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 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"/>