How to Read Data from SQLite Database in Android?

In the 1st part of our SQLite database, we have seen How to Create and Add Data to SQLite Database in Android. In that article, we have added data to our **SQLite Database**. In this article, we will **read all this data from the SQLite database and display this data in RecyclerView.**

What we are going to build in this article?

We will be working on the existing application which we have built in our previous article. In that application, we are simply adding the course list part where we can get to see our list of courses Note that we are going to implement this project using the **Java** language.

Step by Step Implementation

Step 1: Working with the activity_main.xml file

Go to the **activity_main.xml** file and add a new Button to open a new activity for displaying our list of courses.

```
<!--new button for opening our course list activity-->
<Button
  android:id="@+id/idBtnReadCourse"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_margin="10dp"
  android:text="Read All Courses"
  android:textAllCaps="false" />
```

Now below is the updated code for the **activity_main.xml** file after adding the above code snippet.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <!--Edit text to enter course name-->
    <EditText
        android:id="@+id/idEdtCourseName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:hint="Enter course Name" />

    <!--edit text to enter course duration-->
    <EditText
        android:id="@+id/idEdtCourseDuration"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:hint="Enter Course Duration" />
```

```
    <!--edit text to display course tracks-->
    <EditText
        android:id="@+id/idEdtCourseTracks"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:hint="Enter Course Tracks" />

    <!--edit text for course description-->
    <EditText
        android:id="@+id/idEdtCourseDescription"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:hint="Enter Course Description" />

    <!--button for adding new course-->
    <Button
        android:id="@+id/idBtnAddCourse"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:text="Add Course"
        android:textAllCaps="false" />

    <!--new button for opening our course list activity-->
    <Button
        android:id="@+id/idBtnReadCourse"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:text="Read All Courses"
        android:textAllCaps="false" />

</LinearLayout>
```

Step 2: Creating a modal class for storing our data

Navigate to the **app > java > your app's package name > Right-click on it > New > Java class** and name it as **CourseModal** and add the below code to it. Comments are added inside the code to understand the code in more detail.

```
public class CourseModal {

    // variables for our coursename,
    // description, tracks and duration, id.
    private String courseName;
    private String courseDuration;
    private String courseTracks;
    private String courseDescription;
    private int id;

    // creating getter and setter methods
    public String getCourseName() {
        return courseName;
    }

    public void setCourseName(String courseName) {
        this.courseName = courseName;
    }
```

```
    public String getCourseDuration() {
        return courseDuration;
    }

    public void setCourseDuration(String courseDuration) {
        this.courseDuration = courseDuration;
    }

    public String getCourseTracks() {
        return courseTracks;
    }

    public void setCourseTracks(String courseTracks) {
        this.courseTracks = courseTracks;
    }

    public String getCourseDescription() {
        return courseDescription;
    }

    public void setCourseDescription(String courseDescription) {
        this.courseDescription = courseDescription;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    // constructor
    public CourseModal(String courseName, String courseDuration, String courseTracks,
String courseDescription) {
        this.courseName = courseName;
        this.courseDuration = courseDuration;
        this.courseTracks = courseTracks;
        this.courseDescription = courseDescription;
    }
}
```

Step 5: Updating our Java class for performing SQLite operations.

Navigate to the **app > java > your app's package name > DBHandler** and add the below code to it. In this, we are simply adding a new method for reading all the courses from the SQLite database.

```
// we have created a new method for reading all the courses.
public ArrayList<CourseModal> readCourses() {
        // on below line we are creating a
        // database for reading our database.
        SQLiteDatabase db = this.getReadableDatabase();

        // on below line we are creating a cursor with query to read data from database.
        Cursor cursorCourses = db.rawQuery("SELECT * FROM " + TABLE_NAME, null);

        // on below line we are creating a new array list.
        ArrayList<CourseModal> courseModalArrayList = new ArrayList<>();
```

```
        // moving our cursor to first position.
        if (cursorCourses.moveToFirst()) {
            do {
                // on below line we are adding the data from cursor to our array list.
                courseModalArrayList.add(new CourseModal(cursorCourses.getString(1),
                                                 cursorCourses.getString(4),
                                                 cursorCourses.getString(2),
                                                 cursorCourses.getString(3)));
            } while (cursorCourses.moveToNext());
            // moving our cursor to next.
        }
        // at last closing our cursor
        // and returning our array list.
        cursorCourses.close();
        return courseModalArrayList;
}
```

Below is the updated code for the **DBHandler.java** file after adding the above code snippet.

```
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import java.util.ArrayList;

public class DBHandler extends SQLiteOpenHelper {

    // creating a constant variables for our database.
    // below variable is for our database name.
    private static final String DB_NAME = "coursedb";

    // below int is our database version
    private static final int DB_VERSION = 1;

    // below variable is for our table name.
    private static final String TABLE_NAME = "mycourses";

    // below variable is for our id column.
    private static final String ID_COL = "id";

    // below variable is for our course name column
    private static final String NAME_COL = "name";

    // below variable id for our course duration column.
    private static final String DURATION_COL = "duration";

    // below variable for our course description column.
    private static final String DESCRIPTION_COL = "description";

    // below variable is for our course tracks column.
    private static final String TRACKS_COL = "tracks";

    // creating a constructor for our database handler.
    public DBHandler(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
    }

    // below method is for creating a database by running a sqlite query
```

```
    @Override
    public void onCreate(SQLiteDatabase db) {
        // on below line we are creating
        // an sqlite query and we are
        // setting our column names
        // along with their data types.
        String query = "CREATE TABLE " + TABLE_NAME + " ("
                + ID_COL + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + NAME_COL + " TEXT,"
                + DURATION_COL + " TEXT,"
                + DESCRIPTION_COL + " TEXT,"
                + TRACKS_COL + " TEXT)";

        // at last we are calling a exec sql
        // method to execute above sql query
        db.execSQL(query);
    }

    // this method is use to add new course to our sqlite database.
    public void addNewCourse(String courseName, String courseDuration, String
courseDescription, String courseTracks) {

        // on below line we are creating a variable for
        // our sqlite database and calling writable method
        // as we are writing data in our database.
        SQLiteDatabase db = this.getWritableDatabase();

        // on below line we are creating a
        // variable for content values.
        ContentValues values = new ContentValues();

        // on below line we are passing all values
        // along with its key and value pair.
        values.put(NAME_COL, courseName);
        values.put(DURATION_COL, courseDuration);
        values.put(DESCRIPTION_COL, courseDescription);
        values.put(TRACKS_COL, courseTracks);

        // after adding all values we are passing
        // content values to our table.
        db.insert(TABLE_NAME, null, values);

        // at last we are closing our
        // database after adding database.
        db.close();
    }

    // we have created a new method for reading all the courses.
    public ArrayList<CourseModal> readCourses() {
        // on below line we are creating a
        // database for reading our database.
        SQLiteDatabase db = this.getReadableDatabase();

        // on below line we are creating a cursor with query to read data from database.
        Cursor cursorCourses = db.rawQuery("SELECT * FROM " + TABLE_NAME, null);

        // on below line we are creating a new array list.
        ArrayList<CourseModal> courseModalArrayList = new ArrayList<>();

        // moving our cursor to first position.
        if (cursorCourses.moveToFirst()) {
```

```
            do {
                // on below line we are adding the data from cursor to our array list.
                courseModalArrayList.add(new CourseModal(cursorCourses.getString(1),
                                                cursorCourses.getString(4),
                                                cursorCourses.getString(2),
                                                cursorCourses.getString(3)));
            } while (cursorCourses.moveToNext());
            // moving our cursor to next.
        }
        // at last closing our cursor
        // and returning our array list.
        cursorCourses.close();
        return courseModalArrayList;
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // this method is called to check if the table exists already.
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
    }
}
```

Step 6: Creating a new Activity for displaying our list of courses

To create a new Activity we have to navigate to the **app > java > your app's package name > Right click on package name > New > Empty Activity** and name your activity as **ViewCourses** and create new Activity. Make sure to select the **empty activity**.

Step 7: Working with the MainActivity.java file

As we have added a new button to our **activity_main.xml** file so we have to add **setOnClickListener()** to that button in our **MainActivity.java** file.

```
readCourseBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // opening a new activity via a intent.
                Intent i = new Intent(MainActivity.this, ViewCourses.class);
                startActivity(i);
        }
});
```

Below is the updated code for the **MainActivity.java** file after adding the above code snippet.

```
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    // creating variables for our edittext, button and dbhandler
    private EditText courseNameEdt, courseTracksEdt, courseDurationEdt,
courseDescriptionEdt;
    private Button addCourseBtn, readCourseBtn;
```

```
    private DBHandler dbHandler;

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

        // initializing all our variables.
        courseNameEdt = findViewById(R.id.idEdtCourseName);
        courseTracksEdt = findViewById(R.id.idEdtCourseTracks);
        courseDurationEdt = findViewById(R.id.idEdtCourseDuration);
        courseDescriptionEdt = findViewById(R.id.idEdtCourseDescription);
        addCourseBtn = findViewById(R.id.idBtnAddCourse);
        readCourseBtn = findViewById(R.id.idBtnReadCourse);

        // creating a new dbhandler class
        // and passing our context to it.
        dbHandler = new DBHandler(MainActivity.this);

        // below line is to add on click listener for our add course button.
        addCourseBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                // below line is to get data from all edit text fields.
                String courseName = courseNameEdt.getText().toString();
                String courseTracks = courseTracksEdt.getText().toString();
                String courseDuration = courseDurationEdt.getText().toString();
                String courseDescription = courseDescriptionEdt.getText().toString();

                // validating if the text fields are empty or not.
                if (courseName.isEmpty() && courseTracks.isEmpty() &&
courseDuration.isEmpty() && courseDescription.isEmpty()) {
                    Toast.makeText(MainActivity.this, "Please enter all the data..",
Toast.LENGTH_SHORT).show();
                    return;
                }

                // on below line we are calling a method to add new
                // course to sqlite data and pass all our values to it.
                dbHandler.addNewCourse(courseName, courseDuration, courseDescription,
courseTracks);

                // after adding the data we are displaying a toast message.
                Toast.makeText(MainActivity.this, "Course has been added.",
Toast.LENGTH_SHORT).show();
                courseNameEdt.setText("");
                courseDurationEdt.setText("");
                courseTracksEdt.setText("");
                courseDescriptionEdt.setText("");
            }
        });

        readCourseBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // opening a new activity via a intent.
                Intent i = new Intent(MainActivity.this, ViewCourses.class);
                startActivity(i);
            }
        });
    }
```

```
}
```

Step 8: Creating a new layout file for our item of RecyclerView

Navigate to the **app > res > layout > Right-click on it > New > Layout resource file** and name it as **course_rv_item** and add the below code to it.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="5dp"
    android:elevation="8dp"
    app:cardCornerRadius="4dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="3dp"
        android:orientation="vertical">

        <!--text view for our course name-->
        <TextView
            android:id="@+id/idTVCourseName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="3dp"
            android:text="Course Name"
            android:textColor="@color/black" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:weightSum="2">

            <!--text view for our course tracks-->
            <TextView
                android:id="@+id/idTVCourseTracks"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:padding="3dp"
                android:text="Course Tracks"
                android:textColor="@color/black" />


            <!--text view for our course duration-->
            <TextView
                android:id="@+id/idTVCourseDuration"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:padding="3dp"
                android:text="Duration"
                android:textColor="@color/black" />

        </LinearLayout>
```

```
        <!--text view for our course description-->
        <TextView
            android:id="@+id/idTVCourseDescription"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:padding="3dp"
            android:text="Description"
            android:textColor="@color/black" />

    </LinearLayout>

</androidx.cardview.widget.CardView>
```

Step 9: Creating an Adapter class for setting data to our items of RecyclerView

Navigate to the **app > java > your app's package name > Right-click on it > New > Java class** and name it as **CourseRVAdapter** and add the below code to it. Comments are added inside the code to understand the code in more detail.

```
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;

public class CourseRVAdapter extends RecyclerView.Adapter<CourseRVAdapter.ViewHolder> {

    // variable for our array list and context
    private ArrayList<CourseModal> courseModalArrayList;
    private Context context;

    // constructor
    public CourseRVAdapter(ArrayList<CourseModal> courseModalArrayList, Context context) {
        this.courseModalArrayList = courseModalArrayList;
        this.context = context;
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        // on below line we are inflating our layout
        // file for our recycler view items.
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.course_rv_item, parent, false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        // on below line we are setting data
        // to our views of recycler view item.
        CourseModal modal = courseModalArrayList.get(position);
        holder.courseNameTV.setText(modal.getCourseName());
        holder.courseDescTV.setText(modal.getCourseDescription());
```

```
        holder.courseDurationTV.setText(modal.getCourseDuration());
        holder.courseTracksTV.setText(modal.getCourseTracks());
    }

    @Override
    public int getItemCount() {
        // returning the size of our array list
        return courseModalArrayList.size();
    }

    public class ViewHolder extends RecyclerView.ViewHolder {

        // creating variables for our text views.
        private TextView courseNameTV, courseDescTV, courseDurationTV, courseTracksTV;

        public ViewHolder(@NonNull View itemView) {
            super(itemView);
            // initializing our text views
            courseNameTV = itemView.findViewById(R.id.idTVCourseName);
            courseDescTV = itemView.findViewById(R.id.idTVCourseDescription);
            courseDurationTV = itemView.findViewById(R.id.idTVCourseDuration);
            courseTracksTV = itemView.findViewById(R.id.idTVCourseTracks);
        }
    }
}
```

Step 10: Working with the activity_view_course.xml file

Navigate to the **app > res > layout > activity_view_course.xml** and add the below code to that file. Below is the code for the **activity_view_course.xml** file.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ViewCourses">

    <!--recycler view for displaying our courses-->
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/idRVCourses"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</RelativeLayout>
```

Step 11: Working with the ViewCourses.java file

Navigate to the **app > java > your app's package name > ViewCourses.java** file and add the below code to it. Comments are added inside the code to understand the code in more detail.

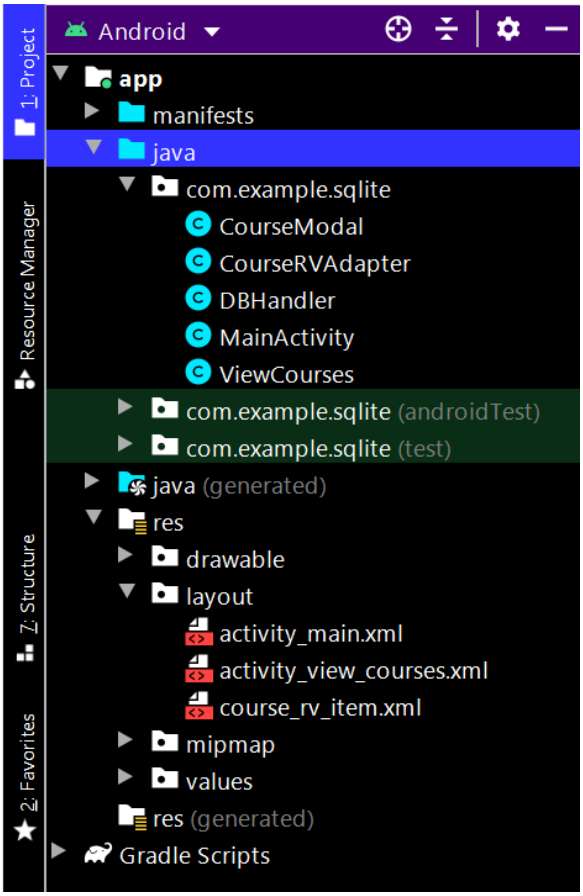
```
import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;
```

```
public class ViewCourses extends AppCompatActivity {

    // creating variables for our array list,
    // dbhandler, adapter and recycler view.
    private ArrayList<CourseModal> courseModalArrayList;
    private DBHandler dbHandler;
    private CourseRVAdapter courseRVAdapter;
    private RecyclerView coursesRV;

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

        // initializing our all variables.
        courseModalArrayList = new ArrayList<>();
        dbHandler = new DBHandler(ViewCourses.this);

        // getting our course array
        // list from db handler class.
        courseModalArrayList = dbHandler.readCourses();

        // on below line passing our array lost to our adapter class.
        courseRVAdapter = new CourseRVAdapter(courseModalArrayList, ViewCourses.this);
        coursesRV = findViewById(R.id.idRVCourses);

        // setting layout manager for our recycler view.
        LinearLayoutManager linearLayoutManager = new
LinearLayoutManager(ViewCourses.this, RecyclerView.VERTICAL, false);
        coursesRV.setLayoutManager(linearLayoutManager);

        // setting our adapter to recycler view.
        coursesRV.setAdapter(courseRVAdapter);
    }
}
```

Now run your app and see the output of the app. Make sure to add the data before reading the data.

Output:

Below is the complete project file structure after performing the read operation:


Android
app
manifests
java
com.example.sqlite
CourseModal
CourseRVAdapter
DBHandler
MainActivity
ViewCourses
com.example.sqlite (androidTest)
com.example.sqlite (test)
java (generated)
res
drawable
layout
activity_main.xml
activity_view_courses.xml
course_rv_item.xml
mipmap
values
res (generated)
Gradle Scripts
1: Project
Resource Manager
7: Structure
2: Favorites