android:padding="5dp"
        android:minHeight="48dp"
        android:hint="정보를 입력하세요"
        android:textSize="20sp" />

</TableRow>
```

여기서는 TableLayout에서 행의 구분선이 필요하여 <View>를 이용하여 표현

두 번째 행

❑ 레이아웃 - activity_main.xml

```
    <View android:layout_height="5dp" />

    <TableRow>

        <TextView
            android:id="@+id/textView03"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="body : "
            android:textSize="20sp" />

        <EditText
            android:id="@+id/editText03"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#BDBDBD"
            android:inputType="textPersonName"
            android:padding="5dp"
            android:minHeight="48dp"
            android:hint="정보를 입력하세요"
            android:textSize="20sp" />
    </TableRow>

    <View android:layout_height="5dp" />

</TableLayout>
```

세 번째 행

❑ 레이아웃 - activity_main.xml

```
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/ver01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintGuide_percent="0.33" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/ver02"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintGuide_percent="0.67" />
```

버튼을 3개를 동일한 크기로 나누어 표시하기 위한 가이드라인

```
<Button
    android:id="@+id/button01"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="3dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="3dp"
    android:text="Insert"
    android:textAllCaps="false"
    app:layout_constraintEnd_toStartOf="@+id/ver01"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/table01" />
```

버튼 생성 부분

- 위 가이드라인을 이용하여 동일한 크기로 배정되게 표시

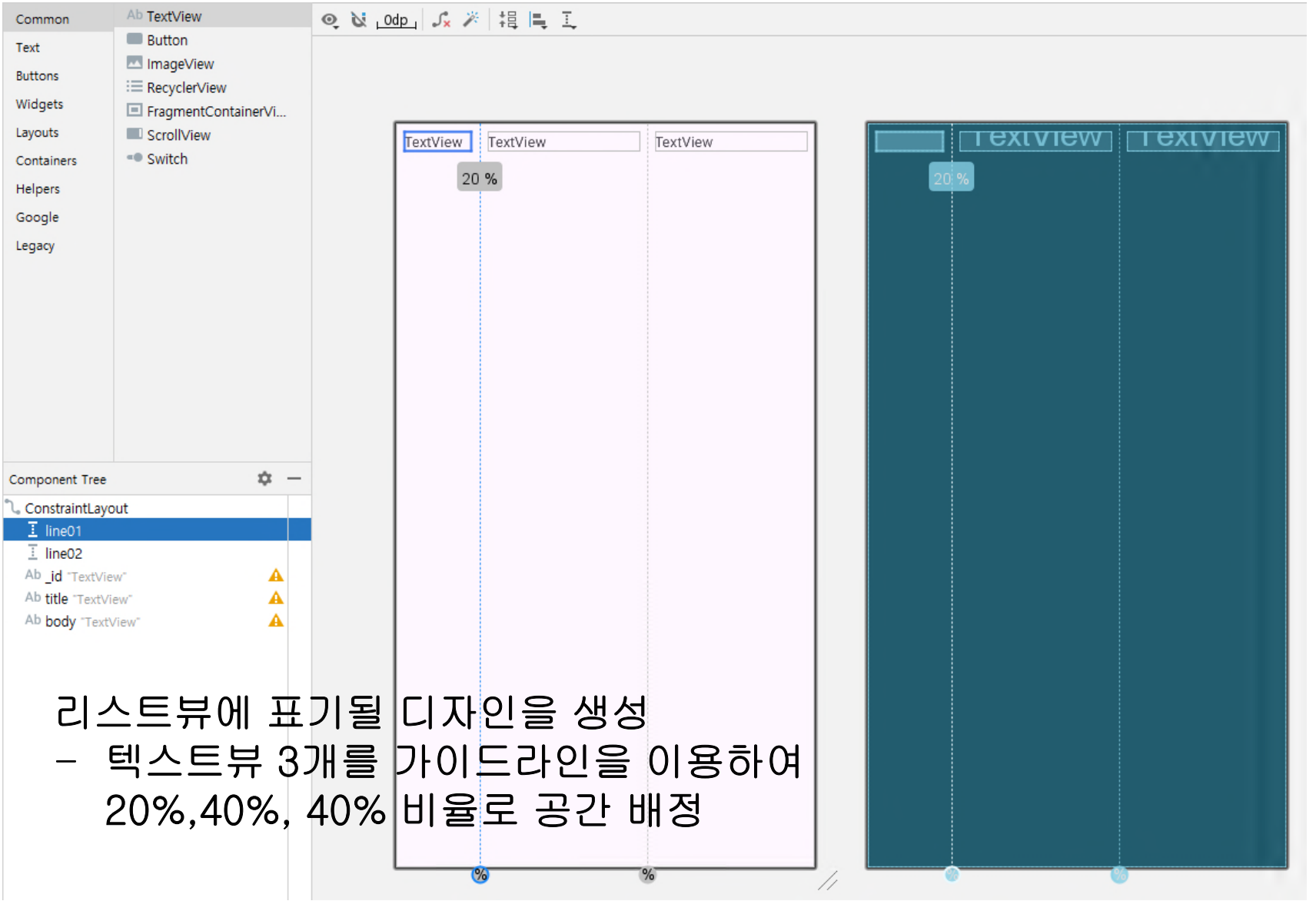
❑ 레이아웃 - activity_main.xml

```
    <Button
        android:id="@+id/button02"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="3dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="3dp"
        android:text="Update"
        android:textAllCaps="false"
        app:layout_constraintEnd_toStartOf="@+id/ver02"
        app:layout_constraintStart_toStartOf="@+id/ver01"
        app:layout_constraintTop_toBottomOf="@+id/table01" />

    <Button
        android:id="@+id/button03"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="3dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="3dp"
        android:text="Delete"
        android:textAllCaps="false"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="@+id/ver02"
        app:layout_constraintTop_toBottomOf="@+id/table01" />
```

❑ 레이아웃 - activity_main.xml

```
    <ListView
        android:id="@+id/list01"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button02" />

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

❑ 레이아웃 - data_row.xml



리스트뷰에 표기될 디자인을 생성

- 텍스트뷰 3개를 가이드라인을 이용하여 20%,40%, 40% 비율로 공간 배정

❍ 데이터 출력을 위한 레이아웃 (ListView에서 이용)

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

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/line01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.20" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/line02"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.60" />
```

❑ 레이아웃 - data_row.xml

```
<TextView
    android:id="@+id/_id"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:text="TextView"
    app:layout_constraintEnd_toStartOf="@+id/line01"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/title"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:text="TextView"
    app:layout_constraintEnd_toStartOf="@+id/line02"
    app:layout_constraintStart_toStartOf="@+id/line01"
    app:layout_constraintTop_toTopOf="parent" />
```

❑ 레이아웃 - data_row.xml

```
    <TextView
        android:id="@+id/body"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:text="TextView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="@+id/line02"
        app:layout_constraintTop_toTopOf="parent" />

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

❑ 데이터베이스 생성

```
private static class DatabaseHelper extends SQLiteOpenHelper {
   public  DatabaseHelper(Context context) {
       super(context, DATABASE_NAME, null, DATABASE_VERSION);
   }
```

- 데이터베이스의 오픈을 실행하기 위해 SQLiteOpenHelper를 상속받아 작성
- 생성자에서 데이터베이스 파일명과 버전을 지정

```
super(context, DATABASE_NAME, null, DATABASE_VERSION);
```

- Context 인스턴스, 데이터베이스 파일명, 커서 팩토리, 데이터베이스 스키마 버전

❑ onCreate()

```
public void onCreate(SQLiteDatabase db) {
    //테이블을 새로 작성
    db.execSQL( CREATE_TABLE );
  }
```

- SQLiteOpenHelper의 서브클래스에서 onCreate() 메소드를 오버라이딩
- 데이터베이스가 존재하지 않는 상태에서 데이터베이스를 오픈하려고 하는 경우 onCreate() 메소드가 호출
- execSQL 메소드를 사용하여 테이블을 작성하거나 초기값 설정 등을 실행
- 테이블 생성 예
  - create table notes(_id integer primary key autoincrement, title text not null, body text not null);

❑ onUpgrade()

```
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
    Log.w(TAG, "Upgrading database from version " + oldVersion + " to "
        + newVersion + ", which will destroy all old data");
    db.execSQL("DROP TABLE IF EXISTS notes");
    onCreate(db);
  }
```

- 데이터베이스의 버전관리
- 테이블의 구조 변경 등이 발생한 경우

❑ Insert()

- 데이터를 조작하기 위한 방법들 가운데 하나
- Insert()
  - 대상이 되는 테이블 이름, null 처리 컬럼명, ContentValues 객체에 컬럼별 값을 넣어 인자로 넘김

```
Public long insert(테이블 명, null 처리 컬럼명, 컬럼별 값을 넣은 ContentValues 객체);
```

```
public long insertNote(String title, String body) {
   ContentValues initialValues = new ContentValues();
   initialValues.put(KEY_TITLE, title);
   initialValues.put(KEY_BODY, body);
   return mDb.insert(DATABASE_TABLE, null, initialValues);
 }
```

❑ Update()

- 대상 테이블 이름과 변경할 값이 들어 있는 Contentvalues 객체를 넘김
- 값을 변경할 대상을 한정지으려면 WHERE 구문과 함께 조건에 해당하는 값을 넘김

Public int update(테이블 명, Contentvalues 객체, WHERE 구문, WHERE 조건에 해당하는 값)

```
public boolean updateNote(String rowId, String title, String body) {
    ContentValues args = new ContentValues();
    args.put(KEY_TITLE, title);
    args.put(KEY_BODY, body);
    return mDb.update(DATABASE_TABLE, args, KEY_ROWID + "=" + rowId,
null) > 0;
  }
```

❑ Delete()

- 데이터 삭제에 사용
- 테이블 이름, WHERE 구문 사용

Public int delete(테이블 명, WHERE 구문, WHERE 조건에 해당하는 값)

```
public boolean deleteNote(String rowId) {
    Log.i("Delete called", "value__" + rowId);
    return mDb.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId, null) >
0;
  }
```

❑ 코드 분석 - DbAdapter

```
package com.practice.ex.sqlite;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class DbAdapter {  5 usages
    private static final String DATABASE_NAME = "data";  1 usage
    private static final int DATABASE_VERSION = 3;  1 usage
    private static final String DATABASE_TABLE = "notes";  4 usages

    static final String KEY_ROWID = "_id";  3 usages
    static final String KEY_TITLE = "title";  4 usages
    static final String KEY_BODY = "body";  4 usages
```

컬럼 이름 정의

❑ 코드 분석 - DbAdapter

```
private static final String CREATE_TABLE =  1 usage
        "create table notes (_id integer primary key autoincrement, "
                + "title text not null, body text not null);";
```

데이터베이스 초기화에 필요한 SQL 문장

```
private static final String TAG = "DbAdapter";  1 usage
```

```
private SQLiteDatabase mDb;  5 usages
```

데이터베이스 인스턴스
-실제 데이터베이스에 접근할 때 사용하는 인스턴스
-해당 인스턴스에서 Insert(), update(), query(), delete() 등의 메소드를 호출

```
private DatabaseHelper mDbHelper;  3 usages
```

DatabaseHelper의 인스턴스
- 데이터베이스를 열고 닫는 것을 도와주는 클래스를 이용하기 위함

```
private final Context mCtx;  2 usages

public DbAdapter(Context ctx) {  1 usage
    this.mCtx = ctx;
}
```

외부 클래스 생성자

❑ 코드 분석 - DbAdapter

```
private static class DatabaseHelper extends SQLiteOpenHelper {   2 usages
```

내부클래스

```
public DatabaseHelper(Context context) {   1 usage
    super(context, DATABASE_NAME,  factory: null, DATABASE_VERSION);
}
```

데이터베이스의 파일명과 버전을 지정

생성자

```
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL( CREATE_TABLE );
}
```

SQL 명령어를 이용하여 테이블을 새로 작성

데이터베이스 생성

```
@Override   no usages
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    Log.w(TAG,  msg: "Upgrading database from version " + oldVersion + " to "
            + newVersion + ", which will destroy all old data");
    db.execSQL("DROP TABLE IF EXISTS notes");
    onCreate(db);
}
```

데이터베이스가
업그레이드될 필요가 있을 때 호출

```
}
```

SQLiteOpenHelper
- 데이터베이스 열기/닫기 지원
- 데이터베이스 생성, 업그레이드

데이터베이스 업그레이드
- 필요에 따라 추가 구현

❑ 코드 분석 - DbAdapter

```
public DbAdapter open() throws SQLException {  1 usage
    mDbHelper = new DatabaseHelper(mCtx);
    mDb = mDbHelper.getWritableDatabase();
    return this;
}

public void close() {  1 usage
    mDbHelper.close();
}
```

- getWritableDatabase()
  읽기, 쓰기 모드로 데이터베이스 오픈
  처음 호출되는 경우에는 onCreate()가 호출

databaseHelper 부분 참조
해당 인스턴스를 이용하여 데이터베이스를 열고 닫음

❑ 코드 분석 - DbAdapter

```
    public long insertNote(String title, String body) {  1 usage
        ContentValues initialValues = new ContentValues();
        initialValues.put(KEY_TITLE, title);
        initialValues.put(KEY_BODY, body);
        return mDb.insert(DATABASE_TABLE,  nullColumnHack: null, initialValues);
    }

    public Cursor fetchAllNotes() {  2 usages
        return mDb.query(DATABASE_TABLE, new String[] { KEY_ROWID, KEY_TITLE,
                KEY_BODY },  selection: null,  selectionArgs: null,  groupBy: null,  having: null,  orderBy: "_id DESC");
    }

    public boolean updateNote(String rowId, String title, String body) {  1 usage
        ContentValues args = new ContentValues();
        args.put(KEY_TITLE, title);
        args.put(KEY_BODY, body);
        return mDb.update(DATABASE_TABLE, args,  whereClause: KEY_ROWID + "=" + rowId,  whereArgs: null) > 0;
    }

    public boolean deleteNote(String rowId) {  1 usage
        Log.i( tag: "Delete called",  msg: "value__" + rowId);
        return mDb.delete(DATABASE_TABLE,  whereClause: KEY_ROWID + "=" + rowId,  whereArgs: null) > 0;
    }
}
```

데이터를 다루기 위한 방법
- execSQL()
- 기본 제공 메소드

❑ rawQuery(), query() 메소드

- rawQuery()
  - 메소드를 생성해 select 구문을 직접 실행
- query()
  - 메소드를 인자로 각 부분의 값을 넘겨 실행

```
Cursor c = db.rawQuery("SELECT * FROM sqlite_master WHERE type =
'table' AND name = 'constants'", null);
```

```
Cursor mCursor = mDb.query
        (DATABADE_TABLE, // 대상 테이블
        culumns, // 가져올 칼럼 이름의 배열
        "_id=?", //WHERE 구문
        parms, // WHERE 구문에 들어가는 매개변수 값
        null, // GROUP BY 구문 …
        null, // HAVING 구문
        null // ORDER BY 구문 …
);
```

❑ 코드 분석 - MainActivity

```
package com.practice.ex.sqlite;

import android.database.Cursor;
import android.os.Bundle;
import android.provider.BaseColumns;
import android.view.View;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.cursoradapter.widget.SimpleCursorAdapter;

public class MainActivity extends AppCompatActivity implements View.OnClickListener,
                                                    AdapterView.OnItemClickListener {

    private DbAdapter dbAdapter;  8 usages
    private SimpleCursorAdapter mAdapter;  4 usages
    private EditText eText1, eText2, eText3;  6 usages
    private Button insert_btn, update_btn, delete_btn;  2 usages
    ListView list01;  3 usages
```

❑ 코드 분석 - MainActivity

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_main);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {...});

    eText1 = (EditText)findViewById(R.id.editText01);
    eText2 = (EditText)findViewById(R.id.editText02);
    eText3 = (EditText)findViewById(R.id.editText03);
    eText1.setEnabled(false);
    insert_btn = (Button)findViewById(R.id.button01);
    update_btn = (Button)findViewById(R.id.button02);
    delete_btn = (Button)findViewById(R.id.button03);
    insert_btn.setOnClickListener(this);
    update_btn.setOnClickListener(this);
    delete_btn.setOnClickListener(this);

    list01 = (ListView)findViewById(R.id.list01);
    list01.setOnItemClickListener(this);
}
```

레이아웃 객체와 연결

첫번째 EditText(tv)는 id가 자동으로 들어갈 부분이므로 비활성화

레이아웃 객체와 연결

리스너 등록

❑ 코드 분석 - MainActivity

```
@Override
protected void onResume() {
    super.onResume();

    dbAdapter = new DbAdapter( ctx: this);
    dbAdapter.open();
    Cursor c = dbAdapter.fetchAllNotes();

    String[] from = new String[] {BaseColumns._ID,
            DbAdapter.KEY_TITLE,
            DbAdapter.KEY_BODY
    };

    int[] to = new int[] { R.id._id, R.id.title, R.id.body};

    mAdapter = new SimpleCursorAdapter(
            context: this, R.layout.data_row, c, from, to, flags: 0);

    list01.setAdapter(mAdapter);
}
```

커서는 쿼리 결과물을 담고 있는 것으로,
필요할 때마다 그 결과값을 손쉽게 가져올 수 있는 객체

데이터베이스를 오픈함

모든 데이터에 대한 커서를 얻어옴

리스트뷰에 데이터베이스의 저장된 데이터를 연결

ListView에 대한 부분 참조

```
this        // Context
R.layout.note_row   // 한 행을 보여줄 레이아웃
c       // 어댑터에게 커서를 연결
from    // 어댑터에게 커서의 컬럼을 연결
to      // 한 행을 보여줄 XML 파일의 텍스트 뷰 id를 저장한 배열
flags   // Adapter 상태
```

❑ 코드 분석 - MainActivity

```java
@Override
protected void onPause() {
    super.onPause();
    dbAdapter.close();
}
```

필요시 기능 추가 구현

```java
@Override
public void onClick(View view) {
    String txt1, txt2, txt3;

    if(view.getId() == R.id.button01) {
        txt2 = eText2.getText().toString().trim();
        txt3 = eText3.getText().toString().trim();
        if(!txt2.isEmpty() && !txt3.isEmpty()){
            dbAdapter.insertNote(txt2, txt3);
            toastMemo( str: "insert - success");
        }else{
            toastMemo( str: "insert - failure");
        }
    }
```

각 기능에 따른 메소드를 호출하여 처리

❑ 코드 분석 - MainActivity

```
if(view.getId() == R.id.button02) {
    txt1 = eText1.getText().toString().trim();
    txt2 = eText2.getText().toString().trim();
    txt3 = eText3.getText().toString().trim();
    if(!txt2.isEmpty() && !txt3.isEmpty()){
        if(dbAdapter.updateNote(txt1, txt2, txt3))
            toastMemo( str: "update - success");
        else
            toastMemo( str: "update - failure");
    }
}
if(view.getId() == R.id.button03) {
    txt1 = eText1.getText().toString().trim();
    if(dbAdapter.deleteNote(txt1))
        toastMemo( str: "delete - success");
    else
        toastMemo( str: "delete - failure");
}

eText1.setText("");
eText2.setText("");
eText3.setText("");
setEnabled(false);
Cursor c = dbAdapter.fetchAllNotes();
mAdapter.changeCursor(c);
mAdapter.notifyDataSetChanged();
}
```

각 기능에 따른 메소드를 호출하여 처리

갱신된 커서를 이용해 리스트 갱신

❑ 코드 분석 - MainActivity

```
@Override
public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
    ConstraintLayout container = (ConstraintLayout) view;
    TextView id_text = (TextView)container.findViewById(R.id._id);
    TextView title_text = (TextView)container.findViewById(R.id.title);
    TextView body_text = (TextView) container.findViewById(R.id.body);
    eText1.setText(id_text.getText());
    eText2.setText(title_text.getText());
    eText3.setText(body_text.getText());
    setEnabled(true);
}
```

리스트뷰의 항목이 선택되었을 때
선택된 항목의 내용을
상단의 EditText에 표시하기 위한 부분

관련 내용은 ListView 참조

```
private void setEnabled(boolean enabled) {  2 usages
    update_btn.setEnabled(enabled);
    delete_btn.setEnabled(enabled);
}
```

아이템이 선택된 경우에만 갱신, 삭제 작업을 수행, 따라서 버튼 활성화

- 버전에 따라 보이는 모습이 일부 차이가 있을 수 있음

```
private void toastMemo(String str) {  6 usages
    if(str.isEmpty())
        return;
    Toast toast = Toast.makeText(getApplicationContext(), str, Toast.LENGTH_LONG);
    toast.show();
}

}
```

❑ 참고

- SQLite 예제를 동작시키고 수정 과정에서 실행에 문제가 있는 경우,
  - 코드 작성에서 오류가 없이 정상적인 진행이 되었을 경우, 데이터베이스 갱신이 이루어지지 않아 문제가 될 수 있음
  - 데이터베이스 버전을 통한 갱신을 수행해야 하지만 구현 시에는 간단히 기존 데이터베이스를 삭제하여 수행해도 됨
  - 조치 방법
    - ❖ 에뮬레이터 실행 후에 Device File Explorer 선택
    - ❖ Data → Data → 해당 프로젝트 선택 (예, com.practice.ex.sqlite)
    - ❖ 해당 프로젝트의 databases 아래 파일을 삭제 후 실행
      - 삭제는 우클릭 후 Delete 명령 수행