Second Part

▼ **TOC**

Q1) What is an Adapter?

When we want to connect data from a data source (such as ArrayList, HashMap, SQLite, etc.) to a UI component (such as ListView, GridView, etc.), we use an Adapter. The Adapter serves as a mediator between the UI component and data source. To create an Adapter, we can extend the BaseAdapter class, which is the parent class for all other adapter classes, such as ArrayAdapter, SimpleAdapter, and so on. With an Adapter, we can bind the data to the UI component, and we can customize the way the data is displayed on the UI. This is a common pattern in Android development, and it allows us to display dynamic data in various UI components.

Q2) What is Adapter View?

1. An Adapter View can be used to display large sets of data efficiently in form of List or Grid etc, provided to it by an Adapter.
2. An Adapter View is capable of displaying millions of items on the User Interface, while keeping the memory and CPU usage very low and without any noticeable lag.

3. It also reuses the already created layout to show data items as the user scrolls, hence saving the CPU usage
4. Types of views used for adapter:
    a. ListView:
        i. ListView is a view group when you have to show items in a vertically scrolling list
        ii. You can set divider between every item and set its height and color as per your UI design
        iii. Inside a ListView, we can show list of Text items by using TextView, or pictures using ImageView, or any other view or a combination of views.
        iv. Eg: Device's Contact List
        v. Eg:

```
<ListView
android:id="@+id/listView"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:divider="@android:color/black"
android:dividerHeight="1dp"/>
```

```
ListView listView;
String[] festivals = {"Diwali", "Holi", "Christmas"};

listView = (ListView) findViewById(R.id.listView);
final ArrayAdapter adapter = new ArrayAdapter(this, R.layout.list_item, f
estivals);
listView.setAdapter(adapter);

listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i,
long l) {
        Toast.makeText(MainActivity.this, "Item clicked " + i, Toast.LENG
TH_SHORT).show();
    }
});
```

    b. SearchView:
        i. Search view widget provides UI where one can enter and search query
        ii. Eg:

```
<SearchView
        android:id="@+id/searchView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:iconifiedByDefault="false"/>
```

iii. SearchView methods:

1. getQuery():
    a. This function is used to get the query string currently in the text field of a search view.
    b. This method returns CharSequence type value.
2. getQueryHint():
    a. This function is used for getting the hint text that will be displayed in the query text field.
    b. This method returns a CharSequence type value
3. public boolean onQueryTextSubmit(String query):
    a. It searches the query on the submission of content over SearchView editor.
    b. It is case dependent.
4. public boolean onQueryTextChange(String newText):
    a. It searches the query at the time of text change over SearchView editor.

c. Android AutoComplete TextView:

i. AutoCompleteTextView is a view that is similar to EditText, except that it shows a list of completion suggestions automatically while the user is typing.

ii. The list of suggestions is displayed in drop down menu.

iii. Eg:

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <AutoCompleteTextView
        android:id="@+id/autoCompleteTextView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:hint="@string/hint"/>

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/submit"/>

</LinearLayout>
```

```
AutoCompleteTextView autocomplete;
String[] arr = { "Paries,France", "PA,United States","Parana,Brazil", "Pa
dua,Italy", "Pasadena,CA,United States"};

autocomplete = (AutoCompleteTextView) findViewById(R.id.autoCompleteTextV
iew);
ArrayAdapter<String> adapter = new ArrayAdapter<String> (this,android.R.l
ayout.select_dialog_item, arr);
autocomplete.setThreshold(2); // suggest after 2 characters are typed
autocomplete.setAdapter(adapter);
```

Q3) Array Adapter and Custom Array Adapter?

1. ArrayAdapter – It is used whenever we have a list of single items which is backed by an array
2. Custom ArrayAdapter – It is used whenever we need to display a custom list
3. Array Adapter is an implementation of Base Adapter so if we want our custom array adapter then we can extend array adapter and override all the mehods of base adapter
4. Eg:

```
public class MyAdapter extends ArrayAdapter<String> {

    public MyAdapter(Context context, int resource, int textViewResourceId, List<S
tring> objects) {
        super(context, resource, textViewResourceId, objects);
    }

    @Override
    public int getCount() {
        return super.getCount();
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        return super.getView(position, convertView, parent);
    }
}
```

5. Methods:
   a. getCount():
      i. The getCount() function returns the total number of items to be displayed in a list.
      ii. It counts the value from arraylist size or an array’s length.
   b. getView(int i, View view, ViewGroup viewGroup):
      i. This function is automatically called when the list item view is ready to be displayed
      ii. In this function we set the layout for list items using LayoutInflater class and then add the data to the views like ImageView, TextView etc.
   c. public MyAdapter(Context context, int resource, int textViewResourceId, List objects):
      i. Context: returns the current context/reference of the current class
      ii. Resource: returns the resource id of the layout file
      iii. Objects: returns the objects in the list

Q4) Simple Adapter and Custom Simple Adapter?

1. SimpleAdapter is an easy Adapter to map static data to views defined in an XML file(layout)
2. It stores/add data in the form of HashMap
3. It implements the Map interface which allows us to store key and value pair, where keys should be unique
4. We cant perform onclick event on simple adapter and if we want customized simple adapter then we use custom simple adapter which extends simple adapter
5. Eg:

```
public class CustomAdapter extends SimpleAdapter {

    public CustomAdapter(Context context, List<? extends Map<String, ?>> data, int
resource, String[] from, int[] to) {
        super(context, data, resource, from, to);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        return super.getView(position, convertView, parent);
    }

    @Override
    public int getCount() {
        return super.getCount();
    }
}
```

6. Methods:
    a. getCount():
        i. The getCount() function returns the total number of items to be displayed in a list.
        ii. It counts the value from arraylist size or an array’s length.
    b. getView(int i, View view, ViewGroup viewGroup):
        i. This function is automatically called when the list item view is ready to be displayed
        ii. In this function we set the layout for list items using LayoutInflater class and then add the data to the views like ImageView, TextView etc.
    c. public MyAdapter(Context context, int resource, int textViewResourceId, List objects):
        i. Context: returns the current context/reference of the current class
        ii. Resource: returns the resource id of the layout file
        iii. Objects: returns the objects in the list
    d. from: A list of column names that will be added to the Map associated with each item.
    e. to:
        i. The views that should display column in the “from” parameter.

ii. These should all be TextViews.

iii. The first N views in this list are given the values of the first N columns in the from parameter.

Q5) Adapter View Flipper?

1. It is a subclass of ViewAnimator class and is used to flip between two or more views such that only one view is displayed at a time
2. It consists of transition element which helps to add transitions to the views
3. It can be used with Text View, Image View, etc.
4. AdapterViewFlipper is used to display all child views.
5. So there is room for recycling views and loading views dynamically.

Q6) Expandable ListView Adapter?

1. ExpandableListAdapter is an Adapter that links the ExpandableListView with the underlying data
2. The implementation of this interface will provide the data for the children and also initiate the views for the children and groups.

Methods	Use
setChildIndicator(Drawable)	This is used to show an indicator besides each item representing the current state
setGroupIndicator(Drawable)	An indicator is drawn besides the group representing its state i.e. expanded or collapsed.
getGroupView()	It returns view for the list group header
getChildView()	It returns view for list child item
ExpandableListView.OnChildClickListener	It is invoked when a child in the expanded list is clicked
ExpandableListView.OnGroupClickListener	It is invoked when a group header in the expanded list is clicked
ExpandableListView.OnGroupCollapseListener	It is used for notifying when a group is collapsed
ExpandableListView.OnGroupExpandListener	It is used to notify when a group is expanded

Q7) Recycler View?

1. The RecyclerView class extends the ViewGroup class and implements ScrollingView interface
2. It is an advanced version of ListView and GridView, it is a container for displaying large amount of data that can be scrolled efficiently by maintaining a limited number of views.
3. Recycler View should be used when we have to display large amount of data and the size is dynamic
4. ListView and GridView just recycle the item layout , but no references are kept with blayout children
5. Everytime we call a child we need to call findViewById for every child which is not efficient
6. This problem is solved by using recycler view as it uses View Holder to store the reference of the View's for one entry in the RecyclerView.
7. Components of RecylerView :
    a. Layout managers:
        i. Adapters are only responsible for creating and managing views for items (called ViewHolder), these classes do not decide how these views are arranged when displaying them. Instead, they rely on a separate class called LayoutManager.
        ii. LayoutManager is a class that tells Adapters how to arrange those items.
        iii. 3 types of Layout Manager :
            1. Linear Layout Manager – It is used for displaying the data items in a horizontal or vertical scrolling List
            2. GridLayoutManager – It is used to show the items in grid format
            3. StaggeredGridLayoutManager – It is used to show the items in staggered Grid.
    b. ViewHolder:
        i. ViewHolder is used to store the reference of the View's for one entry in the RecyclerView.
        ii. A ViewHolder is a static inner class in our Adapter which holds references to the relevant view's.

    c. Recycler.ViewAdapter:
        i. It is similar to other adpaters but we override two main methods for handling views.
        ii. First one to inflate the view and its viewholder and another one to bind the data to the view
        iii. public void onBindViewHolder and public class MyViewHolder extends RecyclerView.ViewHolder
    d. ItemAnimator:
        i. RecyclerView.ItemAnimator will animate ViewGroup modification such as delete, select, add that notify the Adapter

Q8) Grid View and GridLayout Manager?

1. A GridView is a type of AdapterView that displays items in a two-dimensional scrolling grid.
2. Items are inserted into this grid layout from a database or from an array.
3. Two types of grid managers are :
    a. Grid Layout manager:
        i. It is used for displaying the data items in grid format
        ii. We use the GridLayoutManager for displaying RecyclerView as a GridView.
        iii. GridLayoutManager (Context context, int spanCount, int orientation, boolean reverseLayout): In this constructor first parameter is used to set the current context and second parameter is used to set the span Count means the number of columns in the grid and the third parameter is boolean value, when set to false value for reverseLayout to show the items from start to end
    b. StaggeredGridLayoutManager:
        i. It is used to create a StaggeredGridLayoutManager with given parameters.
        ii. StaggeredGridLayoutManager(int spanCount, int orientation): First parameter is used to set spanCount means number of columns if orientation is vertical or number of rows if orientation is horizontal,

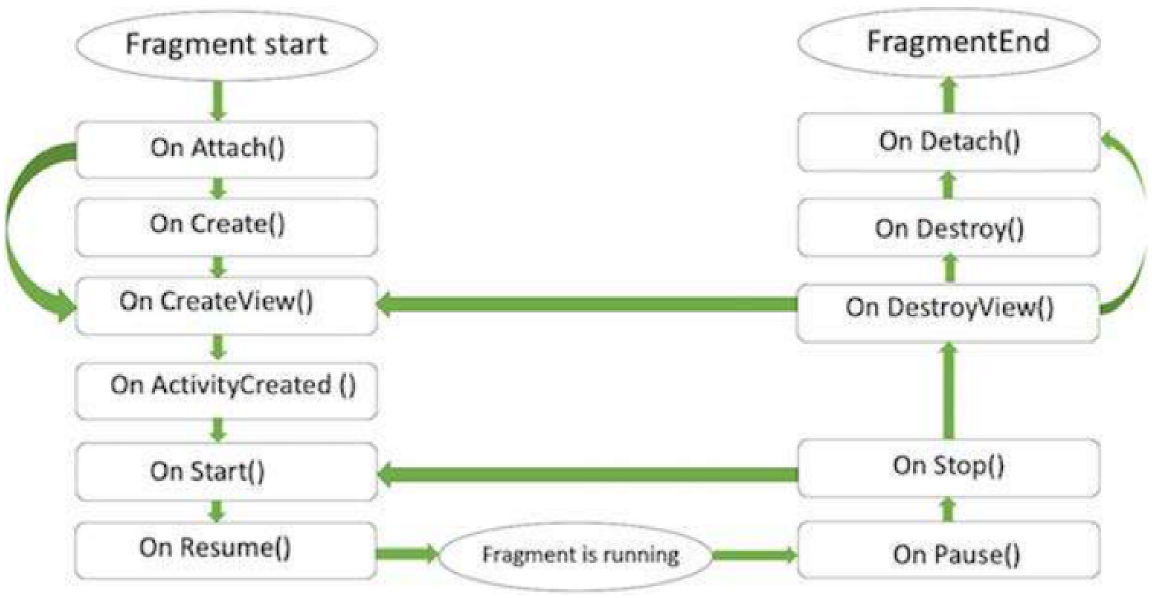
Second Parameter is used to set the Orientation, it should be vertical or horizontal.

Q9) Fragments?

1. Android Fragment is the part of activity, it is also known as sub-activity.
2. A fragment has its own layout and its own behavior
3. There can be more than one fragment in an activity.
4. Fragments represent multiple screen inside one activity.
5. Fragments can be dynamically added and removed as per the requirements.
6. Fragments improve the adaptability & user experience
7. There are 2 types of Fragments :
    a. Single Fragments:
        i. Display only one single view on the device screen. This type of fragment is mostly used for mobile phones.
    b. List Fragment:
        i. This Fragment is used to display a list-view from which the user can select the desired sub-activity.
        ii. Eg: Gmail
    c. Fragment Transaction:
        i. This kind of fragments supports the transition from one fragment to another at run time. Users can switch between multiple fragments like switching tabs.
8. FragmentManager is the class responsible for performing actions on your app's fragments, such as adding, removing, or replacing them, and adding them to the back stack.
9. Adding Fragments:
    a. Statically:
        i. To add fragment statically we must mention it in our activirty_main.xml
    b. Dynamically:

i. To add fragment dynamically we embed our fragment in activity using Fragment Manager

10. Fragment Life Cycle:



Methods	Description
onAttach()	Executes when fragment attaches to an activity
onCreate()	Used to create a fragment
On CreateView()	Executes when we need to create UI for fragment
On ActivityCreated()	Invoked when an activity is created
On Start()	invoked once the fragment gets visible.
On Resume()	Fragment becomes active
OnPause()	Invoked when fragment is no longer active and user is about t leave it
OnStop()	Invoked when fragment is no longer visible
OnDestroyView()	Invoked to destroy the view
OnDestroy()	Allows fragment to clean up fragment state
OnDetach()	Used to detach fragment from activity

Q10) What is intent? Types of intent?

1. An intent is a messaging object used to request any action from another app component.
2. Intent is to perform an action.

3. An intent is a system message
4. Android intents are mainly used to :
    a. Start activity
    b. Start services
    c. Send broadcast receiver
    d. Send message between two activities
    e. Dial a phone call
5. There are two types of intent:
    a. Implicit Intent:
        i. In android, Implicit Intents won’t specify any name of the component to start, instead it declare an action to perform
        ii. In implicit intent we have to pass an action using setAction()
        iii. Eg:
            1. Intent intent=new Intent(Intent.ACTION_VIEW);

               intent.setData(Uri.parse("https://www.geeksforgeeks.org/"));

               startActivity(intent);
    b. Explicit Intents:
        i. By using explicit intents we can send data content from one activity to another activity based on our requirements.
        ii. To create an Explicit Intent, we need to define the component name for Intent object.
6. Transfering data using Bundle :
    a. Bundle is a class which is used to pass data from one activity to another activity within an android application.
    b. We can pass data using key and value pairs using bundles.
    c. Eg:

       a.java:

```
Bundle b = new Bundle();
b.putString("Key", value);
```

```
Intent i = new Intent(a.this, b.class);
i.putExtras(b);
startActivity(i);
```

b.java:

```
Bundle b = getIntent().getExtras();

if (b != null) {
String value = b.getString("Key");
TextView t1 = (TextView) findViewById(R.id.t1);
t1.setText(value);
}
```

7. Methods in Intent:

Method	Description
Context.startActivity()	This is to launch a new activity or get an existing activity to be action.
Context.startService()	This is to start a new service or deliver instructions for an existing service.
Context.sendBroadcast()	This is to deliver the message to broadcast receivers.
getIntent().getExtras()	It is used to get values from intent that are stored in bundle

Q11) Explain DatePicker and TimePicker with example?

1. Date Picker:
   a. Android Date Picker allows you to select the date consisting of day, month and year in your custom user interface.
   b. The android.widget.DatePicker is the subclass of FrameLayout class.
   c. Methods:

Methods	Description
onDateSetListener()	These callback method is invoked when the user is done with filling the date
getDayOfMonth()	This method gets the selected day of month
getMonth()	This method gets the selected month
getYear()	This method gets the selected year
getCalendarView()	This method returns calendar view

2. Time Picker:
    a. Android Time Picker allows you to select the time of day in either 24 hour or AM/PM mode. The time consists of hours, minutes and clock format.
    b. The android.widget.TimePicker is the subclass of FrameLayout class.
    c. Methods:

Methods	Description
onTimeSetListener()	These callback method is invoked when the user is done with filling the time
getCurrentHour()	This method gets the current hour
getCurrentMinute()	This method gets the current minute
is24HourView()	This method returns true if this is in 24 hour view else false
setCurrentHour(Integer currentHour)	This method sets the current hour
setCurrentMinute(Integer currentMinute)	This method sets the current minute

Eg:

Activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
 android:id="@+id/bn"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical">

<DatePicker
    android:id="@+id/date"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

<TimePicker
    android:id="@+id/time"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

<Button
    android:id="@+id/b1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Date" />
```

```
<Button
    android:id="@+id/b2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Time" />

<TextView
    android:id="@+id/dates"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Date" />

<TextView
    android:id="@+id/times"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Time"/>
</LinearLayout>
```

Activity_main.java:

```
DatePicker picker;
TimePicker picker2;
Button displayDate;
Button displayTime;
TextView t1;
TextView t2;

t1 = (TextView)findViewById(R.id.times);
picker = (DatePicker)findViewById(R.id.date);
displayDate = (Button)findViewById(R.id.b1);
t2 = (TextView)findViewById(R.id.times);
picker2 = (TimePicker)findViewById(R.id.time);
displayTime = (Button)findViewById(R.id.b2);

t1.setText("Current Date: " + getCurrentDate());
displayDate.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        t1.setText("Change Date: " + getCurrentDate());
    }
});

public String getCurrentDate() {
    StringBuilder builder = new StringBuilder();
    builder.append((picker.getMonth() + 1) + "/");
    builder.append(picker.getDayOfMonth() + "/");
    builder.append(picker.getYear());
    return builder.toString();
}

timepicker.setIs24HourView(true);

displayTime.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        t2.setText(getCurrentTime());
    }
});

public String getCurrentTime() {
    String currentTime = "Current Time: " + timepicker.getCurrentHour() + ":" + timepi
cker.getCurrentMinute();
    return currentTime;
}
```

Q12) Explain Alert Dialogue in Android?

1. It is used to prompt a dialogue to the user with message and buttons to perform an action to proceed further
2. It consists of 3 regions: Title, Content area, Action buttons

Methods	Description
setTitle()	Used to set title
setIcon()	Used to set Icon
setMessage()	Used to set Message
setpositiveButton()	Set positive button
setNegativeButton()	Set negative button
setNeutralButton()	Set the neutral button

3. Types of Dilogue are:
    a. Alert
    b. DatePicker
    c. TimePicker

Q13) What are shared preferences in android?

1. Android provides many ways of storing data of an application. One of this way is called Shared Preferences.
2. Shared Preferences allow you to save and retrieve data in the form of key,value pair.
3. 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.

4. Shared Preferences are used to store the users data and return the data to the user when the app is opened again
5. getSharedpreference(String name, int mode):
    a. This method takes two arguments, the first being the name of the SharedPreference(SP) file and the other is the context mode that we want to store our file in.
    b. MODE_PUBLIC will make the file public which could be accessible by other applications on the device
    c. MODE_PRIVATE keeps the files private and secures the user's data.
    d. MODE_APPEND is used while reading the data from the SP fil

Methods	Description
Edit()	It is used to create a new editor to edit the preferences
getAll()	It retrieves all the values from the preferences
registerOnSharedPreferencechangeListener	This method is used to register a callback to be invoked when a change happens to a preference.
unregisterOnSharedPreferencechangeListener	This method is used to unregister a previous callback.
getString()	It retrieves the String value from the preference

Q14) Explain SQLite Database?

1. SQLite is an open-source relational database i.e. used to perform database operations on android devices such as storing, manipulating or retrieving persistent data from the database.
2. It is embedded in android by default , theres no need to perform any database setup
3. The Lite in SQLite refers that SQL is a lightweight open source DB as compared to others
4. Features:
    a. Server less

b. Zero Configuration

c. Cross platform

d. Reliable

e. Scalable

f. Light weight

g. Transactional

h. Self contained

5. SQlite OpenHelper Class:

    a. The android.database.sqlite.SQLiteOpenHelper class is used for database creation and version management.

    b. For performing any database operation, you have to provide the implementation of onCreate() and onUpgrade() methods of SQLiteOpenHelper class.

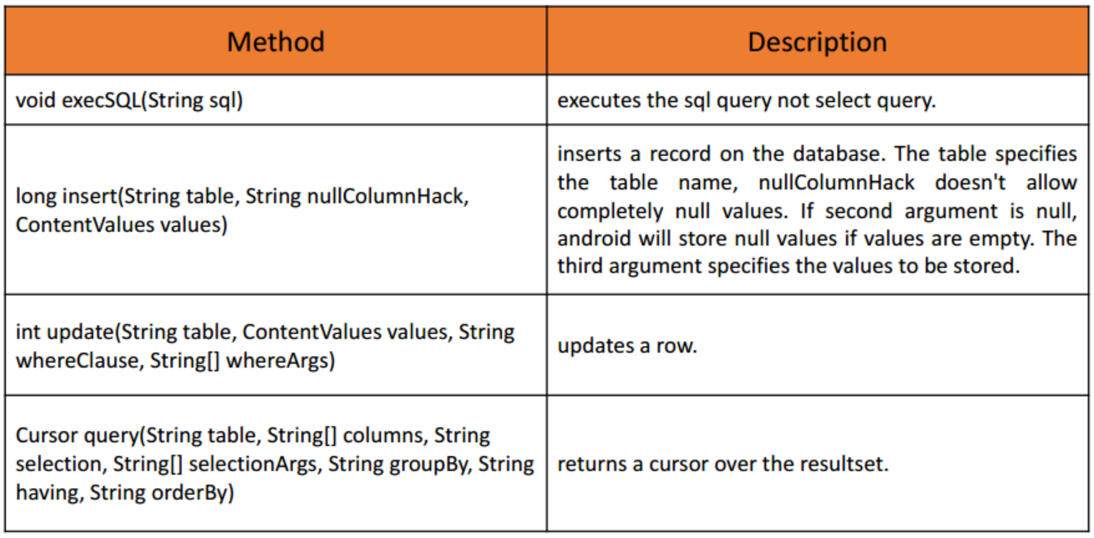
    c. Methods:

        i. onCreate(SQLiteDatabase db): called only once when database is created for the first time.

        ii. onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion): called when database needs to be upgraded.

6. SQLiteDatabase class:

    a. It contains methods to be performed on sqlite database such as create, update, delete, select etc.



Method	Description
void execSQL(String sql)	executes the sql query not select query.
long insert(String table, String nullColumnHack, ContentValues values)	inserts a record on the database. The table specifies the table name, nullColumnHack doesn't allow completely null values. If second argument is null, android will store null values if values are empty. The third argument specifies the values to be stored.
int update(String table, ContentValues values, String whereClause, String[] whereArgs)	updates a row.
Cursor query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy)	returns a cursor over the resultset.

Q15) Explain Menu in Android?

1. In android, Menu is an important part of the UI component which is used to provide some common functionality around the application.
2. In order to use menu, we should define it in a separate XML file and use that file in our application based on our requirements.
3. Also, we can use menu APIs to represent user actions
4. Following is the method used:
    a. <menu>: A <menu> element defines a menu, which is a container for menu items that holds one or more elements.
    b. <item> It is used to create a single item in the menu. It also contains nested <menu> element in order to create a submenu.
    c. <group> It is optional and invisible for <item> elements to categorize the menu items
5. There are 3 types of menu:
    a. Option Menu:
        i. This type of menu is a primary collection of menu items in an app and is useful for actions that have a global impact on the searching app.
        ii. The Option Menu can be used for settings, searching, deleting items, sharing, etc.
    b. Context Menu:

   i. This type of menu is a floating menu that only appears when a user presses for a long time on an element and is useful for elements that affect the selected content or context frame.

c. Popup Menu:

   i. Using Popup Menu we can display a list of items in a vertical list which presents the view that invokes the menu.

   ii. Popup Menu is useful since it can provide an overflow of actions which are related to any specific content.