What are Preferences? Discuss using example how to work with the Preferences.

- Android provides many ways of storing data of an application.
- One of this way is called Shared Preferences. Shared Preferences allow you to save and retrieve data in the form of key, value pair.

Creating SharedPreference

- In order to use shared preferences, you have to call a method getSharedPreferences() that returns a SharedPreference instance pointing to the file that contains the values of preferences.

```
var sp =
application.getSharedPreferences("STUDENT",
Context.MODE_PRIVATE)
```

- The first parameter is the key and the second parameter is the MODE.
- Mode can be :-
  - MODE_APPEND :- This will append the new preferences with the already existing preferences.
  - MODE_PRIVATE :- By setting this mode, the file can only be accessed using calling application.

Saving Shared Preference

- You can save something in the shared preferences by using SharedPreferences.Editor class.
- You will call the edit() method of Shared Preference instance and will receive it in an editor object.

Example

```
var editor = sp.edit()

editor.putString("name", "RAM");
editor.putString("email", "RAM@gmail.com");
editor.putString("city", "Rajkot");
editor.commit();
```

- Apart from the putString() method , there are methods available in the editor class that allows manipulation of data inside shared preferences.

Retrieving Shared Preference

- You can retrieve something from the sharedpreferences by using **getString()** method.

  Example
  ```
  sp.getString("name","");
  sp.getString("email","");
  sp.getString("city","");
  ```

Methods of SharedPreference.editor Class :-

Followings are **methods** available in the **editor** class that allows manipulation of data inside shared preferences.

- **clear()** :- It will remove all values from the editor
- **remove(String key)** :- It will remove the value whose key has been passed as a parameter.
- **putLong(String key, long value)** :- It will save a long value in a preference editor
- **putInt(String key, int value)** :- It will save a integer value in a preference editor
- **putFloat(String key, float value)** :- It will save a float value in a preference editor

SQLite Database

- SQLite is a well-regarded relational database management system (RDBMS).
- It is:
  - Open-source
  - Standards-compliant
  - Lightweight
  - Single-tier
- Using SQLite you can create fully encapsulated relational databases for your applications.
- Use them to store and manage complex, structured application data.
- Android databases are stored in the **/data/data/<package name>/databases folder on your device (or emulator).**
- All databases are private, accessible only by the application that created them.
- Database design is a big topic that deserves more thorough coverage than is possible within this book.
- It is worth highlighting that standard database best practices still apply in Android.

Helper Class

- Using helper class, we can create the database, tables and we can insert the records too.
- Using helper class, we can access the database in any activity.

```
import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class MyDBHelper(context:Context?) : SQLiteOpenHelper(context, name: "EMPDB", factory: null, version: 1) {
    override fun onCreate(db: SQLiteDatabase?) {
        db?.execSQL( sql: "CREATE TABLE EMP(EMPNO INTEGER PRIMARY KEY AUTOINCREMENT,ENAME TEXT, ESAL INTEGER)")
        db?.execSQL( sql: "INSERT INTO EMP(ENAME,ESAL) VALUES('TATSAT SHUKLA',25000)")
        db?.execSQL( sql: "INSERT INTO EMP(ENAME,ESAL) VALUES('HARIOM',21000)")
        db?.execSQL( sql: "INSERT INTO EMP(ENAME,ESAL) VALUES('PARTH SWADAS',15000)")
    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {
    }
}
```

To Initializer Helper and SQLiteDatabase Instance

```
// Initialize helper and db instance
var helper = MyDBHelper(applicationContext)
var db : SQLiteDatabase! = helper.readableDatabase
```

After initialization of database instance we can retrieve the data and can perform CRUD operations.

Initialization of Cursor Variable and Access First Record

```
//Select Data and Display First Record
var rs : Cursor!  = db.rawQuery("SELECT * FROM EMP",null)
if(rs.moveToNext()) {
    editText1.setText(rs.getString(0))
    editText2.setText(rs.getString(1))
    editText3.setText(rs.getString(2))
}
else
    Toast.makeText(applicationContext, text: "Record Not Found", Toast.LENGTH_LONG).show()
```

INSERT in SQLiteDatabase

```
//Insert
button5.setOnClickListener { it: View!
        var cv = ContentValues()
        cv.put("ENAME", editText2.text.toString())
        cv.put("ESAL", editText3.text.toString())
        db.insert( table: "EMP",  nullColumnHack: null, cv)
}
```

UPDATE in SQLiteDatabase

```
//UPDATE
button6.setOnClickListener { it: View!
    var cv = ContentValues()
    cv.put("ENAME",editText2.text.toString())
    cv.put("ESAL",editText3.text.toString())
    db.update( table: "EMP",cv, whereClause: "EMPNO = ?", arrayOf(editText1.text.toString()))
    rs.requery()
}
```

DELETE in SQLiteDatabase

```
//DELETE
button7.setOnClickListener { it: View!
    db.delete( table: "EMP", whereClause: "EMPNO = ?", arrayOf(editText1.text.toString()))
    rs.requery()
}
```

To Get First and Next Record from SQLite Database

cursor.moveToFirst() function is used to get first record.
Cursor.moveToNext() function is used to get next record.

```
//First
button1.setOnClickListener { it: View!
    if(rs.moveToFirst()) {
        editText1.setText(rs.getString(0))
        editText2.setText(rs.getString(1))
        editText3.setText(rs.getString(2))
    } else
        Toast.makeText(applicationContext, text: "Record Not Found",Toast.LENGTH_LONG).show()
}

//Next
button2.setOnClickListener { it: View!
    if(rs.moveToNext()) {
        editText1.setText(rs.getString(0))
        editText2.setText(rs.getString(1))
        editText3.setText(rs.getString(2))
    }
    else if(rs.moveToFirst()) {
        editText1.setText(rs.getString(0))
        editText2.setText(rs.getString(1))
        editText3.setText(rs.getString(2))
    }
    else
        Toast.makeText(applicationContext, text: "Record Not Found",Toast.LENGTH_LONG).show()
}
```

To Get Last and Previous Record

```
//Previous
button3.setOnClickListener { it: View!
    if(rs.moveToPrevious()) {
        editText1.setText(rs.getString(0))
        editText2.setText(rs.getString(1))
        editText3.setText(rs.getString(2))
    }
    else if(rs.moveToLast()) {
        editText1.setText(rs.getString(0))
        editText2.setText(rs.getString(1))
        editText3.setText(rs.getString(2))
    }
    else
        Toast.makeText(applicationContext, text: "Record Not Found",Toast.LENGTH_LONG).show()
}

//Last
button4.setOnClickListener { it: View!
    if(rs.moveToLast()) {
        editText1.setText(rs.getString(0))
        editText2.setText(rs.getString(1))
        editText3.setText(rs.getString(2))
    } else
        Toast.makeText(applicationContext, text: "Record Not Found",Toast.LENGTH_LONG).show()
}
```

To Search for Specific Record

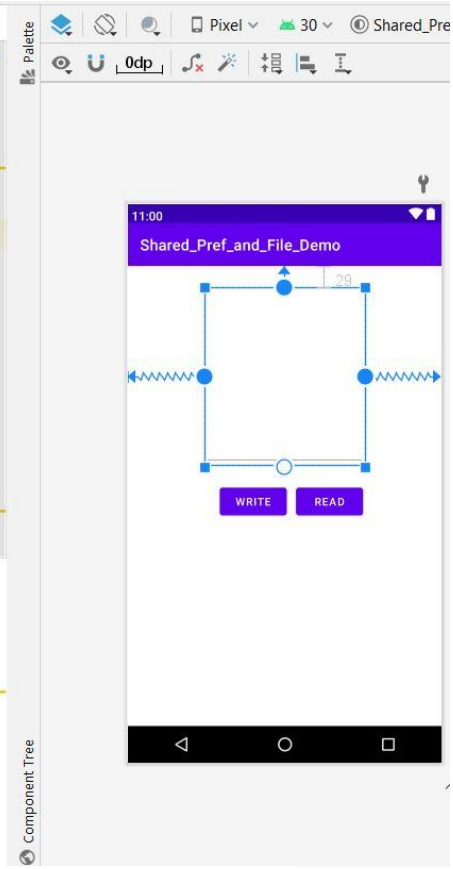
```
//Searching
button8.setOnClickListener { it: View!
        var rs1 : Cursor!  = db.rawQuery("SELECT * FROM EMP WHERE EMPNO = ?",
                                         arrayOf(editText1.text.toString()))
        if (rs1.moveToNext()) {
            editText1.setText(rs1.getString(0))
            editText2.setText(rs1.getString(1))
            editText3.setText(rs1.getString(2))
        } else
            Toast.makeText(
                applicationContext,
                text: "Record Not Found " + editText1.text.toString(),
                Toast.LENGTH_LONG
            ).show()
}
```

To Retrieve All Records and bind it in ListView using SimpleCursorAdapter

```
var from : Array<String>  = arrayOf("ENAME","ESAL")
var to : IntArray  = intArrayOf(android.R.id.text1,android.R.id.text2)


var helper = MyDBHelper(applicationContext)
var db : SQLiteDatabase!  = helper.readableDatabase
var rs : Cursor!  = db.rawQuery("SELECT EMPNO _id, ENAME,ESAL FROM EMP",null)

var adapter = SimpleCursorAdapter(applicationContext,
    android.R.layout.simple_list_item_2,
    rs,
    from,
    to, flags: 0)
listview.adapter = adapter
```

What is File? Explain how to create file, write into file and how to read from file with example?

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http:
    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=".FileActivity">
    <EditText
        android:id="@+id/editTextTextMultiLine"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="29dp"
        android:ems="10"
        android:gravity="start|top"
        android:inputType="textMultiLine"
        android:lines="10"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="17dp"
        android:layout_marginTop="22dp"
        android:layout_marginEnd="12dp"
        android:text="Write"
        app:layout_constraintEnd_toStartOf="@+id/button4"
```




```
class FileActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_file)

        var ed1 = findViewById<EditText>(R.id.editTextTextMultiLine)
        var b1 = findViewById<Button>(R.id.button3)
        var b2 = findViewById<Button>(R.id.button4)
        b1.setOnClickListener { it: View!
            var fos = openFileOutput( name: "myfile",Context.MODE_PRIVATE)
            fos.write(ed1.text.toString().toByteArray())
        }
        b2.setOnClickListener { it: View!
            var fin = openFileInput( name: "myfile")
            var br = BufferedReader(InputStreamReader(fin))
            var line : String? = ""
            while(line!=null){
                line = br.readLine()
                if(line!=null)
                    ed1.append(line+"\n")
            }

        }
    }
}
```

What is Content Provider? What are the Built-in Content Providers? Explain Call Log Content Provider with example.

- A content provider manages access to a central repository of data.
- A provider is part of an Android application, which often provides its own UI for working with the data.
- However, content providers are primarily intended to be used by other applications, which access the provider using a provider client object.
- Typically you work with content providers in one of two scenarios; you may want to implement code to access an existing content provider in another application, or you may want to create a new content provider in your application to share data with other applications.

Built-in Content Provider:-

- CallLog
- ContactsContract
- MediaStore
- Browser
- Calendar

Contact Content Provider:-

```
var cols = arrayOf(
ContactsContract.CommonDataKinds.Phone
.DISPLAY_NAME,
ContactsContract.CommonDataKinds.Phone
.NUMBER,
ContactsContract.CommonDataKinds.Phone
._ID)


var from =
arrayOf(ContactsContract.CommonDataKin
ds.Phone.DISPLAY_NAME,

ContactsContract.CommonDataKinds.Phone
.NUMBER)

var to =
intArrayOf(android.R.id.text1,
android.R.id.text2)

var rs =
contentResolver.query(ContactsContract
.CommonDataKinds.Phone.CONTENT_URI,
cols,null,null,
ContactsContract.CommonDataKinds.Phone
.DISPLAY_NAME)

var adapter =
SimpleCursorAdapter(this,android.R.lay
out.simple_list_item_2,
rs,from,to,0)
listview1.adapter = adapter
```

CallLog Content Provider :-

Fields:

```
var cols= arrayOf(CallLog.Calls._ID,
CallLog.Calls.NUMBER,
CallLog.Calls.TYPE,
CallLog.Calls.DURATION)
```

Content URI:

```
CallLog.Calls.CONTENT_URI,
```

MediaStore Content Provider:-

Field:

```
MediaStore.Audio.AudioColumns._ID,
MediaStore.Audio.AudioColumns.ALBUM,
MediaStore.Audio.AudioColumns.TITLE,
MediaStore.Audio.AudioColumns.ARTIST
```

Content Uri :

MediaStore.Audio.Media.External_CONTENT_URI

Related Permissions :-

```
<uses-permission
android:name="android.permission.READ_CALL_LOG">
</uses-permission>

<uses-permission
android:name="android.permission.READ_CONTACTS"
></uses-permission>

<uses-permission
android:name="android.permission.READ_EXTERNAL_
STORAGE"/>
```

Example – Contact Content Provider

```
class MainActivity : AppCompatActivity() {
    var cols : Array<String>  = arrayOf(
                        ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME,
                        ContactsContract.CommonDataKinds.Phone.NUMBER,
                        ContactsContract.CommonDataKinds.Phone._ID
    )
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        if(ActivityCompat.checkSelfPermission( context: this,Manifest.permission.READ_CONTACTS)
                        != PackageManager.PERMISSION_GRANTED){
            ActivityCompat.requestPermissions( activity: this,
                                               arrayOf(Manifest.permission.READ_CONTACTS),
                                               requestCode: 111)
        }
        else
            readContact()
    }

    override fun onRequestPermissionsResult(requestCode: Int,
                                            permissions: Array<out String>,
                                            grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults)
        if(requestCode==111 && grantResults[0] == PackageManager.PERMISSION_GRANTED)
            readContact()
    }
```

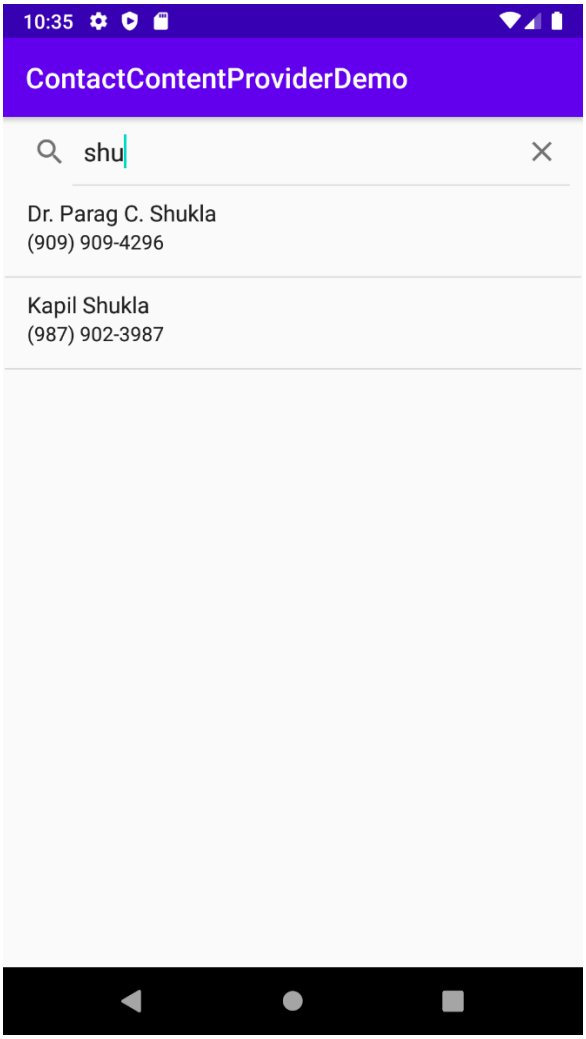
```
    private fun readContact() {
        var from : Array<String>  = arrayOf(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME,
            ContactsContract.CommonDataKinds.Phone.NUMBER)
        var to : IntArray  = intArrayOf(android.R.id.text1, android.R.id.text2)
        var rs : Cursor?  = contentResolver.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
                                                  cols, selection: null, selectionArgs: null,
                                                   ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME)
        var adapter = SimpleCursorAdapter( context: this,android.R.layout.simple_list_item_2,
                                           rs,from,to, flags: 0)
        listview1.adapter = adapter
        searchView.setQueryHint("${rs?.count} Contacts")
```

```
        searchView.setOnQueryTextListener(object: SearchView.OnQueryTextListener{
            override fun onQueryTextSubmit(p0: String?): Boolean {
                    return false
            }

            override fun onQueryTextChange(p0: String?): Boolean {
                rs = contentResolver.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
                    cols, selection: "${ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME} LIKE ?",
                    Array( size: 1){"%$p0%"},
                    ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME)
                adapter.changeCursor(rs)
                return false
            }
        })
    }
}
```

Design

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns: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">
    <SearchView
        android:id="@+id/searchView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="1dp"
        android:layout_marginTop="1dp"
        android:layout_marginEnd="1dp"
        android:iconifiedByDefault="false"
        android:layout_marginBottom="1dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0" />
    <ListView
        android:id="@+id/listview1"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="1dp"
        android:layout_marginEnd="1dp"
        android:layout_marginBottom="1dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/searchView" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Output



Contact Add Example

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    if(ActivityCompat.checkSelfPermission( context: this,
            Manifest.permission.WRITE_CONTACTS)!=PackageManager.PERMISSION_GRANTED)
    {
        ActivityCompat.requestPermissions( activity: this,
            arrayOf(Manifest.permission.WRITE_CONTACTS, Manifest.permission.READ_CONTACTS),
             requestCode: 111)
    }else{
        writeContact()
        readContacts()
    }


  private fun writeContact() {
      var cv = ContentValues()
      var rowUri = contentResolver.insert(ContactsContract.RawContacts.CONTENT_URI,cv)
      var rowContactId = ContentUris.parseId(rowUri!!)

      cv.put(ContactsContract.Data.RAW_CONTACT_ID, rowContactId)
      cv.put(ContactsContract.Data.MIMETYPE,
          ContactsContract.CommonDataKinds.StructuredName.CONTENT_ITEM_TYPE)
      cv.put(ContactsContract.CommonDataKinds.StructuredName.DISPLAY_NAME,"Anand Tank")
      contentResolver.insert(ContactsContract.Data.CONTENT_URI, cv)

      cv.put(ContactsContract.Data.RAW_CONTACT_ID, rowContactId)
      cv.put(ContactsContract.Data.MIMETYPE,
          ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE)
      cv.put(ContactsContract.CommonDataKinds.Phone.NUMBER, 9099090991)
      contentResolver.insert(ContactsContract.Data.CONTENT_URI, cv)
  }
```

Call Log Content Provider Example

```
class MainActivity : AppCompatActivity() {

    var cols : Array<String> = arrayOf(CallLog.Calls._ID,
                        CallLog.Calls.NUMBER,
                        CallLog.Calls.TYPE,
                        CallLog.Calls.DURATION)

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        if(ActivityCompat.checkSelfPermission( context: this,
                Manifest.permission.READ_CALL_LOG) != PackageManager.PERMISSION_GRANTED){
            ActivityCompat.requestPermissions( activity: this,
                arrayOf(Manifest.permission.READ_CALL_LOG), requestCode: 101)
        }
        else
            displayLog()
    }

    override fun onRequestPermissionsResult(requestCode: Int,
                                            permissions: Array<out String>,
                                            grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults)
        if(requestCode==101 && grantResults[0]==PackageManager.PERMISSION_GRANTED)
            displayLog()
    }

    private fun displayLog() {
        var from : Array<String>  = arrayOf(CallLog.Calls.NUMBER,
                CallLog.Calls.DURATION,
                CallLog.Calls.TYPE)
        var to : IntArray  = intArrayOf(R.id.textView1,R.id.textView2,R.id.textView3)
        var rs : Cursor? =  contentResolver.query(CallLog.Calls.CONTENT_URI,
                            cols, selection: null, selectionArgs: null,
                    sortOrder: "${CallLog.Calls.LAST_MODIFIED} DESC")
        var adapter = SimpleCursorAdapter(applicationContext,
                                          R.layout.mylayout,
                                           rs,
                                          from,
                                           to, flags: 0)
        listview.adapter = adapter
    }
}
```

activity_main.xml

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns
    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">

    <ListView
        android:id="@+id/listview"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="1dp"
        android:layout_marginTop="1dp"
        android:layout_marginEnd="1dp"
        android:layout_marginBottom="1dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

cardview – mylayout.xml

```
<androidx.cardview.widget.CardView xmlns:android="http:/
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    app:cardUseCompatPadding="true"
    app:cardCornerRadius="20dp"
    app:contentPadding="10dp"
    app:cardBackgroundColor="#BBFBBB"
    >
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
        <TextView
            android:id="@+id/textView1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="20dp"
            android:textStyle="bold"
            android:text="TextView" />
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="horizontal">
            <TextView
                android:id="@+id/textView2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="18dp"
                android:layout_weight="1"
                android:text="TextView" />
            <TextView
                android:id="@+id/textView3"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:textSize="18dp"
                android:text="TextView" />
        </LinearLayout>
    </LinearLayout>
```

Output

MediaStore – Images Content Provider Example – Load Images from SD Card

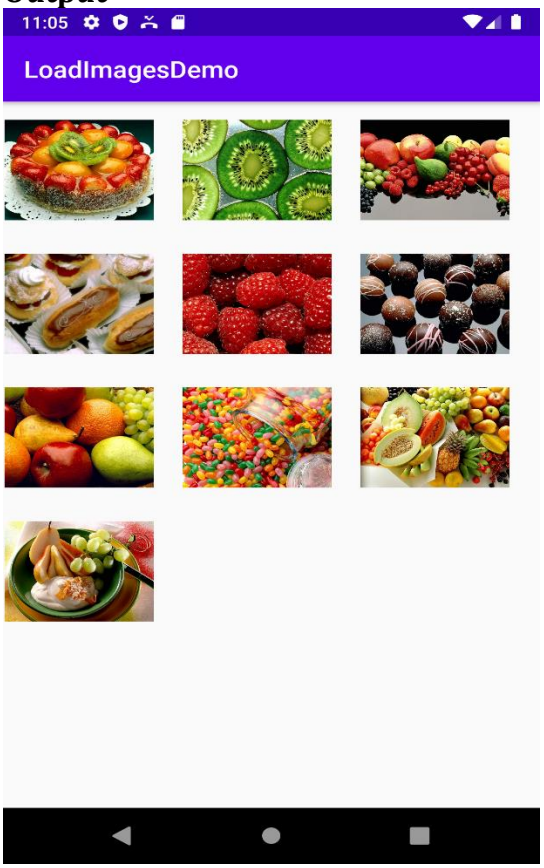
```
class MainActivity : AppCompatActivity() {
    lateinit var rs:Cursor
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        if(ActivityCompat.checkSelfPermission( context: this,
                Manifest.permission.READ_EXTERNAL_STORAGE) !=
                                    PackageManager.PERMISSION_GRANTED){
            ActivityCompat.requestPermissions( activity: this,
                arrayOf(Manifest.permission.READ_EXTERNAL_STORAGE),
                 requestCode: 121)
        }
        listImages()
    }

    override fun onRequestPermissionsResult(requestCode: Int,
                                            permissions: Array<out String>,
                                            grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults)
        if(requestCode == 121 && grantResults[0] == PackageManager.PERMISSION_GRANTED)
            listImages()
    }
    private fun listImages() {
        var cols : Array<String>  = arrayOf(MediaStore.Images.Thumbnails.DATA)
        rs = contentResolver.query(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
                cols, selection: null, selectionArgs: null, sortOrder: null)!!

        if(rs.moveToNext())
            Toast.makeText(applicationContext,rs.getString(0),Toast.LENGTH_LONG).show()

        gridview.adapter = ImageAdapter(applicationContext)
        gridview.setOnItemClickListener { adapterView, view, i, l ->
            rs.moveToPosition(i)
            var path : String!  = rs.getString(0)
            var i = Intent(applicationContext, DisplayImageActivity::class.java)
            i.putExtra( name: "path",path)
            startActivity(i)
        }
    }
```

```
    inner class ImageAdapter : BaseAdapter{
        lateinit var context:Context
        constructor(context: Context){
            this.context = context
        }
        override fun getView(p0: Int, p1: View?, p2: ViewGroup?): View {
            var iv = ImageView(context)
            rs.moveToPosition(p0)
            var path : String! = rs.getString(0)
            var bitmap : Bitmap! = BitmapFactory.decodeFile(path)
            iv.setImageBitmap(bitmap)
            iv.layoutParams = AbsListView.LayoutParams( w: 300, h: 300)
            return iv;
        }

        override fun getItem(p0: Int): Any {
            return p0
        }

        override fun getItemId(p0: Int): Long {
            return p0.toLong()
        }

        override fun getCount(): Int {
            return rs.count
        }
    }
}
```

Design

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns
    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">

    <GridView
        android:id="@+id/gridview"
        android:numColumns="3"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="1dp"
        android:layout_marginTop="1dp"
        android:layout_marginEnd="1dp"
        android:layout_marginBottom="1dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Output



How to create custom content provider explain it with steps.

Content URIs: To query a content provider, you specify the query string in the form of a URI which has following format

<prefix>://<authority>/<data_type>/<id>
Where,

- **prefix**: This is always set to content://
- **authority**: This specifies the name of the content provider
- **data_type**: This indicates the type of data that this particular provider provides.
- **id**: This specifies the specific record requested.

Creating Content Provider: This involves number of simple steps to create your own content provider.

- First of all you need to create a Content Provider class that extends the *ContentProvider* base class.
  Example*:*

class MyProvider extends ContentProvider { . . .}

- Second, you need to define your content provider URI address which will be used to access the content.
  Example:

```
companion object{
   val PROVIDER_NAME = "packagename/provider"
   val URL = "content://$PROVIDER_NAME/tablename"
   val CONTENT_URI = Uri.parse(URL)
}
```

- Next you will need to create your own database to keep the content.
- Usually, Android uses SQLite database and framework needs to override *onCreate()* method which will use SQLite Open Helper method to create or open the provider's database.
- When your application is launched, the *onCreate()* handler of each of its Content Providers is called on the main application thread.

Example:

```
class MyHelper(context: Context?) :
SQLiteOpenHelper(context,"MyDB",null,1){
   override fun onCreate(db: SQLiteDatabase?) {
         //table creation statement
   }
   override fun onUpgrade(p0: SQLiteDatabase?, p1:
Int, p2: Int) {
   }
}
```

- Next you will have to implement Content Provider queries to perform different database specific operations.

Example:

```
override fun onCreate(): Boolean {
         //your code
}
override fun query(
   uri: Uri, cols: Array<out String>?,
   condition: String?,   condition_val: Array<out
String>?,   order: String?): Cursor? {
         //your code
}
override fun insert(uri: Uri, cv: ContentValues?): Uri? {
         //your code
}
override fun update(uri: Uri, cv: ContentValues?,
condition: String?, condition_val: Array<out String>?):
Int {
         //your code
}
override fun delete(uri: Uri, condition: String?,
condition_val: Array<out String>?): Int {
         //your code
}
override fun getType(p0: Uri): String? {
         //your code
}
```

Finally register your Content Provider in your activity file using <provider> tag.

Example:

```
<provider
android:name="CustomContentProvider"
android:authorities="com.example.MyApplicatio
n.MyProvider"></provider>
```

Example Custom Content Provider

Step-1) Create the Helper Class

```
class MyHelper(context: Context?) : SQLiteOpenHelper(context, name: "MyDB", factory: null, version: 1){
    override fun onCreate(db: SQLiteDatabase?) {
        db?.execSQL( sql: "CREATE TABLE ACTABLE( _id INTEGER PRIMARY KEY AUTOINCREMENT, NAME TEXT,MEANING TEXT)");
        db?.execSQL( sql: "INSERT INTO ACTABLE(NAME,MEANING) VALUES('MCA','MASTER OF COMPUTER APPLICATIONS')");
        db?.execSQL( sql: "INSERT INTO ACTABLE(NAME,MEANING) VALUES('BCA','BACHLOR OF COMPUTER APPLICATIONS')");
        db?.execSQL( sql: "INSERT INTO ACTABLE(NAME,MEANING) VALUES('WWW','WORLD WIDE WEB')");
    }
    override fun onUpgrade(p0: SQLiteDatabase?, p1: Int, p2: Int) {
    }
}
```

Step-2) Create your own provider class which inherits from ContentProvider

```
class AcronymProvider : ContentProvider() {
    companion object{
        val PROVIDER_NAME = "pcs.mca.atmiya.customcontentproviderapp/AcronymProvider"
        val URL = "content://$PROVIDER_NAME/actable"
        val CONTENT_URI : Uri! = Uri.parse(URL)

        val _ID = "_id"
        val NAME = "NAME"
        val MEANING = "MEANING"

    }
    lateinit var db : SQLiteDatabase

    override fun onCreate(): Boolean {
        var helper = MyHelper(getContext())
        db = helper.writableDatabase
        return if (db == null) false else true
    }

     override fun query(
         uri: Uri,
         cols: Array<out String>?,
         condition: String?,
         condition_val: Array<out String>?,
         order: String?
    ): Cursor? {
            return db.query( table: "actable",cols,condition,condition_val,
                                groupBy: null, having: null,order)
    }
```

```
    override fun insert(uri: Uri, cv: ContentValues?): Uri? {
        db.insert( table: "actable", nullColumnHack: null,cv);
        getContext()?.getContentResolver()?.notifyChange(uri, observer: null);
        return uri;
    }

    override fun update(uri: Uri, cv: ContentValues?, condition: String?,
                        condition_val: Array<out String>?): Int {
        var count : Int  = db.update( table: "actable",cv,condition,condition_val)
        getContext()?.getContentResolver()?.notifyChange(uri, observer: null);
        return count
    }

    override fun delete(uri: Uri, condition: String?,
                        condition_val: Array<out String>?): Int {
        var count : Int  = db.delete( table: "actable",condition,condition_val)
        getContext()?.getContentResolver()?.notifyChange(uri, observer: null);
        return count
    }

    override fun getType(p0: Uri): String? {
        return "vnd.android.cursor.dir/vnd.example.actable";
    }
}
```

Step-3 Register Your Provider in AndroidManifest.xml

```
<provider
    android:authorities="pcs.mca.atmiya.customcontentproviderapp"
    android:name=".AcronymProvider"
    android:exported="true"
    android:grantUriPermissions="true"
    ></provider>
```

Step-4 Access Provider in MainActivity

To Insert Record using own content provider

```
val cv = ContentValues()
cv.put(AcronymProvider.NAME, editText.getText().toString())
cv.put(AcronymProvider.MEANING, editText2.getText().toString())
contentResolver.insert(AcronymProvider.CONTENT_URI, cv)
```

To Update Record using own content provider

```
val cv = ContentValues()
cv.put(AcronymProvider.MEANING, editText2.getText().toString())
val count : Int  = contentResolver.update(AcronymProvider.CONTENT_URI, cv,
                                  where: "NAME=?", arrayOf(editText.getText().toString()))
Toast.makeText( context: this,  text: "Record Updated :$count", Toast.LENGTH_SHORT).show()
```

To Delete Record using own content provider

```
val count : Int  = contentResolver.delete(AcronymProvider.CONTENT_URI,
     where: "NAME=?",arrayOf(editText.getText().toString()))
Toast.makeText( context: this,  text: "$count : Deleted", Toast.LENGTH_SHORT).show()
```

To Display Record using own content provider

```
var rs :Cursor?  = getContentResolver().query(AcronymProvider.CONTENT_URI,
        arrayOf(AcronymProvider._ID,AcronymProvider.NAME,AcronymProvider.MEANING),
        selection: null, selectionArgs: null, sortOrder: null);
while (rs?.moveToNext()!!)
    Toast.makeText( context: this,  text: rs.getString(1)+"\n"+rs.getString(2), Toast.LENGTH_SHORT).show();
```

Summary:

- ✓ This material covers Creating, saving and retrieving shares preferences.
- ✓ Introducing ANDROID SQLite database.
- ✓ Covered the Operation of SQLite Database like INSERT, UPDATE, DELETE, SEARCH, SELECT ALL
- ✓ Use of Content Values and Cursors
- ✓ Native Content Provider like
  - Contact Content Provider
  - Call Log Content Provider
  - MediaStore Image Content Provider
- ✓ Creating Custom/Own content provider
- ✓ Permission Used in this Unit
  - android.permission.READ_CALL_LOG
  - android.permission.READ_CONTACTS
  - android.permission.READ_EXTERNAL_STORAGE