1. Illustrate in detail about SQLLite in Android

SQLite is an embedded database system in Android, which allows apps to store, retrieve, and manipulate structured data locally. When bypassing SQLiteOpenHelper, you manually handle database creation, table creation, and data operations without relying on helper methods for lifecycle events.

1. Create or Open a Database

You can create or open an existing database using SQLiteDatabase.openOrCreateDatabase(). Specify the database file path for the database.

```
// Create or open a database
SQLiteDatabase db = SQLiteDatabase.openOrCreateDatabase(
        “myDatabase.db", // Database Name
        null // No custom CursorFactory
);
```

2. Create a Table

Execute SQL commands directly to create a table. Use the execSQL() method for running raw SQL commands.

```
// Create a 'users' table
db.execSQL("CREATE TABLE IF NOT EXISTS users (" +
 "id INTEGER PRIMARY KEY, " +
 "name TEXT, " +
 "age INTEGER" +
 ");");
```

3. Insert Data

You can insert records into the database using either raw SQL queries or the ContentValues approach.

Using Raw SQL:

```
db.execSQL("INSERT INTO users (name, age) VALUES ('John Doe', 30);");
```

4. Read Data

To retrieve data, use the rawQuery() method. The returned data is processed via a Cursor object.

```
Cursor cursor = db.rawQuery("SELECT * FROM users", null); // Query to fetch all records

// Iterate over the cursor to process data
while (cursor.moveToNext()) {
        int id = cursor.getInt(cursor.getColumnIndex("id"));
        String name = cursor.getString(cursor.getColumnIndex("name"));
        int age = cursor.getInt(cursor.getColumnIndex("age"));
        System.out.println("ID: " + id + ", Name: " + name + ", Age: " + age);
}
cursor.close(); // Always close the Cursor
```

5. Update Data

Update specific rows in the table by using SQL commands or the update() method.

Using Raw SQL:

```
db.execSQL("UPDATE users SET age = 35 WHERE name = 'John Doe';");
```

6. Delete Data

Delete records using the execSQL() method or the delete() method.

```
db.execSQL("DELETE FROM users WHERE id = 1;");
```

7. Close the Database

Once you're done with database operations, close the database to free up resources.

```
db.close(); // Always close the database
```

2. **App that sends a reminder notification to the user to take a break after 2 hours of continuous work. The notification should have a custom layout with a "Dismiss"**

button. Explain how you would implement this functionality using notifications.

1. Setup the Notification System

You will use the NotificationCompat library to build and manage notifications. Create a custom layout for the notification and set up a timer to send a reminder after 2 hours.

2. Implementation Steps

Step 1: Create the Custom Layout

Design a custom layout XML for the notification in your res/layout folder. For example:

custom_notification.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="16dp">

        <TextView
 android:id="@+id/notification_title"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Time to take a break!"
 android:textStyle="bold"
 android:textSize="18sp" />

        <Button
 android:id="@+id/notification_dismiss"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_below="@id/notification_title"
 android:layout_marginTop="8dp"
 android:text="Dismiss" />
</RelativeLayout>
```

Step 2: Create the Reminder Notification

Use the NotificationCompat.Builder to create the notification, and link the custom layout.

```
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.os.Build;
import androidx.core.app.NotificationCompat;
import androidx.core.app.NotificationManagerCompat;

public class ReminderNotificationHelper {
        private static final String CHANNEL_ID = "break_reminder_channel";

        public static void showReminderNotification(Context context) {
 // Create NotificationChannel for Android 8.0+
 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
NotificationChannel channel = new NotificationChannel(
 CHANNEL_ID,
 "Break Reminder Notifications",
 NotificationManager.IMPORTANCE_HIGH
 );
 NotificationManager notificationManager =
 context.getSystemService(NotificationManager.class);
notificationManager.createNotificationChannel(channel);
 }

 // Create intent for dismiss button
 Intent dismissIntent = new Intent(context, DismissReceiver.class);
PendingIntent dismissPendingIntent = PendingIntent.getBroadcast(  context, 0,
dismissIntent, PendingIntent.FLAG_UPDATE_CURRENT |
PendingIntent.FLAG_IMMUTABLE
 );
 // Build notification with custom layout
 NotificationCompat.Builder builder = new NotificationCompat.Builder(context,
CHANNEL_ID)
```

```
 .setSmallIcon(android.R.drawable.ic_dialog_info)
 .setPriority(NotificationCompat.PRIORITY_HIGH)
 .setCustomContentView(createCustomLayout(context, dismissPendingIntent))
.setAutoCancel(true);

 // Show the notification
 NotificationManagerCompat notificationManagerCompat =
NotificationManagerCompat.from(context);
 notificationManagerCompat.notify(1, builder.build());
            }

            private static RemoteViews createCustomLayout(Context context, PendingIntent
dismissPendingIntent) {
 RemoteViews remoteViews = new RemoteViews(context.getPackageName(),
R.layout.custom_notification);
 remoteViews.setOnClickPendingIntent(R.id.notification_dismiss, dismissPendingIntent);
return remoteViews;
            }
}
```

Step 3: Handle the "Dismiss" Action

Implement a BroadcastReceiver to handle the dismiss button.

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.app.NotificationManager;

public class DismissReceiver extends BroadcastReceiver {
            @Override
            public void onReceive(Context context, Intent intent) {
 NotificationManager notificationManager =
 (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
notificationManager.cancel(1); // Cancel the notification
            }
}
```
Step 4: Schedule Reminder After 2 Hours

Use a Handler or AlarmManager to schedule the reminder after 2 hours of continuous work.

Example Using Handler:

```
import android.os.Handler;

public class ReminderScheduler {
        public static void scheduleReminder(Context context) {
 Handler handler = new Handler();
 handler.postDelayed(() ->
ReminderNotificationHelper.showReminderNotification(context), 2 * 60 * 60 * 1000); // 2 hours
        }
}
```

3. Demonstrate BroadcastReceiver and Notification in Android with examples

BroadcastReceiver

Definition

A BroadcastReceiver is a component in Android that responds to broadcast messages (intents) sent by the system or other applications. These broadcasts can be system-wide (like low battery alerts) or app-specific.

Steps to Implement BroadcastReceiver

1. Create a subclass of BroadcastReceiver.
2. Override the onReceive() method to handle the broadcast.
3. Register the receiver in the AndroidManifest.xml (static registration) or dynamically in your activity.

Example: Listening to Battery Level Changes

```
public class BatteryLevelReceiver extends BroadcastReceiver {
        @Override
        public void onReceive(Context context, Intent intent) {
```

```
 int level = intent.getIntExtra("level", -1);
 Toast.makeText(context, "Battery Level: " + level + "%",
Toast.LENGTH_SHORT).show();
            }
}
```

Register the Receiver Dynamically in Activity:

```
public class MainActivity extends AppCompatActivity {
            private BatteryLevelReceiver receiver;

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

 // Instantiate and register receiver
 receiver = new BatteryLevelReceiver();
 IntentFilter filter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
registerReceiver(receiver, filter);
            }

            @Override
            protected void onDestroy() {
 super.onDestroy();
 // Unregister the receiver
 unregisterReceiver(receiver);
            }
}
```

Notification

Definition

A notification is a small message that appears in the notification bar, informing users of events or actions in the app.

Steps to Implement Notification

1. Use NotificationCompat.Builder to create the notification.

2. Specify title, content, and icon for the notification.
3. Use NotificationManager to display it.

Example: Simple Notification

```
public void sendNotification(Context context) {
        // Create Notification Channel (Required for Android 8.0+)
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
 NotificationChannel channel = new NotificationChannel(
 "channel_id", "Channel Name",
NotificationManager.IMPORTANCE_DEFAULT);
 NotificationManager manager =
context.getSystemService(NotificationManager.class);
 manager.createNotificationChannel(channel);
        }

        // Build the notification
        NotificationCompat.Builder builder = new NotificationCompat.Builder(context,
"channel_id")
 .setSmallIcon(R.drawable.ic_notification)
 .setContentTitle("Hello, User!")
 .setContentText("This is a simple notification.")
 .setPriority(NotificationCompat.PRIORITY_DEFAULT);

        // Display the notification
        NotificationManagerCompat manager =
NotificationManagerCompat.from(context);
        manager.notify(1, builder.build());
}
```

Combined Use Case

You can combine BroadcastReceiver and Notification. For example, send a notification when the device's battery is low.

BroadcastReceiver for Low Battery:

```
public class LowBatteryReceiver extends BroadcastReceiver {
        @Override
        public void onReceive(Context context, Intent intent) {
 if (Intent.ACTION_BATTERY_LOW.equals(intent.getAction())) {  //
Send a notification
 NotificationCompat.Builder builder = new NotificationCompat.Builder(context,
"channel_id")
 .setSmallIcon(R.drawable.ic_battery)
 .setContentTitle("Battery Low!")
 .setContentText("Please charge your device.")
 .setPriority(NotificationCompat.PRIORITY_HIGH);

 NotificationManagerCompat manager = NotificationManagerCompat.from(context);
manager.notify(2, builder.build());
 }
        }
}
```

Register the Receiver in AndroidManifest.xml:

```
<receiver android:name=".LowBatteryReceiver">
        <intent-filter>
 <action android:name="android.intent.action.BATTERY_LOW" />
        </intent-filter>
</receiver>
```

4. Write the code to insert a user's name and email address into an SQLite database. The table structure is as follows: users (id INTEGER PRIMARY KEY, name TEXT, email TEXT) .

Steps:

1. **Database Creation and Table Structure**
    - Create or open the SQLite database.
    - Define the table structure (`users` table).
2. **Insert User Information**

   - Use raw SQL queries or `ContentValues` to insert user data into the database.
3. **View User Information**
   - Query the database using `rawQuery()`

```
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

        private SQLiteDatabase db;

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

 // Create or open the database
 db = SQLiteDatabase.openOrCreateDatabase("/data/data/" + getPackageName() +
"/userDatabase.db", null);

 // Create the users table
 db.execSQL("CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY, name
TEXT, email TEXT);");

 // Insert user data
 insertUser("John Doe", "john.doe@example.com");
 insertUser("Jane Smith", "jane.smith@example.com");

 // View user data
 viewUsers();
        }

        private void insertUser(String name, String email) {
 try {
 String insertQuery = "INSERT INTO users (name, email) VALUES ('" + name + "', '" + email
+ "');";
 db.execSQL(insertQuery);
 Toast.makeText(this, "User inserted: " + name, Toast.LENGTH_SHORT).show();  }
catch (Exception e) {
 Toast.makeText(this, "Error inserting user: " + e.getMessage(),
Toast.LENGTH_SHORT).show();
 }
            }

            private void viewUsers() {
 // Query the users table
 String query = "SELECT * FROM users;";
 try (Cursor cursor = db.rawQuery(query, null)) {
 if (cursor.getCount() > 0) {
 while (cursor.moveToNext()) {
 int id = cursor.getInt(cursor.getColumnIndex("id"));
 String name = cursor.getString(cursor.getColumnIndex("name"));  String email
= cursor.getString(cursor.getColumnIndex("email"));

 // Print or display the user data
 Toast.makeText(this, "ID: " + id + ", Name: " + name + ", Email: " + email,
Toast.LENGTH_LONG).show();
 }
 } else {
 Toast.makeText(this, "No users found.", Toast.LENGTH_SHORT).show();  }
 }
            }

            @Override
            protected void onDestroy() {
 super.onDestroy();
 // Close the database connection
 if (db != null && db.isOpen()) {
 db.close();
 }
            }
}
```

5. You are developing an Android application that listens for incoming SMS messages. When an SMS is received, the app should display a notification with the sender's phone number and the content of the message. Additionally, the app should log the received message and check if the message contains a specific keyword (e.g., "emergency"). If the keyword is found, the app should show a special notification indicating an urgent message.

1. Permissions

Add the following permissions to your `AndroidManifest.xml` for receiving SMS:

```
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.READ_SMS"/>
<uses-permission android:name="android.permission.POST_NOTIFICATIONS"/> <!-- For
notifications (Android 13+) -->
```

2. BroadcastReceiver for SMS

Create a `BroadcastReceiver` to listen for incoming SMS messages.

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.widget.Toast;
import androidx.core.app.NotificationCompat;
import androidx.core.app.NotificationManagerCompat;

public class SMSReceiver extends BroadcastReceiver {

        @Override
        public void onReceive(Context context, Intent intent) {
 // Retrieve SMS messages
 Bundle bundle = intent.getExtras();
 if (bundle != null) {
 Object[] pdus = (Object[]) bundle.get("pdus");
 if (pdus != null) {
 for (Object pdu : pdus) {
 SmsMessage message = SmsMessage.createFromPdu((byte[]) pdu);  String sender
= message.getOriginatingAddress();
 String content = message.getMessageBody();

 // Log the message
 logMessage(sender, content);
 // Check for the keyword
 if (content.toLowerCase().contains("emergency")) {
 sendNotification(context, "Urgent: " + sender, content, true);  } else {
 sendNotification(context, "Message from: " + sender, content, false);  }
 }
 }
 }
            }

            private void sendNotification(Context context, String title, String content, boolean
isUrgent) {
 // Create notification channel (Android 8.0+)
 if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.O) {
NotificationChannel channel = new NotificationChannel(
 "SMS_CHANNEL", "SMS Notifications",
NotificationManager.IMPORTANCE_HIGH);
 NotificationManager manager = context.getSystemService(NotificationManager.class);
manager.createNotificationChannel(channel);
 }

 // Build notification
 NotificationCompat.Builder builder = new NotificationCompat.Builder(context,
"SMS_CHANNEL")
 .setSmallIcon(R.drawable.ic_sms) // Replace with your app's icon
.setContentTitle(title)
 .setContentText(content)
 .setPriority(isUrgent ? NotificationCompat.PRIORITY_MAX :
NotificationCompat.PRIORITY_DEFAULT)
 .setAutoCancel(true);

 // Display the notification
 NotificationManagerCompat manager = NotificationManagerCompat.from(context);
manager.notify(isUrgent ? 100 : 200, builder.build());
            }

            private void logMessage(String sender, String content) {
 // Log message (could save to a file or database)
 System.out.println("SMS from " + sender + ": " + content);
```

```
        }
}
```

3. Register the Receiver

Add the `BroadcastReceiver` in `AndroidManifest.xml` to listen for SMS broadcasts:

```
<application>
        <receiver android:name=".SMSReceiver">
 <intent-filter>
 <action android:name="android.provider.Telephony.SMS_RECEIVED" />
</intent-filter>
        </receiver>
</application>
```

6. You are building an Android app for managing a to-do list. In the main activity, you need to create an Option Menu with the following options:

- **"Add Task" (Opens an activity to add a new task)**
- **"Settings" (Opens the settings page)**
- **"About" (Shows app information in a dialog)**

Write the code to implement the Option Menu and handle the menu item clicks.

1. Create the Menu XML File
Define the menu options in a resource file under `res/menu`.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
        <item
 android:id="@+id/menu_add_task"
 android:title="Add Task"
 android:icon="@drawable/ic_add_task"
 android:showAsAction="ifRoom" />
        <item
 android:id="@+id/menu_settings"
 android:title="Settings"
 android:icon="@drawable/ic_settings"
 android:showAsAction="never" />
        <item
 android:id="@+id/menu_about"
 android:title="About"
 android:icon="@drawable/ic_about"
 android:showAsAction="never" />
</menu>
```

2. Inflate the Menu in the Main Activity

Override the `onCreateOptionsMenu()` method to display the menu.

```
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.widget.Toast;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

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

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
 MenuInflater inflater = getMenuInflater();
 inflater.inflate(R.menu.menu_main, menu); // Inflate the menu  return
true;
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
 switch (item.getItemId()) {
 case R.id.menu_add_task:
```

```
 // Open the "Add Task" activity
 Intent intentAddTask = new Intent(this, AddTaskActivity.class);
startActivity(intentAddTask);
 return true;

 case R.id.menu_settings:
 // Open the "Settings" activity
 Intent intentSettings = new Intent(this, SettingsActivity.class);
startActivity(intentSettings);
 return true;

 case R.id.menu_about:
 // Show the "About" dialog
 showAboutDialog();
 return true;

 default:
 return super.onOptionsItemSelected(item);
 }
            }

            private void showAboutDialog() {
 new AlertDialog.Builder(this)
 .setTitle("About")
 .setMessage("To-Do List App\nVersion 1.0\nDeveloped by [Your Name]")
.setPositiveButton("OK", null)
 .show();
            }
 }
```
7 You are working on an emergency services application, where a user can initiate a phone call to emergency numbers by clicking a button labeled "Call Emergency." The app should detect whether the device has calling functionality and handle permission requests appropriately. How would you implement this phone call feature with permission handling and device compatibility checks? Explain the process and provide relevant code.

Implementation Steps

1. Check for Calling Capability

- Ensure the device supports calling by verifying the `TelephonyManager` features.
- This prevents errors on devices like tablets without calling functionality.

2. Request Runtime Permission

- From Android 6.0 (API level 23) onwards, you must request permissions at runtime. For phone calls, you need the `CALL_PHONE` permission.

3. Handle Button Click

- When the user clicks the "Call Emergency" button, check for calling capability and permissions, then initiate the phone call if both conditions are satisfied.

Code Implementation

```
import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.telephony.TelephonyManager;
import android.widget.Button;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

public class MainActivity extends AppCompatActivity {
        private static final int REQUEST_CALL_PERMISSION = 1;
        private static final String EMERGENCY_NUMBER = "112"; // Replace with emergency
number

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

 Button callEmergencyButton = findViewById(R.id.btn_call_emergency);
 // Set button click listener
 callEmergencyButton.setOnClickListener(v -> {
 if (isCallCapable()) {
 // Check and request permissions
 if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.CALL_PHONE) == PackageManager.PERMISSION_GRANTED) {
initiateCall();
 } else {
 ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.CALL_PHONE}, REQUEST_CALL_PERMISSION);
}
 } else {
 Toast.makeText(this, "This device does not support phone calls.",
Toast.LENGTH_SHORT).show();
 }
 });
            }

            private boolean isCallCapable() {
 // Check if the device supports telephony
 TelephonyManager telephonyManager = (TelephonyManager)
getSystemService(TELEPHONY_SERVICE);
 return telephonyManager != null && telephonyManager.getPhoneType() !=
TelephonyManager.PHONE_TYPE_NONE;
            }

            private void initiateCall() {
 // Start the phone call intent
 Intent callIntent = new Intent(Intent.ACTION_CALL);
 callIntent.setData(android.net.Uri.parse("tel:" + EMERGENCY_NUMBER));
startActivity(callIntent);
            }

            @Override
            public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);
 if (requestCode == REQUEST_CALL_PERMISSION) {  //
Check if permission is granted
 if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
 initiateCall();
 } else {
 Toast.makeText(this, "Permission denied. Cannot make the call.",
Toast.LENGTH_SHORT).show();
 }
 }
            }
}
```

Layout: activity_main.xml

Define a simple layout with the "Call Emergency" button.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical"
            android:gravity="center">

            <Button
 android:id="@+id/btn_call_emergency"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Call Emergency"
 android:padding="16dp"
 android:textSize="18sp" />
</LinearLayout>
```

8. In your Android app, you want to show a product catalog from an e-commerce site inside a WebView. The WebView should support JavaScript to allow interaction with the

site’s dynamic features. How would you enable JavaScript in the WebView and handle any potential issues, such as external links opening in the WebView itself? Explain the steps you would take.

1. Add the WebView to the Layout

Define a WebView component in your app's layout file.

activity_main.xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <WebView
        android:id="@+id/webView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

2. Configure WebView in the Activity

Enable JavaScript and handle potential issues, like external links opening within the WebView.

```
import android.os.Bundle;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

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

        // Initialize WebView
        WebView webView = findViewById(R.id.webView);

        // Enable JavaScript
        WebSettings webSettings = webView.getSettings();
        webSettings.setJavaScriptEnabled(true); // Enables JavaScript for dynamic features

        // Handle external links
        webView.setWebViewClient(new WebViewClient() {
        @Override
        public boolean shouldOverrideUrlLoading(WebView view, String url) {
                // Open all links within the WebView
                view.loadUrl(url);
                return true;
        }
        });

        // Load the product catalog URL
        webView.loadUrl("https://www.example-ecommerce-site.com/product-catalog");
        }
}
```

9. Explain how to insert and view multiple records into an SQLite database using ContentValues. Assume that you have a table called contacts with columns: id INTEGER PRIMARY KEY, contact_name TEXT, contact_number TEXT. Write the code to insert a new contact into the database.

Steps to Insert and View Data

1. Database Creation

You need to open or create the database using `SQLiteDatabase.openOrCreateDatabase()`.

2. Table Creation

Create the `contacts` table with `id`, `contact_name`, and `contact_number`.

3. Insert Multiple Records

Use the `ContentValues` object to insert records into the database.

4. View Records

Query the database using `rawQuery()`

```
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.content.ContentValues;
import android.os.Bundle;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

        private SQLiteDatabase db;

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

 // Create or open the database
 db = SQLiteDatabase.openOrCreateDatabase("/data/data/" + getPackageName() +
"/contactsDatabase.db", null);

 // Create the contacts table
 db.execSQL("CREATE TABLE IF NOT EXISTS contacts (id INTEGER PRIMARY KEY,
contact_name TEXT, contact_number TEXT);");

 // Insert multiple contacts
 insertContact("John Doe", "1234567890");
 insertContact("Jane Smith", "9876543210");
 insertContact("Alice Brown", "5551234567");

 // View all contacts
 viewContacts();
        }
        // Method to insert a new contact
        private void insertContact(String name, String number) {
 try {
 ContentValues values = new ContentValues();
 values.put("contact_name", name);
 values.put("contact_number", number);
 db.insert("contacts", null, values); // Insert the values into the database
Toast.makeText(this, "Contact added: " + name, Toast.LENGTH_SHORT).show();  }
catch (Exception e) {
 Toast.makeText(this, "Error inserting contact: " + e.getMessage(),
Toast.LENGTH_SHORT).show();
 }
            }

            // Method to view all contacts
            private void viewContacts() {
 try {
 Cursor cursor = db.rawQuery("SELECT * FROM contacts", null); // Retrieve all records  if
(cursor.getCount() > 0) {
 StringBuilder result = new StringBuilder();
 while (cursor.moveToNext()) {
 int id = cursor.getInt(cursor.getColumnIndex("id"));
 String name = cursor.getString(cursor.getColumnIndex("contact_name"));  String number =
cursor.getString(cursor.getColumnIndex("contact_number"));  result.append("ID:
").append(id).append(", Name: ").append(name)  .append(", Number:
").append(number).append("\n");
 }
 Toast.makeText(this, result.toString(), Toast.LENGTH_LONG).show();  } else {
 Toast.makeText(this, "No contacts found.", Toast.LENGTH_SHORT).show();  }
 cursor.close(); // Always close the cursor to free resources
 } catch (Exception e) {
 Toast.makeText(this, "Error retrieving contacts: " + e.getMessage(),
Toast.LENGTH_SHORT).show();
 }
            }

            @Override
            protected void onDestroy() {
 super.onDestroy();
 if (db != null && db.isOpen()) {
 db.close(); // Close the database connection
 }
            }
}
```

10. You are developing a contact management application where each contact is displayed in a ListView. Implement a Context Menu that provides the following options when a user long-presses a contact:

"Edit" (Opens a screen to edit the contact details)

"Delete" (Deletes the contact from the list and database)

Write the code to create the context menu and handle the actions for "Edit" and "Delete."

Steps to Implement Context Menu

1. **Create the Layout for the Contact List**: Use a `ListView` in your main activity's layout file.
2. **Register the Context Menu for the** `ListView`: Enable long-press functionality by registering the context menu for the `ListView`.
3. **Define Context Menu Options**: Use `onCreateContextMenu()` to add the menu items "Edit" and "Delete."
4. **Handle Context Menu Item Clicks**: Implement `onContextItemSelected()` to handle the "Edit" and "Delete" actions.
5. **Handle Edit and Delete Logic**:
   - **Edit**: Open a new activity to edit the contact details.
   - **Delete**: Remove the contact from the database and the list.

Code Implementation

MainActivity.java

This code demonstrates the context menu creation, options handling, and actions for "Edit" and "Delete."

```
import android.content.Intent;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

        private ListView contactListView;
        private ArrayAdapter<String> adapter;
        private ArrayList<String> contactList;
        private SQLiteDatabase db;

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

 // Initialize the database
 db = SQLiteDatabase.openOrCreateDatabase("/data/data/" + getPackageName() +
"/contactsDatabase.db", null);
 db.execSQL("CREATE TABLE IF NOT EXISTS contacts (id INTEGER PRIMARY KEY,
contact_name TEXT, contact_number TEXT);");

 // Sample data (This should be fetched from the database in real apps)
contactList = new ArrayList<>();
 contactList.add("John Doe - 1234567890");
 contactList.add("Jane Smith - 9876543210");

 // Set up ListView
 contactListView = findViewById(R.id.contactListView);
 adapter = new ArrayAdapter<>(this, android.R.layout.simple_list_item_1, contactList);
contactListView.setAdapter(adapter);
 // Register Context Menu
 registerForContextMenu(contactListView);

 // Handle item clicks (optional, for direct actions)
 contactListView.setOnItemClickListener((parent, view, position, id) -> {
```

```
Toast.makeText(this, "Clicked: " + contactList.get(position),
Toast.LENGTH_SHORT).show();
 });
            }

            @Override
            public void onCreateContextMenu(ContextMenu menu, View v,
ContextMenu.ContextMenuInfo menuInfo) {
 super.onCreateContextMenu(menu, v, menuInfo);

 if (v.getId() == R.id.contactListView) {
 menu.setHeaderTitle("Options");
 menu.add(0, 1, 0, "Edit"); // Option 1: Edit
 menu.add(0, 2, 0, "Delete"); // Option 2: Delete
 }
            }

            @Override
            public boolean onContextItemSelected(MenuItem item) {
 AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo)
item.getMenuInfo();
 int position = info.position; // Get the position of the clicked item  String
contact = contactList.get(position);

 switch (item.getItemId()) {
 case 1: // Edit
 editContact(position, contact);
 return true;

 case 2: // Delete
 deleteContact(position);
 return true;

 default:
 return super.onContextItemSelected(item);
 }
            }

            private void editContact(int position, String contact) {
```

```
 // Open EditContactActivity to edit contact details
 Intent intent = new Intent(this, EditContactActivity.class);
intent.putExtra("contact", contact);
 intent.putExtra("position", position);
 startActivity(intent);
            }

            private void deleteContact(int position) {
 // Remove contact from the database (optional: Add ID lookup logic)
contactList.remove(position); // Remove from the list
 adapter.notifyDataSetChanged(); // Update ListView
 Toast.makeText(this, "Contact deleted", Toast.LENGTH_SHORT).show(); }

            @Override
            protected void onDestroy() {
 super.onDestroy();
 if (db != null && db.isOpen()) {
 db.close(); // Close the database connection
 }
            }
 }
```