به نام خدا

پایگاه داده، بانک اطلاعاتی یا دیتابیس، هر سه اصطلاح در زبان فارسی برای Database بکار می رود. در این آموزش من بیشتر واژه دیتابیس را بکار می برم.

اما دیتابیس چیست؟ به طور خلاصه به مجموعه‌ای از داده‌ها با ساختار منظم و سامان‌مند گفته می‌شود (برگرفته از ویکی پدیا). دسترسی و مدیریت این اطلاعات عموما از طریق سیستم مدیریت پایگاه داده (DBMS) صورت می پذیرد. از دیتابیس برای ذخیره سازی و مدیریت اطلاعات استفاده می شود. امروزه اکثر نرم افزارها و اپلیکیشن ها برای مدیریت داده های خود از دیتابیس استفاده می کنند. کاربر به واسطه نرم افزار یا اپلیکیشن می تواند اطلاعاتی را بر روی دیتابیس اضافه کرده، مشاهده کند، بروزرسانی و یا حذف نماید. البته نه به این معنی که کاربر مستقیما با دیتابیس سروکار داشته باشد، بلکه این وظیفه بر عهده نرم افزار واسط و در نهایت همان DBMS است. یک اپلیکیشن یادداشت را در نظر بگیرید. کاربر می تواند متنی را به عنوان یادداشت درون آن ذخیره کند. همچنین اگر نیاز به اصلاح داشت آن را ویرایش و بروزرسانی نماید. یا هر زمان که بخواهد یادداشتی را که قبلا ذخیره کرده مشاهده نموده و در نهایت این امکان را دارد تا یادداشت خود را حذف نماید. کاری که DBMS یا همان سیستم مدیریت پایگاه داده انجام می دهد، دریافت همین دستورات از کاربر و اعمال آنها بر روی دیتابیس می باشد. شاید با خود بگویید "خب چرا دیتابیس؟ اطلاعات را به سادگی می توان درون یک فایل متنی ساده txt هم ذخیره کرد!" بله نظر شما صحیح است. در موارد محدودی هم ممکن است افراد از این روش استفاده کنند اما به دلایل زیادی ذخیره سازی اطلاعات درون فایل متنی گزینه مناسبی نیست. یک دلیل آن، عدم کد گذاری روی اطلاعات قبل از ذخیره سازی می باشد که از نظر امنیتی می تواند مشکلاتی ایجاد کند. اگر مایلید اطلاعات بیشتری در خصوص تفاوت بین ذخیره در فایل متنی و دیتابیس داشته باشید با اندکی جستجو در وب به پاسخهای متعددی دست خواهید یافت.

دیتابیس در همه جا کاربرد دارد. از اپلیکیشن های مختلف گوشی های هوشمند گرفته تا نرم افزارهای کاربردی رایانه ها، سرورهای بانکها، سازمانها و ... .

احتمالا واژه های MySQL و SQL Server را قبلا چندین بار دیده و یا شنیده اید. اینها نمونه هایی از نرم افزارهای مدیریت پایگاه داده از نوع RDBMS (Relational Database Management System) هستند که بر اساس مدل رابطه ای (Relational) پیاده سازی شده اند، به این معنی که اطلاعات به صورت جداولی شامل چندین ردیف و ستون ذخیره و نگهداری می شوند.

شناسه	نام	نام خانوادگی	سن
۱	مهدی	جرجانی	۲۶
۲	مریم	محمدی	۲۴

جدول بالا نمونه ساده یک دیتابیس است که شامل ۴ ستون و ۲ ردیف می باشد. اطلاعات جدول (ردیف ها) توسط کاربر و به واسطه نرم افزار مدیریت دیتابیس در هر زمان قابل اضافه، ویرایش و حذف شدن است. بجز MySQL و SQL Server نمونه های دیگری از RDBMS ها برای توسعه دهندگان در دسترس هستند مانند ORACLE، MS ACCESS و SQLite که هرکدام بنا به ویژگی های خاص خود، برای مقاصد خاصی طراحی شده اند. به عنوان مثال از ORACLE (اوراکِل) برای بانکهای اطلاعاتی با حجم بسیار بالا استفاده می شود (مانند بانکها و سازمان ها). یا اگر تابحال با سیستم های مدیریت محتوای وب کار کرده اید احتمال زیاد با MySQL آشنایی مختصری دارید. عموما از MySQL برای وب سایتهای با حجم داده در حد کم و متوسط استفاده می شود. اما وجه اشتراک موارد بالا در SQL می باشد. SQL مخفف Structured Query Language و معنی لغوی آن "زبان ساختارمند پرسش ها" است. زبان SQL یک زبان استاندارد بوده که شامل دستوراتی برای مدیریت اطلاعات دیتابیس ها می باشد. از جمله دستورات کلیدی SQL می توان به INSERT، UPDATE، DELETE، SELECT و DROP اشاره کرد.

چهار عمل اصلی پایگاه داده ها Create (ایجاد)، Read (خواندن)، Update (بروزرسانی) و Delete هستند که به اختصار CRUD نامیده می شود.

به احتمال زیاد مفاهیم بالا مقداری شما را گیج کرده است. اما جای نگرانی نیست. اولا اینکه در عمل ما با مفاهیم سروکار نداریم و صرفا از این جهت به خلاصه ای از پایگاه داده ها اشاره کردم تا یک زمینه کلی در ذهنتان ایجاد شود. دوم اینکه دانستن این اطلاعات درک عمیق تری از پایگاههای داده به شما می دهد و چه خوب است در کنار مباحث اصلی دیتابیس، اندکی هم در وب در خصوص دیتابیس ها مطالعه کنید.

بهتر است بیشتر از این وقت شما را نگیرم و بروم سراغ اندروید. در پاراگراف بالا اشاره ای شد که هرکدام از برنامه های مدیریت بانک اطلاعاتی بنا به ویژگی هایی که دارد، در مقاصد خاصی استفاده می شود. در دیوایس های اندروید که عموما گوشی ها و تبلت ها هستند، دیتابیس ها حجم بسیار کمی را در بر می‌گیرند بنابراین لازم است است از برنامه ای استفاده شود که کمترین میزان از منابع سخت افزاری دیوایس از جمله حافظه RAM و CPU را درگیر کند و حجم خود برنامه نیز تا حد ممکن کم باشد. گوگل SQLite را برای اندروید برگزیده و به صورت پیش فرض درون این سیستم عامل تعبیه شده و شما به عنوان توسعه دهنده اپلیکیشن نیازی به نصب SQLite بر روی دیوایس کاربر ندارید. ما فقط دیتابیس

را ایجاد نموده و به واسطه دستورات SQL مدیریت می کنیم. SQLite یک پایگاه داده بسیار کوچک با حجمی کمتر از یک مگابایت می باشد که در قالب یک کتابخانه (Library) نوشته شده و به صورت اوپن سورس و رایگان منتشر شده است. بنابراین گوگل یا توسعه دهنده نیازی نیست برای استفاده از این پایگاه داده مبلغی را به سازنده بپردازند. همچنین بر خلاف دیتابیسی مانند MySQL که نیاز به سرور دارد، SQLite بی نیاز از سرور بوده و به صورت مستقل روی هر دیوایس مستقر شده که اصطلاحا ServerLess نامیده می شود. بهتر است قبل از اینکه سراغ استفاده از دیتابیس در اندروید برویم قدری با این پایگاه داده و زبان SQL در محیط گرافیکی و ویژوال آشنا شویم تا درک بهتری از مفاهیم کسب کنیم. ابزارهای مختلفی برای کار با SQLite منتشر شده اند که امکانات و محیط کار همگی تا حدی مشابه یکدیگر است و امکان ساخت دیتابیس، جداول، ستون ها، ورود داده ها، حذف داده ها و ... در همه ابزارها محیط گرافیکی و ساده وجود دارد. مزیت این ابزارها در این است که تغییراتی که به صورت ویزاردی کاربر روی دیتابیس انجام می دهد را می تواند در نهایت در قالب چند خط کد که همان زبان SQL هست مشاهده کند. یا اینکه دستوراتی را وارد کند و نتیجه درستی یا نادرستی کد خود را سریعا مشاهده کند. ابزارهایی مانند DB Browser for SQLite ، SQLiteStudio و SQLite Expert Professional که البته مورد آخر رایگان نبوده و اگر مایل به استفاده از آن هستید باید لایسنس را خریداری کرده و یا از نسخه های نال شده استفاده کنید. اما در حدی که ما نیاز داریم ابزارهای رایگان مناسب هستند. حتی پلاگینی به این منظور برای مرورگر FireFox به صورت رایگان منتشر شده که با نصب آن روی فایرفاکس می توانید به مدیریت دیتابیس های SQLite بپردازید. این پلاگین SQLite Manager نام دارد. محیط کاربری این ابزارها تا حدودی مشابه یکدیگر بوده و پیچیدگی را احساس نخواهید کرد.

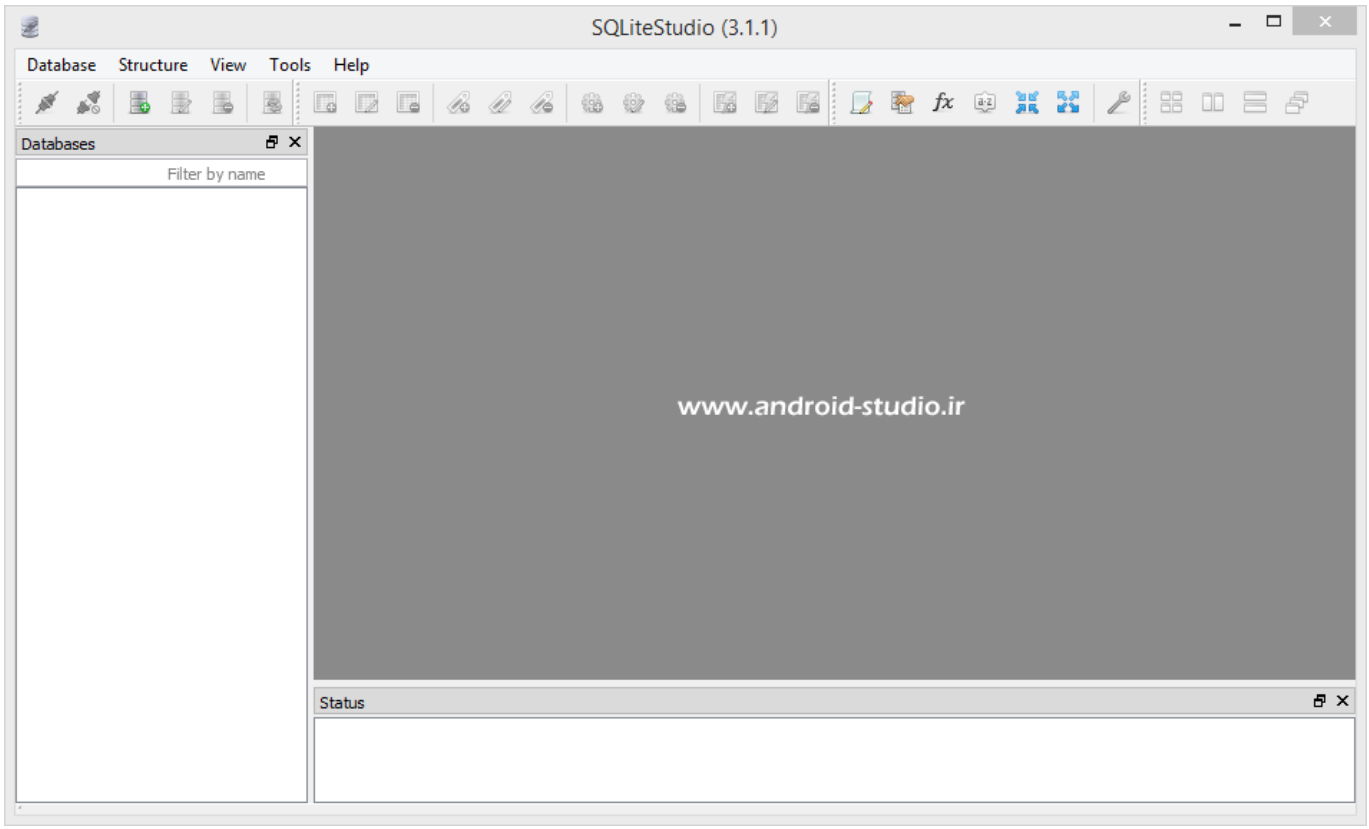
در این آموزش من از SQLiteStudio استفاده می کنم.

تاکید می کنم استفاده از این ابزار در این مبحث آموزشی فقط برای آشنایی با دیتابیس و دستورات SQL و اطمینان از صحت دستوراتی که بنا به نیازمان می نویسیم بوده و قرار نیست ما دیتابیس خروجی این ابزار را به پروژه اندرویدی خود منتقل کنیم.

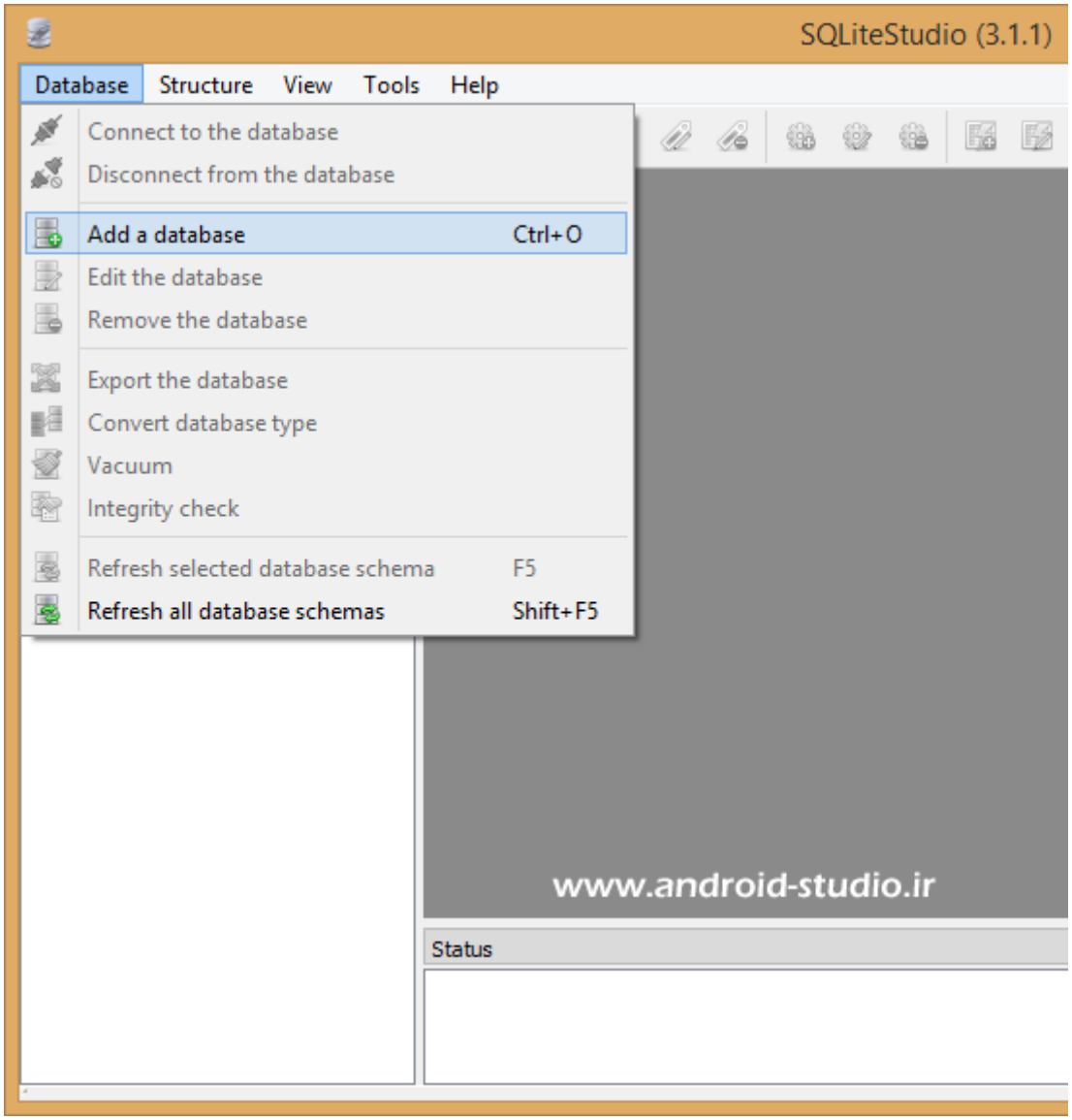
من SQLiteStudio را دانلود و از حالت فشرده خارج کردم. بر خلاف سایر ابزارها، SQLiteStudio نیاز به نصب ندارد و و از نوع Portable است. کافیست SQLiteStudio.exe را اجرا کنید. اگر مثل من به تمیز بودن دسکتاپ خود اهمیت می دهید و تمایل دارید دسترسی سریع به این نرم افزار داشته باشید بهتر است پوشه مربوط به این نرم افزار را در یک درایو قرار داده و فقط از فایل SQLiteStudio.exe یک Shortcut ساخته و به دسکتاپ انتقال دهید.

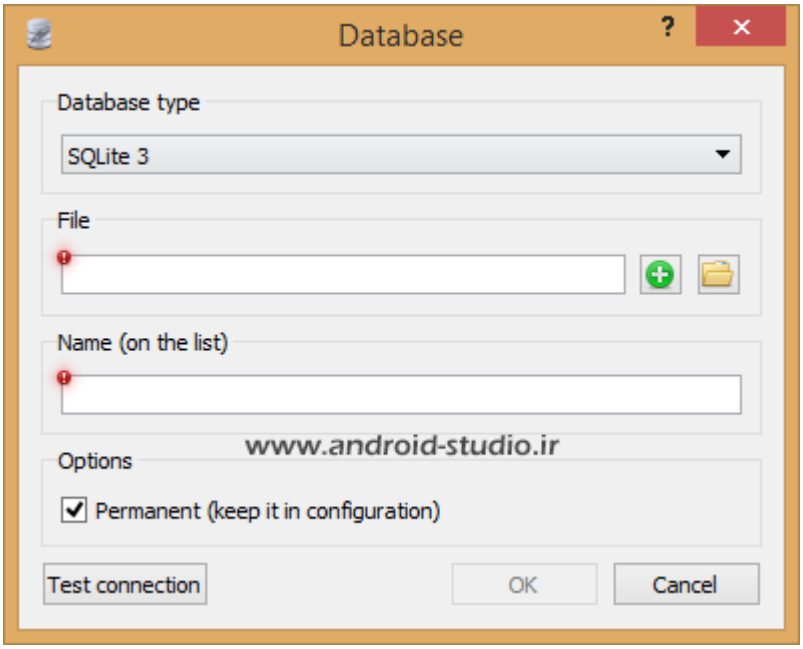
تذکر: امیدوارم این ابزار مدیریت دیتابیس در محیط دسکتاپ را با برنامه های مدیریت دیتابیس (DBMS) اشتباه نگیرید!

SQLiteStudio را باز می کنم.

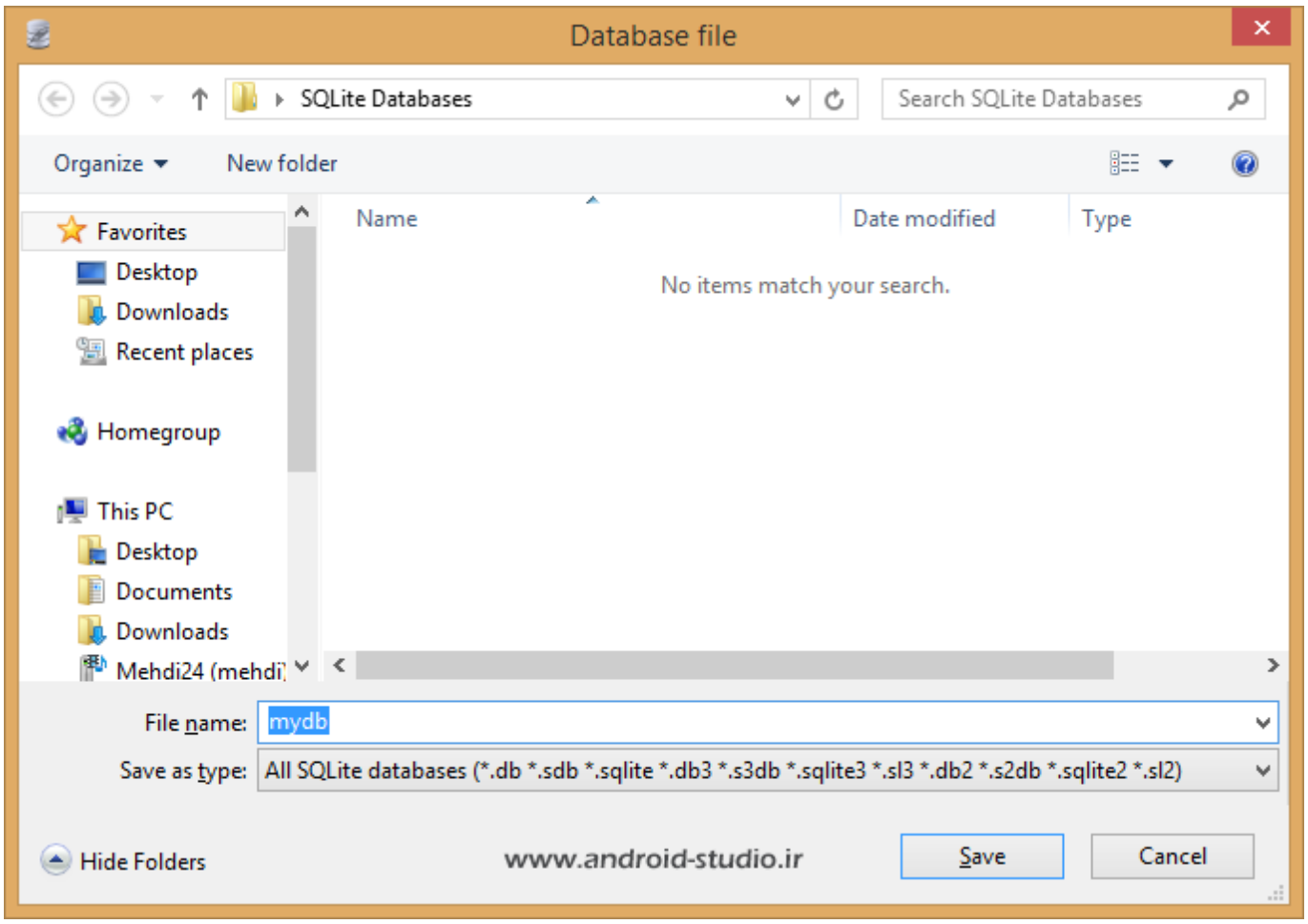


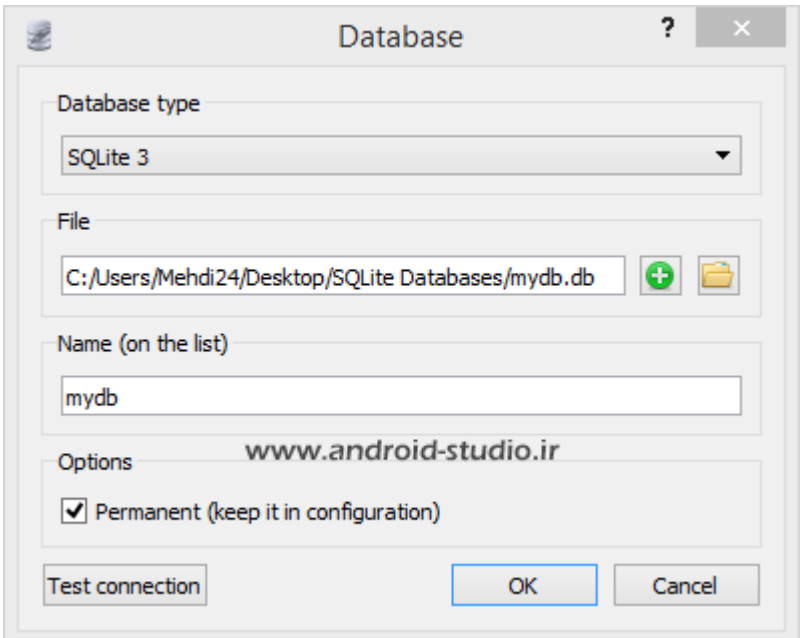
اولین قدم ساخت یک دیتابیس جدید است که در منوی Database و گزینه add a new database قابل دسترسی است:


SQLiteStudio (3.1.1)
Database
Structure
View
Tools
Help
Connect to the database
Disconnect from the database
Add a database Ctrl+O
Edit the database
Remove the database
Export the database
Convert database type
Vacuum
Integrity check
Refresh selected database schema F5
Refresh all database schemas Shift+F5
www.android-studio.ir
Status

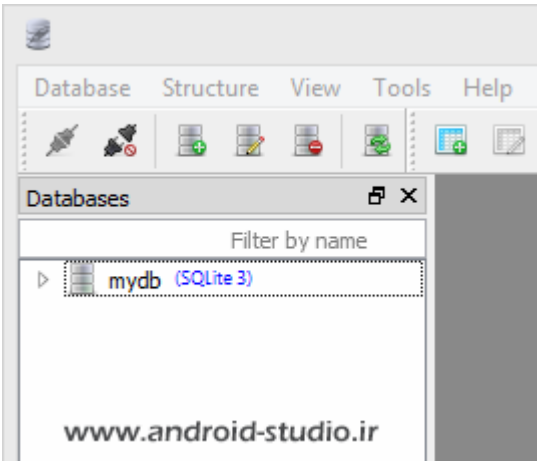

Database
Database type
SQLite 3
File
Name (on the list)
www.android-studio.ir
Options
Permanent (keep it in configuration)
Test connection
OK
Cancel

قسمت Database type مربوط به ورژن SQLite بوده که در زمان نگارش این آموزش نسخه ۳ به صورت پیش فرض انتخاب شده و نیازی به تغییر ندارد. در قسمت File روی گزینه + کلیک کرده و در مسیر مدنظر خود، نامی دلخواه برای فایل دیتابیس تعیین می کنیم:



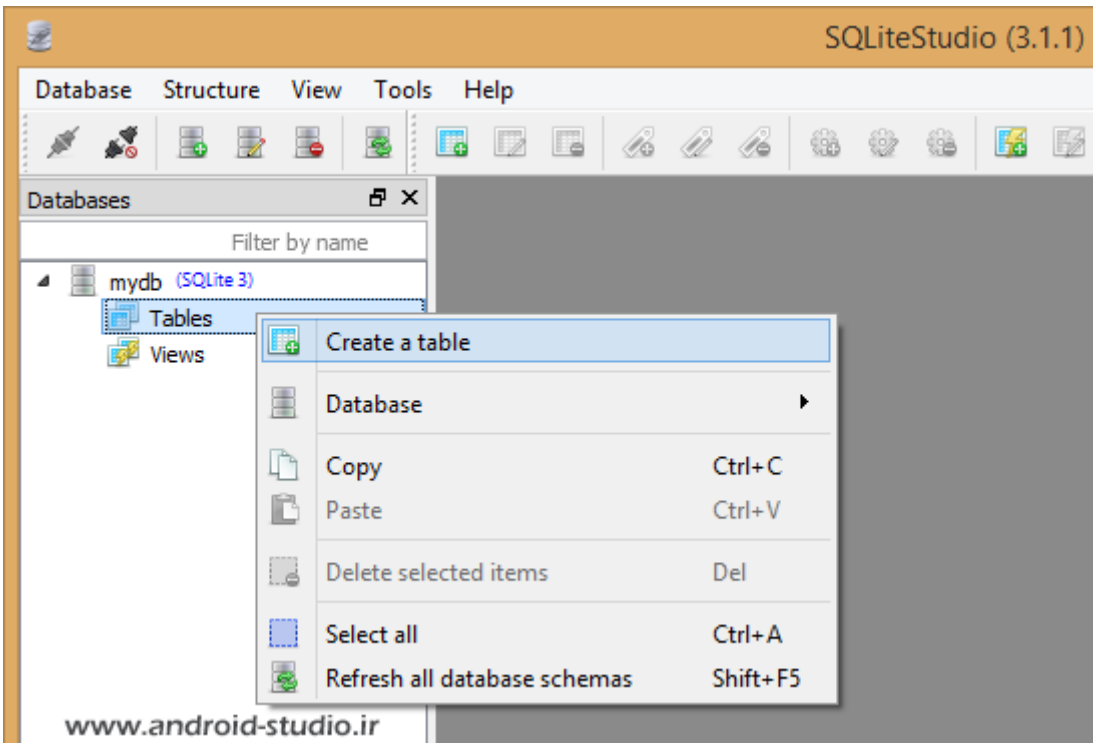


دیتابیس با نام mydb و پسوند db. آماده ساخت است. OK می کنم:

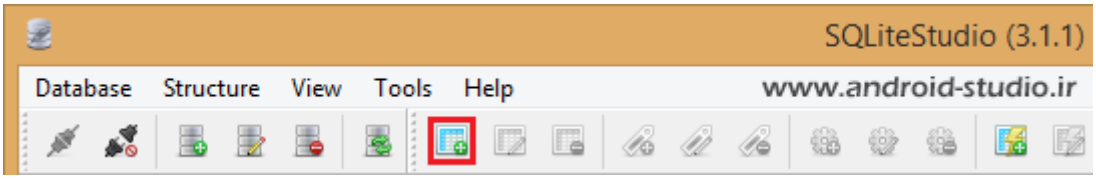


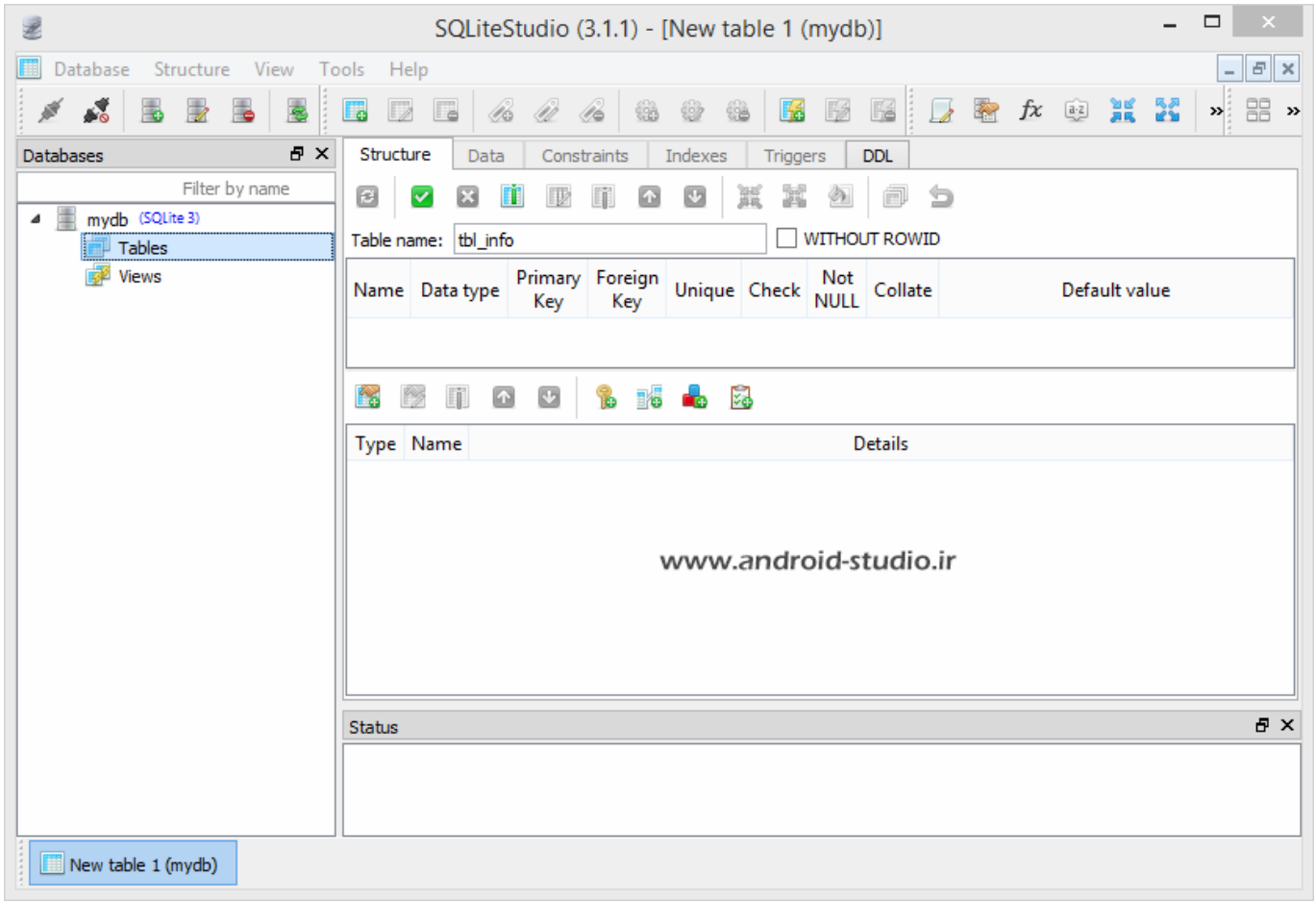
دیتابیس ایجاد شده به لیست Databases اضافه شد. با باز کردن دیتابیس mydb گزینه Tables (جدول ها) مشاهده می شود. در یک دیتابیس به هر تعداد لازم می توان جداول مجزا ایجاد نمود. یک جدول جدید می سازم. ساخت جدول از دو طریق ممکن می باشد:

اول: راست کلیک روی Tables و گزینه Create a Table

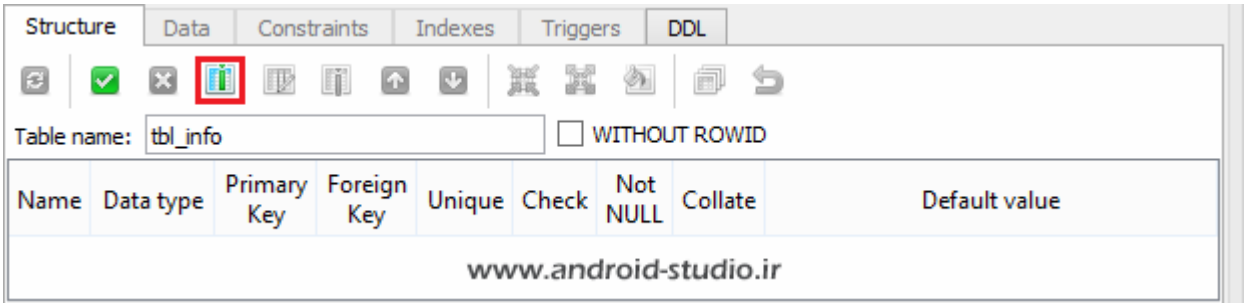


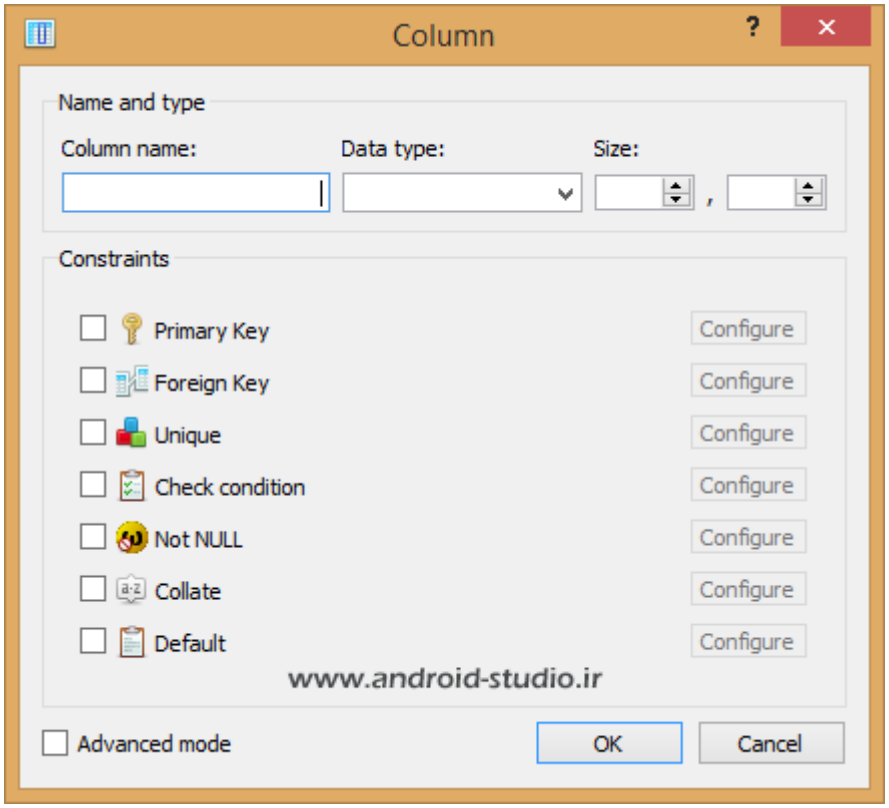
دوم: آیکون جدول





در قسمت Table name یک نام برای جدول تعیین می کنیم. من نام tbl_info را وارد کردم. قبل از ساخت جدول حداقل یک ستون می بایست به جدول اضافه کنیم. ساختار دیتابیس و جدول را همین ستون ها تشکیل می دهند. عموما برای هر ردیف از اطلاعات یک شناسه (ID) اختصاص می دهیم که این شناسه به صورت افزایشی و غیر قابل تکرار می باشد. با زدن گزینه Add column پنجره ای باز می شود که مشخصات ستون را باید وارد کنیم:



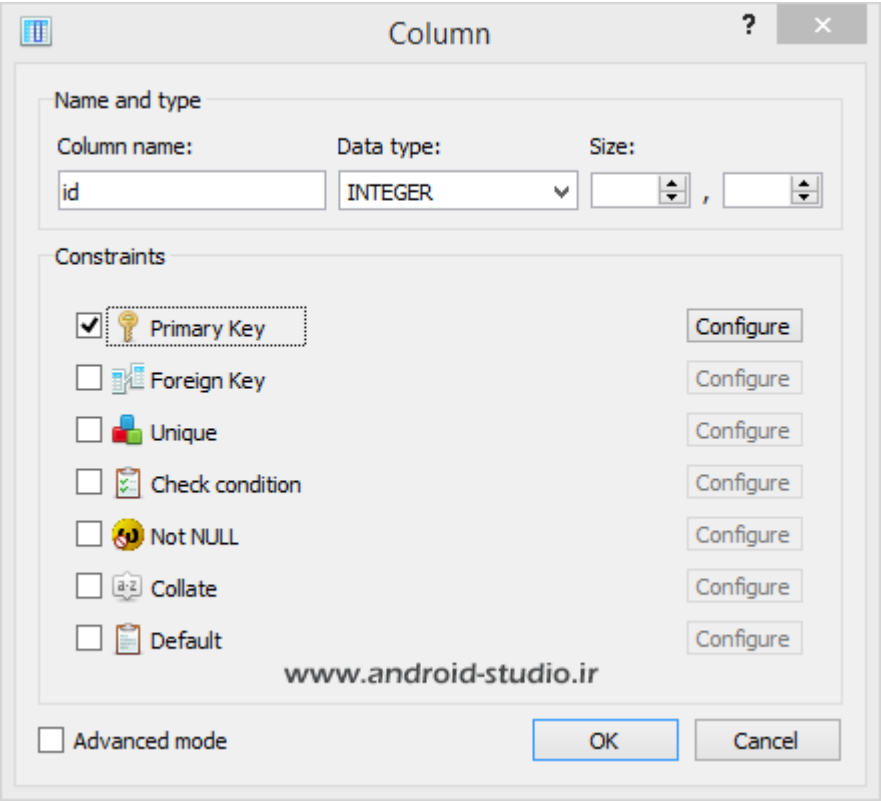


برای فیلد Column name مقدار (نام) **دلخواه** id و Data type را از نوع INTEGER انتخاب کردم. با نوع داده ها قبلا آشنا شده اید. شناسه از نوع اعداد صحیح بوده و بنابراین می بایست نوع integer را انتخاب کنیم.

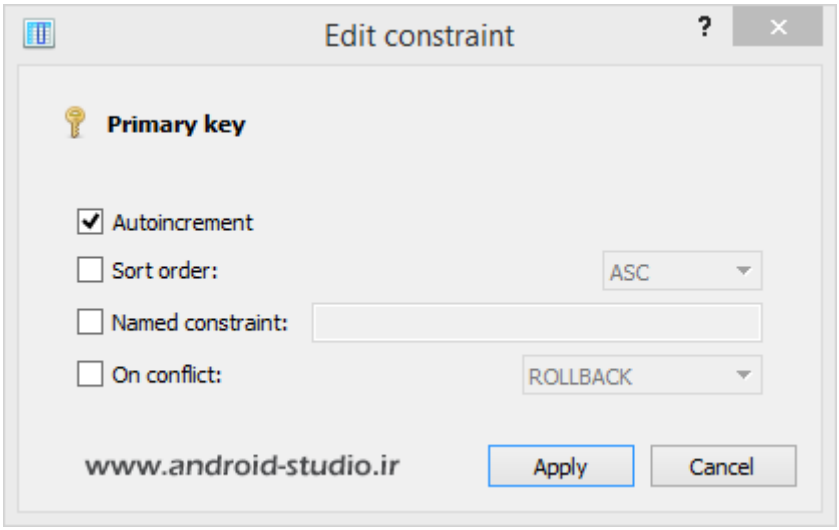
نکته : اگر لازم است بیشتر در مورد نوع داده ها بدانید با یک جستجو در وب به پاسخ های زیادی می رسید. جستجو در وب و یا به اصطلاح سرچ، سلاح همیشه همراه برنامه نویسان، توسعه دهندگان و طراحان وب است! حتی حرفه ای ترین افراد هم در زمان کار روی پروژه های خود نیاز به سرچ دارند. همیشه جزئیاتی هستند که به علت حجم بالای اطلاعات از ذهن انسان پاک شده و نیاز به مرور دارد. در بیشتر موارد رجوع به وب سریعتر شما را به نتیجه می رساند تا گشتن در میان داکیومنت ها و ویدئوهای آموزشی که روی حافظه رایانه خود دارید. به عنوان مثال برای مورد بالا با جستجوی " Data types in Database"، "Data types in SQL"، "Data types in SQlite"، "انواع داده ها در SQL"، "انواع داده ها در دیتابیس"، "انواع داده ها در برنامه نویسی" و ... به نتایج خوبی می رسید که در عرض چند دقیقه

با مطالعه ۲ یا ۳ مطلب پاسخ های مناسبی دریافت می کنید. عمدا تمام مواردی که برای جستجو در این خصوص به ذهنم رسید را لیست کردم تا تصور نکنید عبارت جستجوی شما باید خیلی سنجیده و مشخص باشد. ضمن اینکه در دنیای نرم افزار، مشابهات زیاد هستند و نیاز به جستجوی تخصصی در همه موارد نیست. برای مثال بعد از جستجو در خصوص "انواع داده ها" نیاز نیست حتما به دنبال مقاله ای بگردید که انواع داده ها را در دیتابیس و در SQLite شرح داده باشد چون داده ها در تمامی زمینه ها کاربرد یکسانی دارند (یعنی Integer در Java، C++، PHP، SQL و... مفهوم یکسانی دارد).

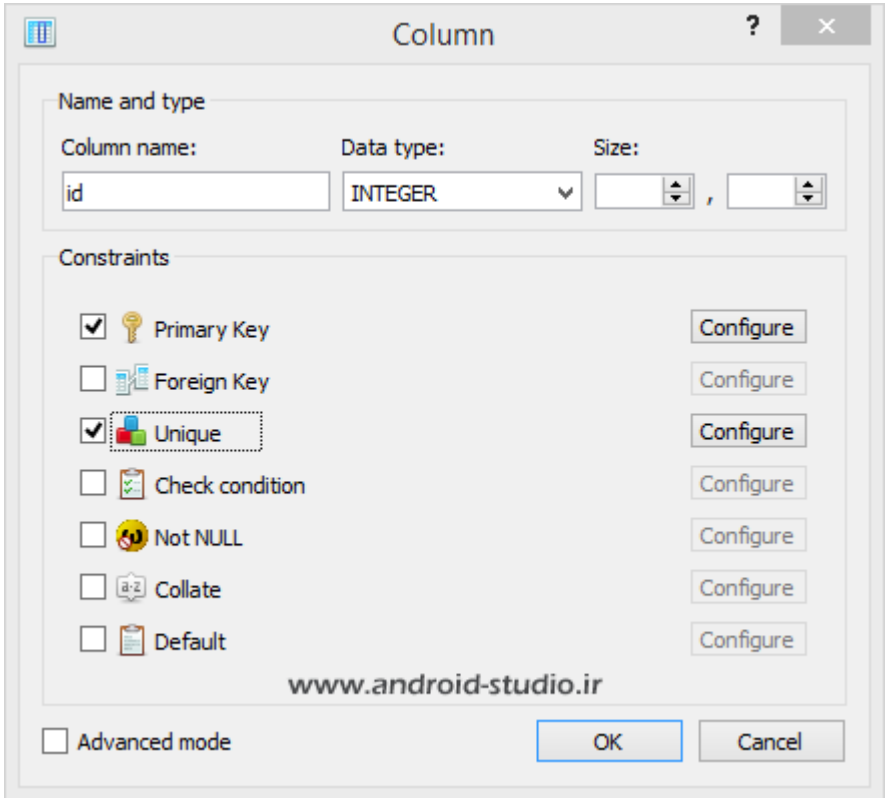
خب، از آنجایی که id ستون اصلی دیتابیس ماست با ویژگی هایی که در پاراگراف قبل اشاره کردم (افزایشی و غیر قابل تکرار) گزینه Primary Key را انتخاب می کنم:



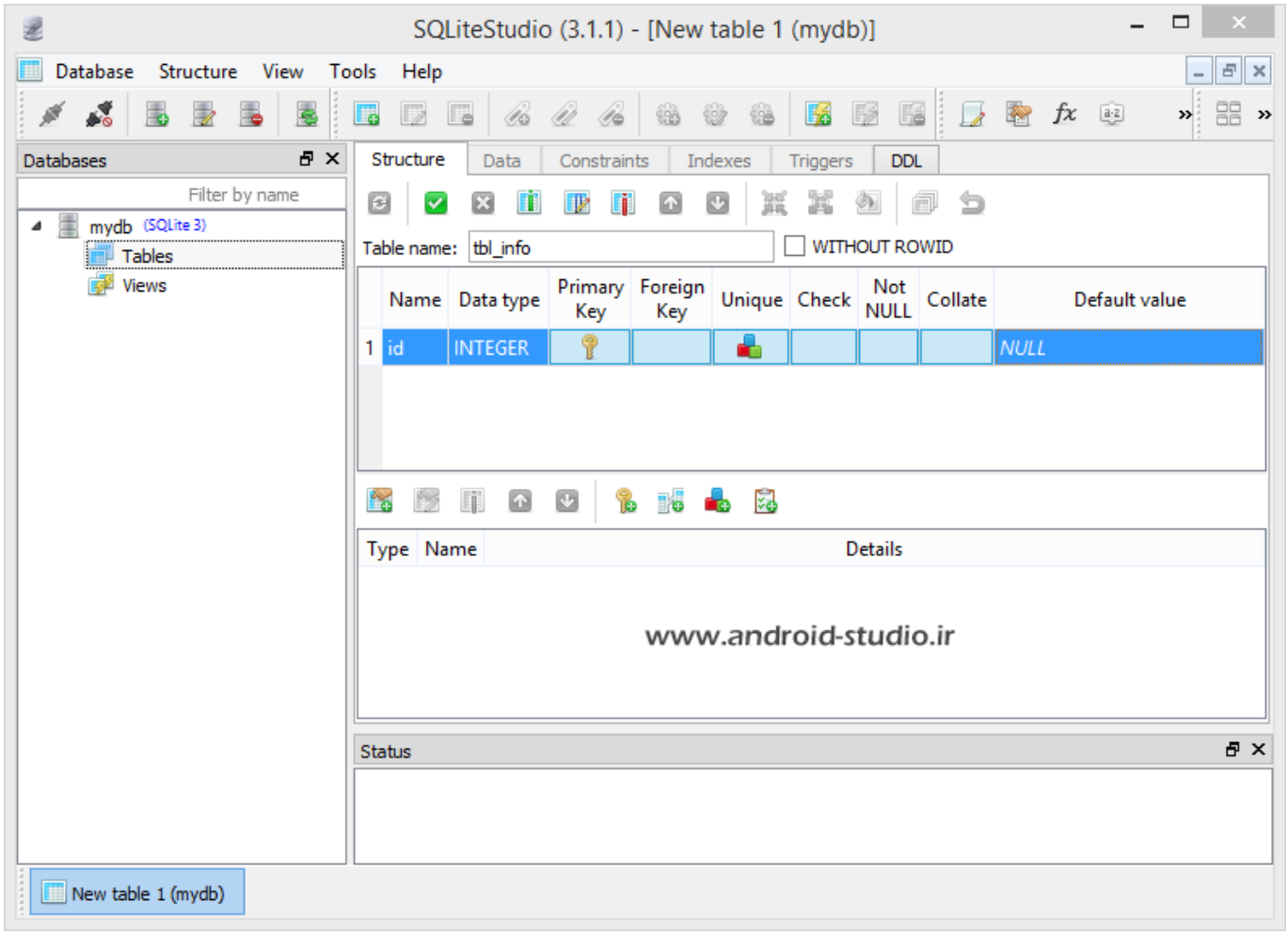
دکمه Configure برای این گزینه فعال شد. روی دکمه کلیک می کنم. گزینه اول Auto increment است که به معنی افزایش خودکار می باشد. شناسه هم باید این ویژگی را داشته باشد تا با اضافه شدن هر سطر از اطلاعات جدید، یک رقم به رقم شناسه قبلی به صورت خودکار اضافه شود. انتخاب کرده و Apply می کنم:



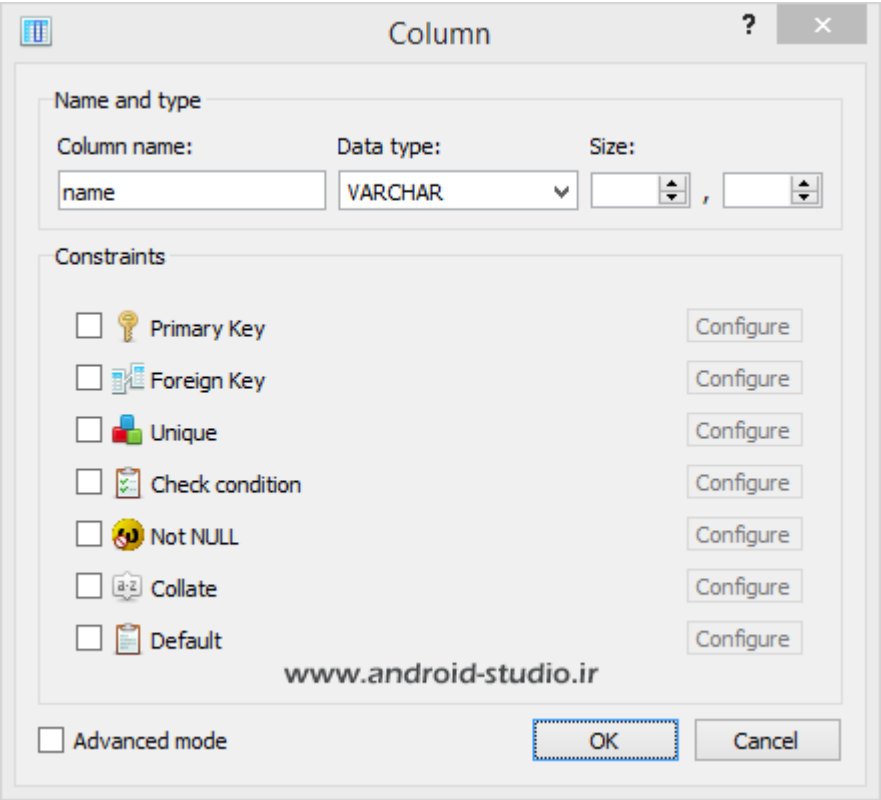
ویژگی دیگری که برای شناسه نام بردم یکتا بودن و تکراری نبودن شناسه ها بود. بنابراین گزینه Unique به معنی یکتا را انتخاب و OK می کنم.



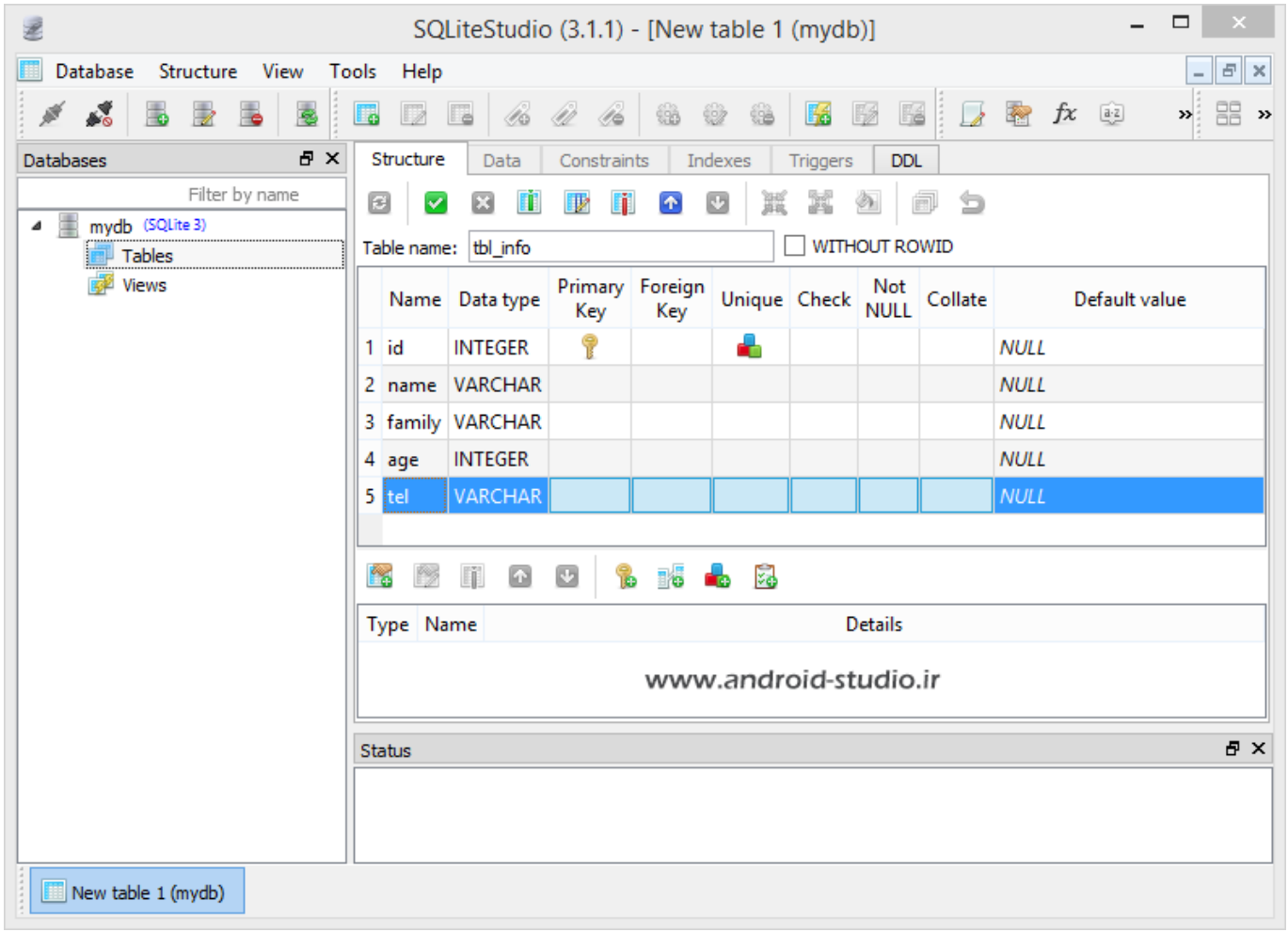
مشاهده می کنید ستون id با موفقیت به جدول اضافه شد:



البته من هنوز تغییراتم را ذخیره نکرده ام (گزینه Commit structure changes). مابقی ستون های مدنظرم را نیز اضافه می کنم. من چند ستون دیگر با عنوان نام، نام خانوادگی، سن و تلفن تماس مدنظر دارم که آنها را نیز اضافه می کنم.

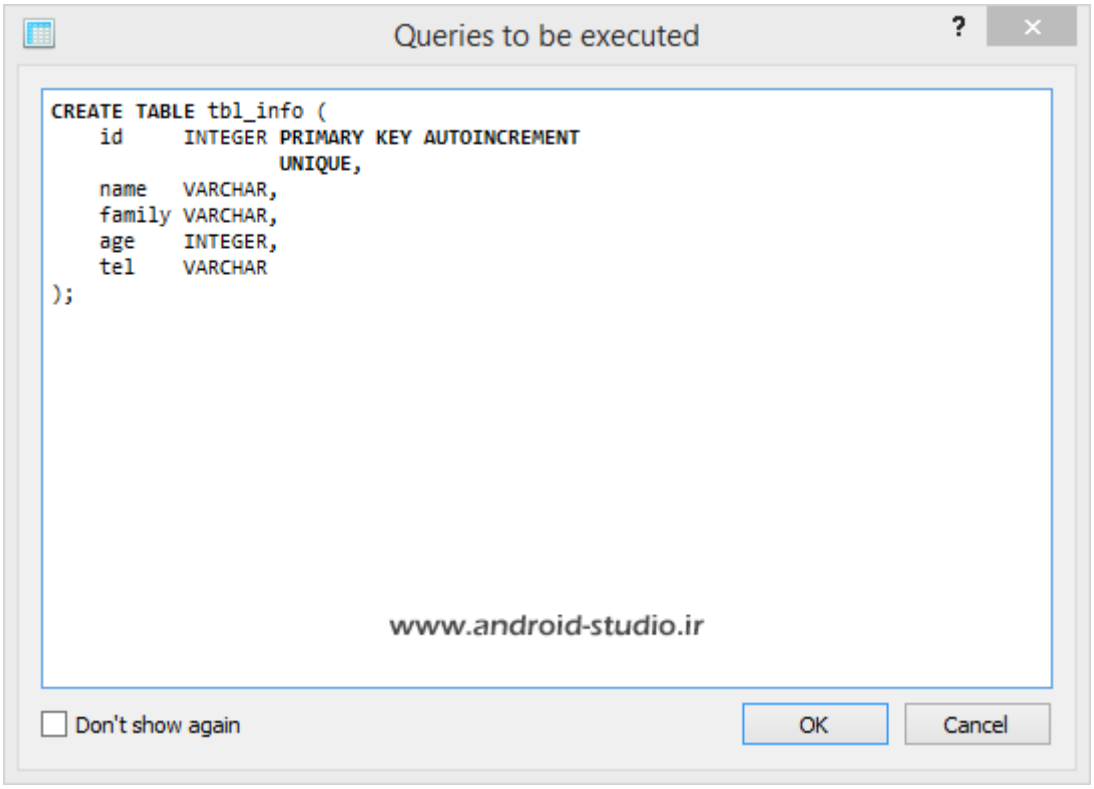


من name را از نوع VARCHAR تعریف کردم. VARCHAR مانند TEXT مقادیر رشته ای می پذیرد اما در موارد مانند نام و ... که تعداد کاراکتر محدودی توسط کاربر وارد می شود، بهتر است این نوع را انتخاب کنیم. نوع VARCHAR نسبت به TEXT فضای کمتری اشغال می کند. نوع TEXT برای رشته های طولانی مانند جملات و پاراگراف ها مناسب است. سایر گزینه ها را دست نخورده باقی می گذارم. به همین ترتیب برای نام خانوادگی یک ستون با نام family می سازم. ستون بعد سن، با نام age و از نوع INTEGER تعیین می کنم. آخرین ستون تلفن تماس بوده که منطقا باید از نوع INTEGER تعریف شود اما چون در داده های عددی، صفر اول نادیده گرفته می شود و یا ممکن است کاربر تلفن خود را به صورت 989158888888+ وارد کند که کاراکتر اول آن غیر عددی است، از نوع VARCHAR و با نام tel انتخاب می کنم.



مشاهده می کنید ستون ها با نام id، name، family، age و tel لیست شده اند. با دوبار کلیک روی هرکدام می توانید تغییراتی اگر نیاز بود را اعمال کنید. در نهایت با گزینه تیک سبز رنگ، مواردی که اضافه کردم را تایید می کنم (توجه داشته باشید بعد از تایید نهایی هم می توان تغییرات احتمالی را روی ستون ها اعمال کرد و یا ستون جدید به جدول اضافه و یا ستونی را حذف نمود).

با زدن گزینه تیک سبز رنگ، قبل از اعمال تغییرات، دستور یا Query مربوط به تغییراتی که قصد اعمال آنها را داریم نمایش داده می شود:



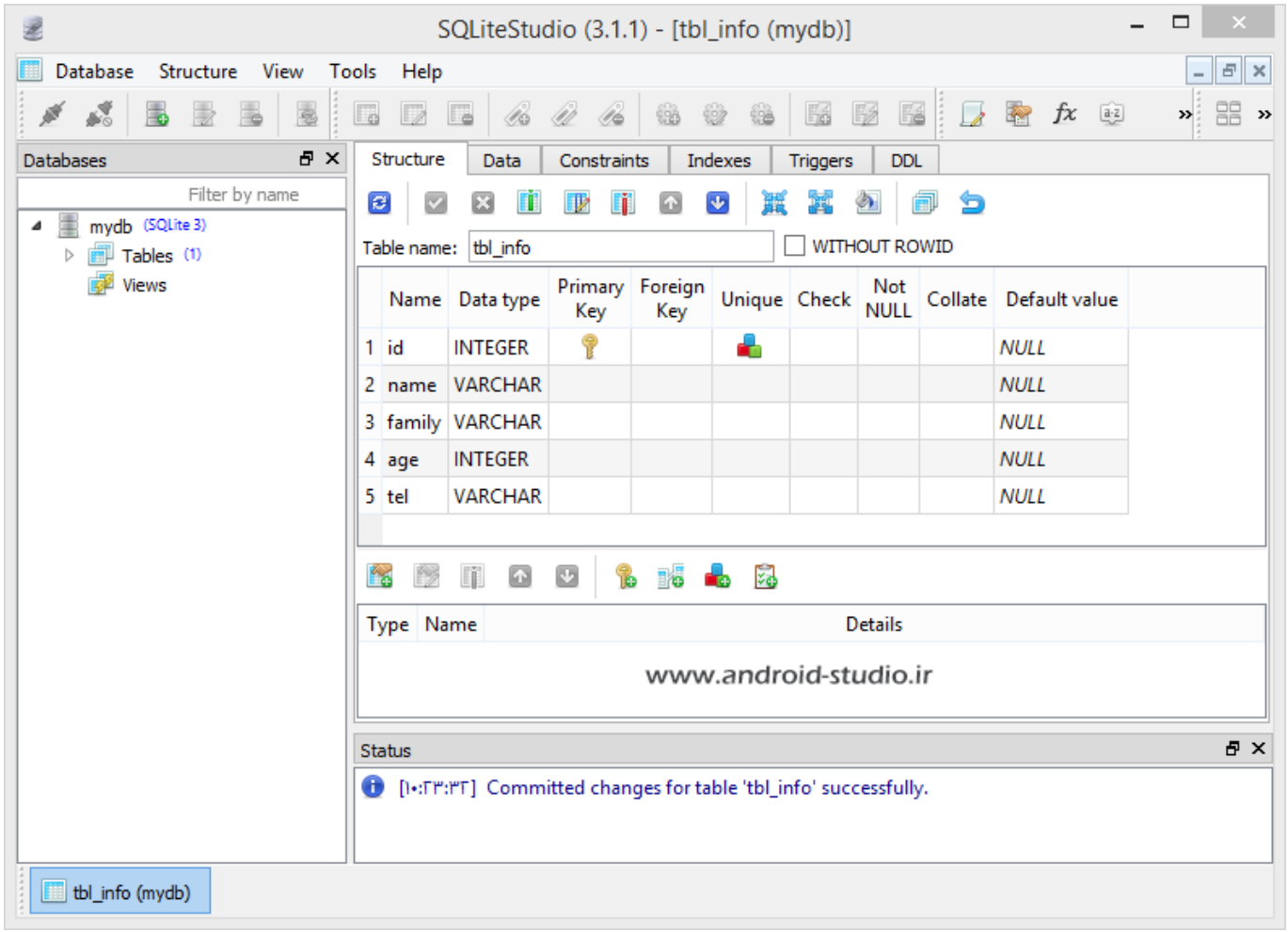
توضیح : Query (کوئری) در زبان فارسی با عنوان "پرس و جو" نیز شناخته می شود. کلیه اعمال مربوط به دیتابیس توسط کوئری ها انجام می شود.

این دقیقا همان چیزی است که در اندروید با آن سروکار داریم. Query را بخوانید. خیلی واضح است چه کارهایی قرار است روی دیتابیس ما انجام شود.

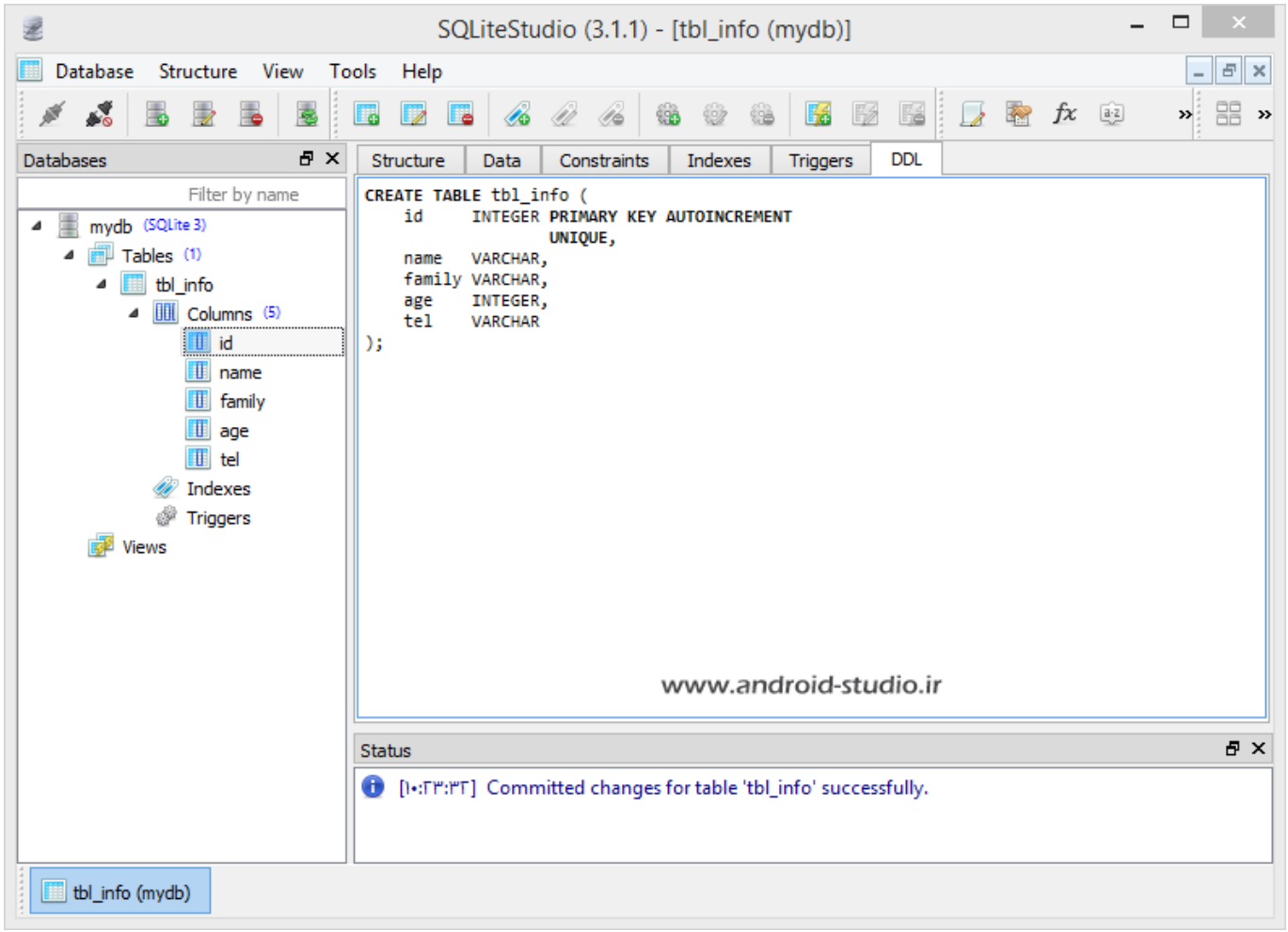
CREATE TABLE tbl_info: ساخت جدولی با نام tbl_info

دستورات مربوط به ستون های جدول نیز درون () قرار گرفته اند که هر خط نشان دهنده یک ستون است. دقیقا با همان تنظیماتی که مدنظر بود. یعنی ستون id از نوع INTEGER، PRIMARY KEY، AUTOINCREMENT و UNIQUE می باشد. هر کدام از ویژگی ها با یک فاصله از ویژگی قبلی جدا شده و هر ستون با یک ویرگول. لازم به ذکر است که در دستورات SQL تعداد فاصله ها، کوچک بودن یا بزرگ بودن حروف (بجز در نامگذاریها) و همچنین قرار گیری هر دستور در یک خط جداگانه یا پشت سر هم اهمیتی نداشته و صرفا به جهت خوانایی بهتر SQLiteStudio آنرا به صورت تفکیک شده و زیر هم نمایش داده است.

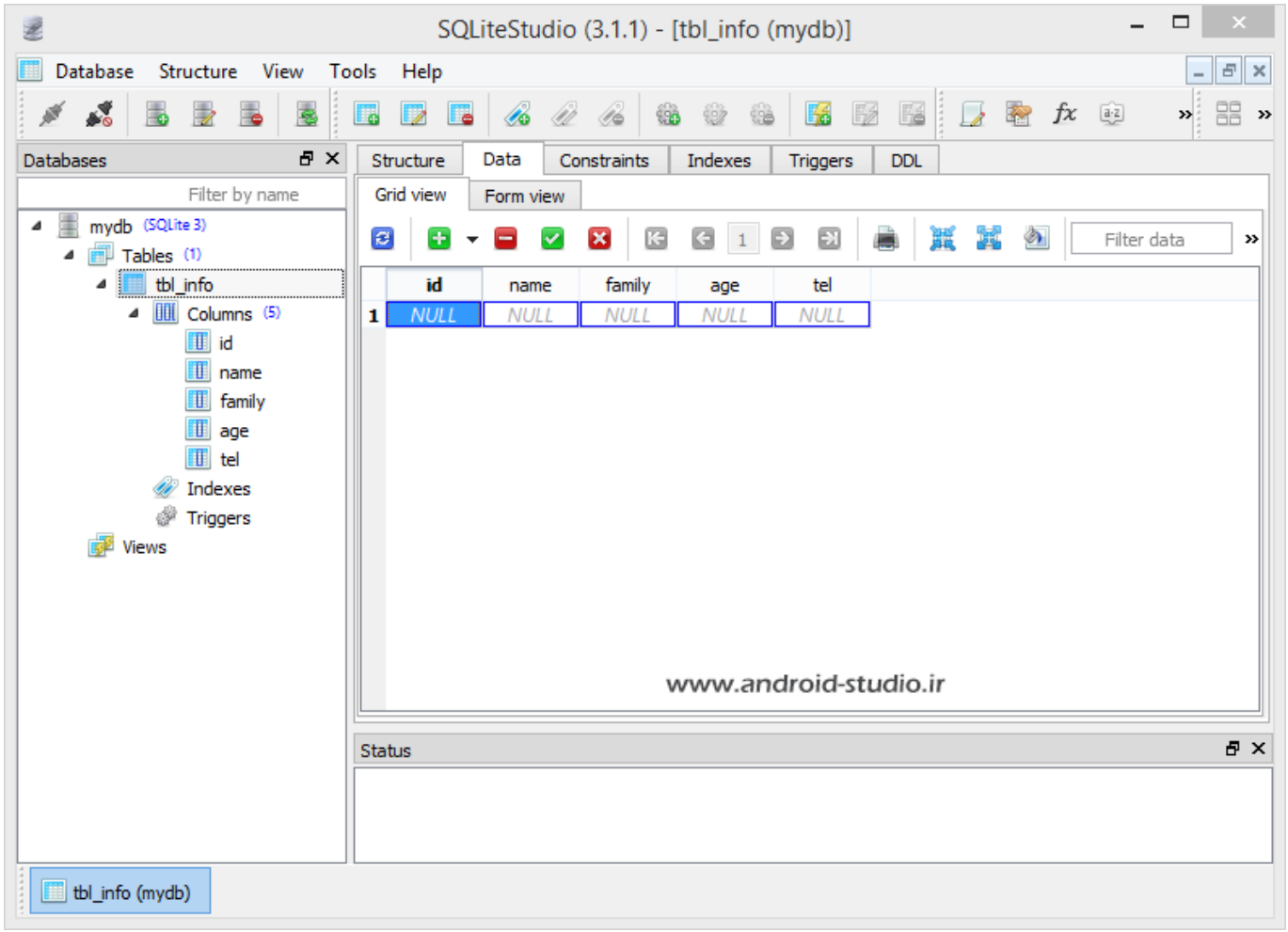
من Query را تایید کردم. جدول tbl_info به همراه ستون های آن با موفقیت ایجاد شد که در تب Structure نمایش داده شده و در کادر Status نیز پیغام موفقیت آمیز بودن تغییرات را دریافت کردم:



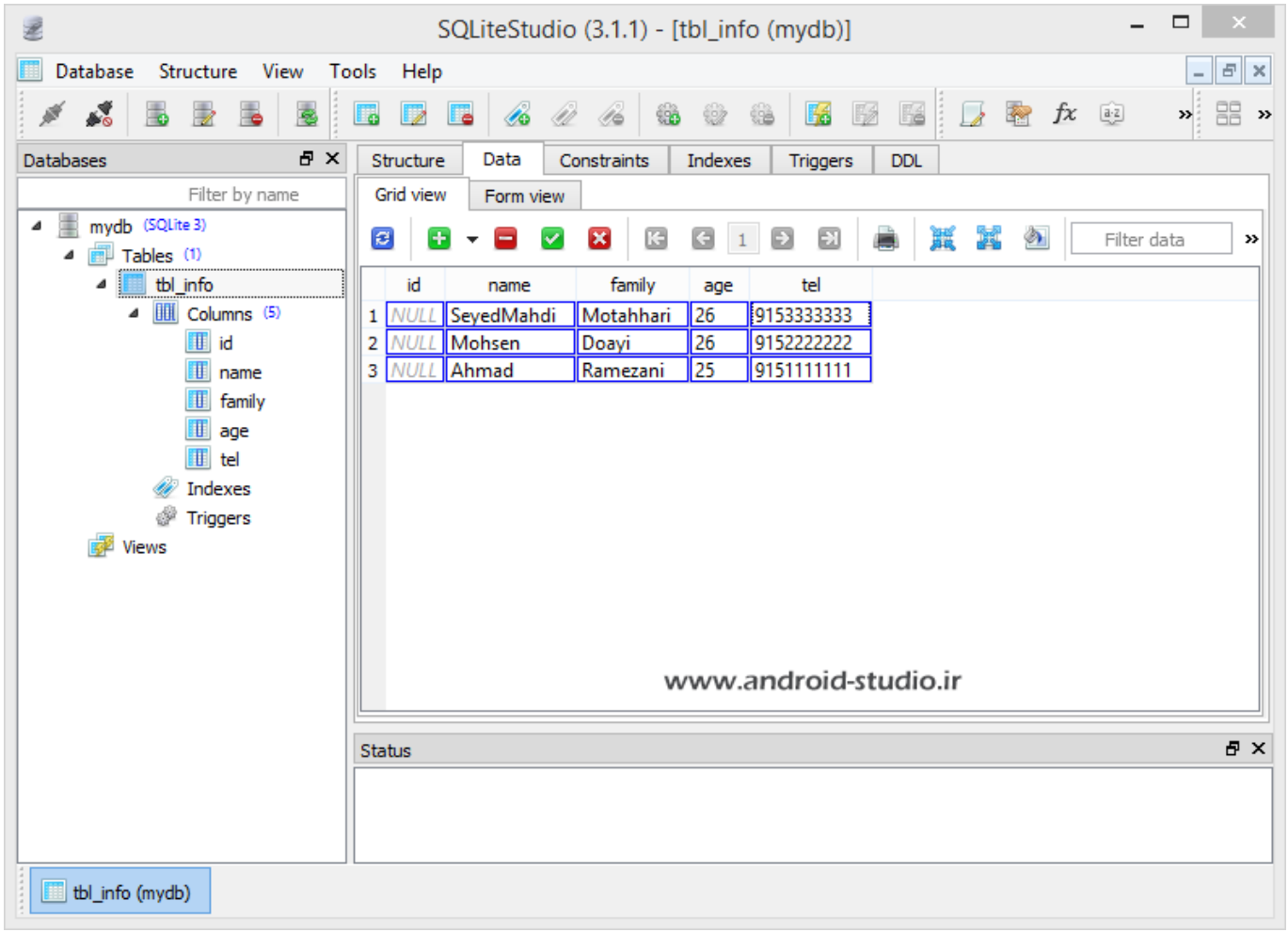
در تب DDL نیز همواره به Query نهایی ساختار دیتابیس خود دسترسی دارید و هرگاه که تغییری در ساختار جدول ایجاد کنید، این کد نیز بروز می شود:



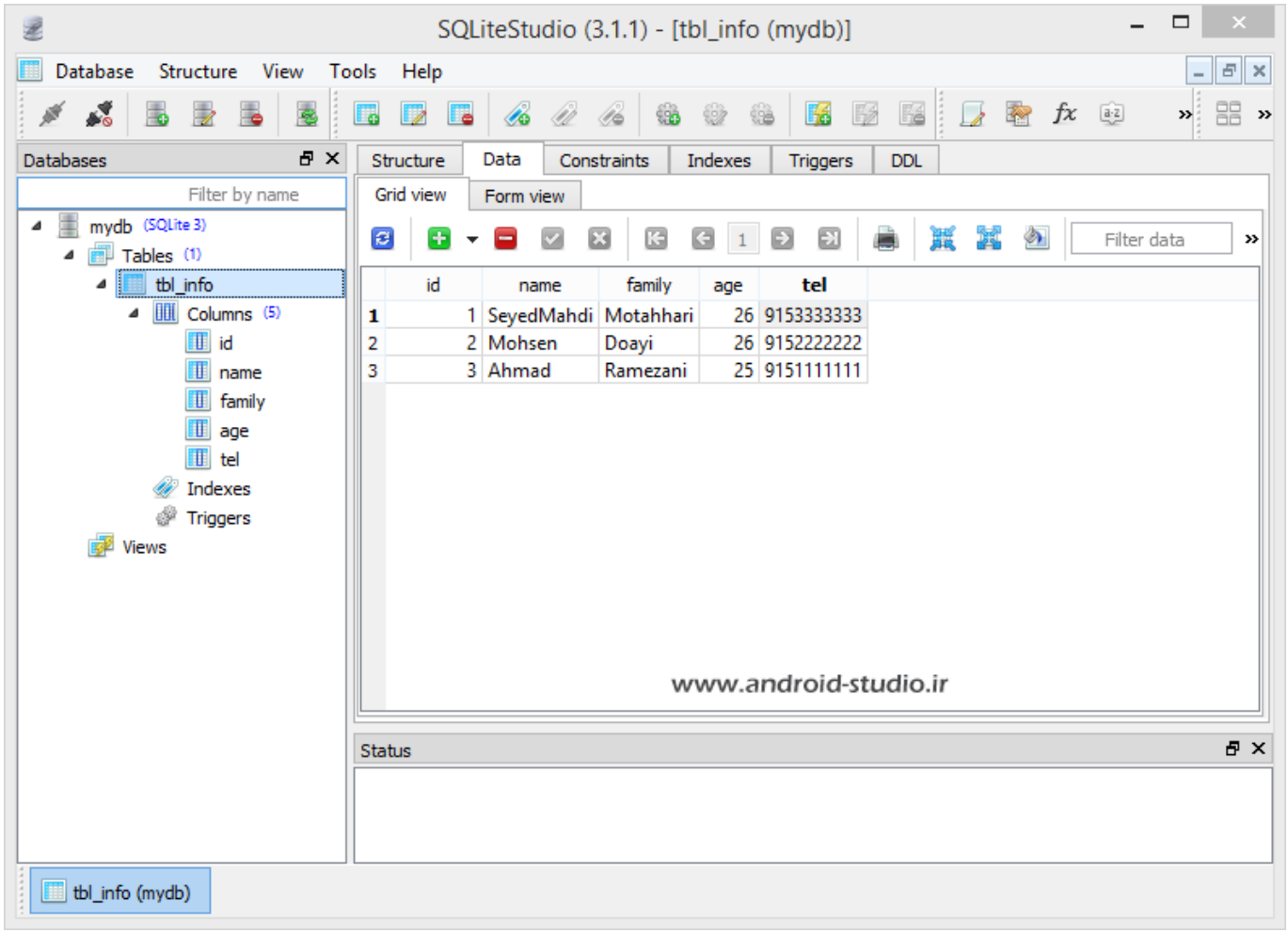
حالا قصد دارم اطلاعات چند شخص را وارد دیتابیس کنم. در برگه Data با زدن گزینه + سبز رنگ (و یا کلید insert کیبورد)، یک ردیف با مقادیر پیش فرض NULL (به معنی تهی) در ستونها ایجاد می شود که آماده وارد کردن اطلاعات است:



ستون id را به صورت Auto increment تعیین کرده بودیم پس من مقدار آن را همان null و دست نخورده باقی می گذارم و سراغ ستون های بعدی می روم. با کلید tab به راحتی می توانید بین ستون ها جابجا شوید و مقادیر را وارد کنید. من ۳ ردیف اطلاعات وارد کردم:



با Commit کردن تغییرات (گزینه تیک سبز رنگ) اطلاعات ثبت می شود:



نکته : در روش وارد کردن اطلاعات در محیط گرافیکی مانند SQLiteStudio، با وجود VARCHAR یا TEXT بودن نوع داده باز هم صفر اول اعداد حذف می شود که برای ما مهم نیست و در ادامه که اطلاعات را از طریق کوئری اضافه کنیم این مشکل رفع می شود.

حالا می خواهم کوئری های اصلی و ضروری را روی این دیتابیس اجرا کنم. کوئری های Insert، Update، Select و Delete.

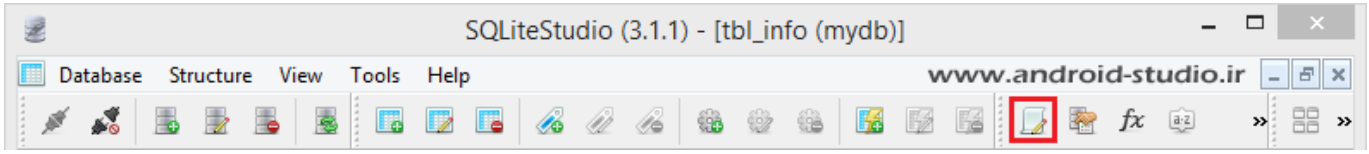
در قدم اول قصد دارم مشخصات یک شخص را بجای استفاده از محیط گرافیکی، از طریق کوئری insert وارد جدول کنم. Syntax دستور insert به اینصورت است:

```
INSERT INTO table_name (column1, column2, column3,...columnN) VALUES (value1, value2, value3,...valueN);
```

سینتکس چیست : نحو یا Syntax به معنای نحوه نگارش و قرارگیری کلمات و عبارات به صورت درست، در یک زبان می باشد. در علم کامپیوتر، سینتکس یک زبان کامپیوتری، مجموعه ای از قوانین می باشد که نحوه قرارگیری و ترکیب کلمات، نشانه ها و علایم یک زبان را به صورتی که معنای درستی بدهند مشخص می کند. (منبع: پارس دیتا)

مابین دو پرانتز ابتدایی نام ستون ها و در پرانتزهای بعد از VALUES (به معنی مقادیر)، مقادیر موردنظر را به ترتیب وارد می کنیم. به جای table_name هم نام دیتابیس جایگزین می شود.

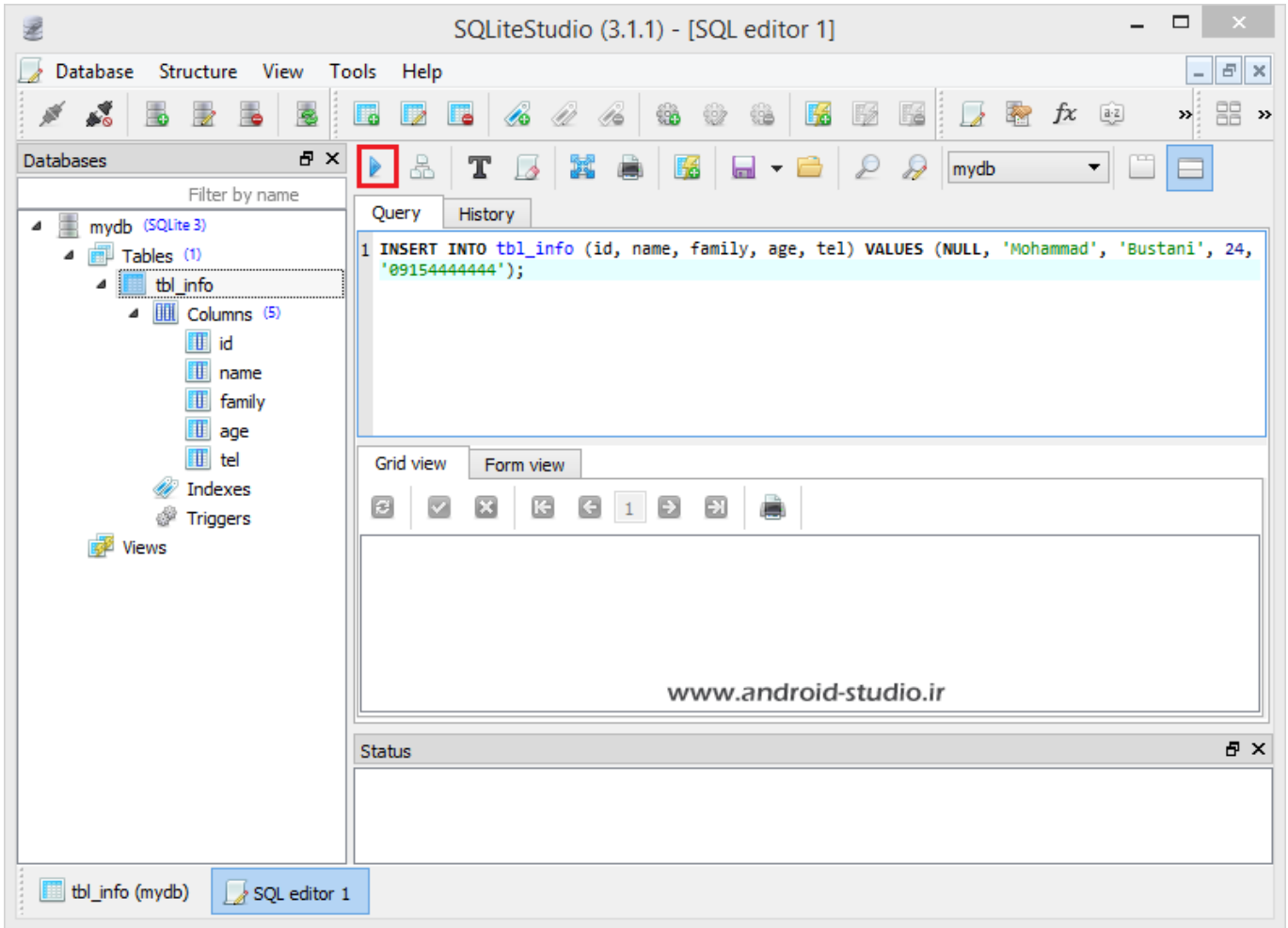
با زدن گزینه Open SQL editor یا کلید میانبر (Alt + E) به برگه ی SQL Editor می روم:

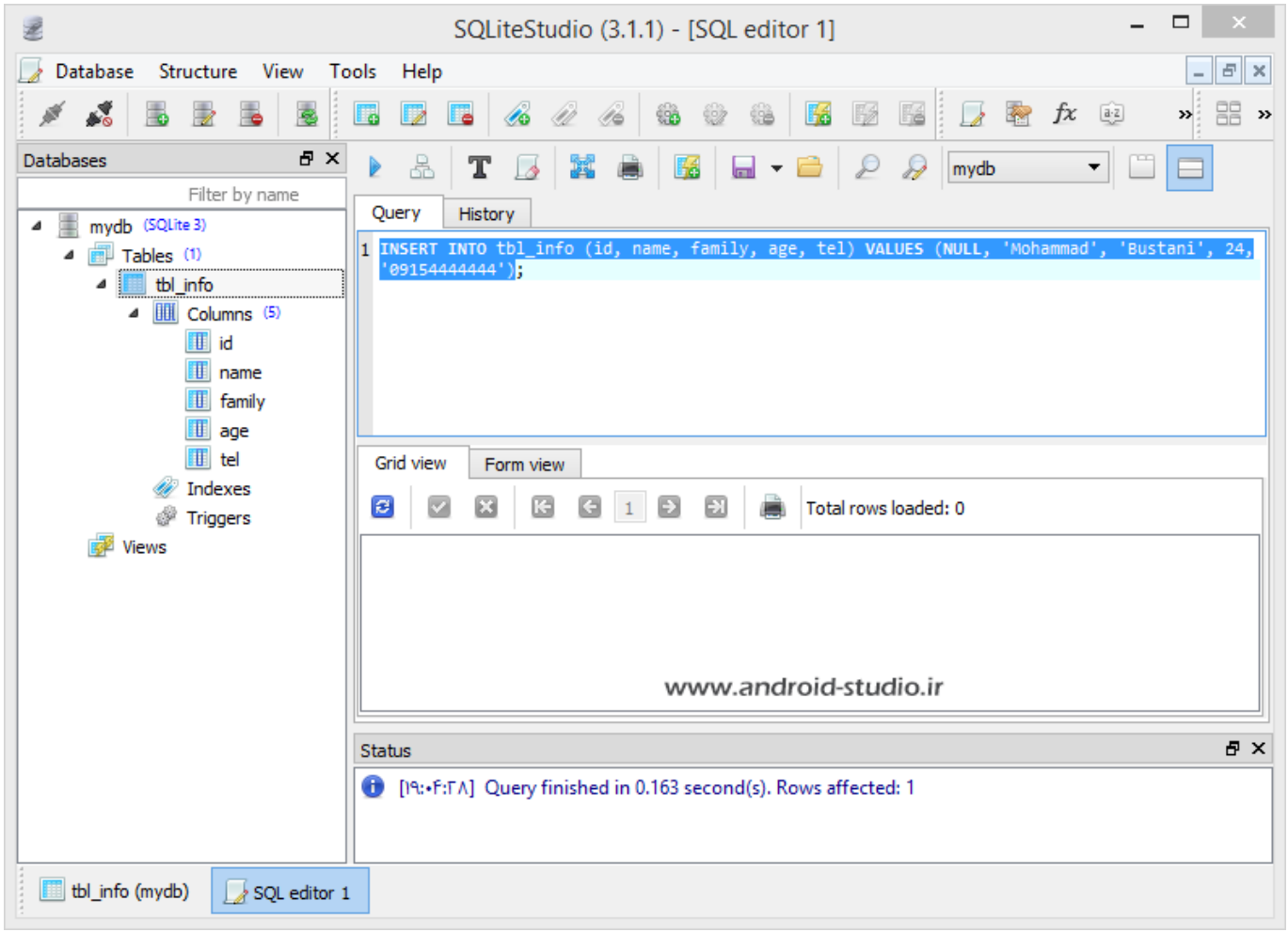


نکته : با نگه داشتن موس روی هر گزینه، نام و کلید میانبر آن نمایش داده می شود.

کوئری زیر را وارد و توسط دکمه Execute query اجرا می کنم (execute به معنی اجرا می باشد):

```
INSERT INTO tbl_info (id, name, family, age, tel) VALUES (NULL, 'Mohammad', 'Bustani', 24, '09154444444');
```





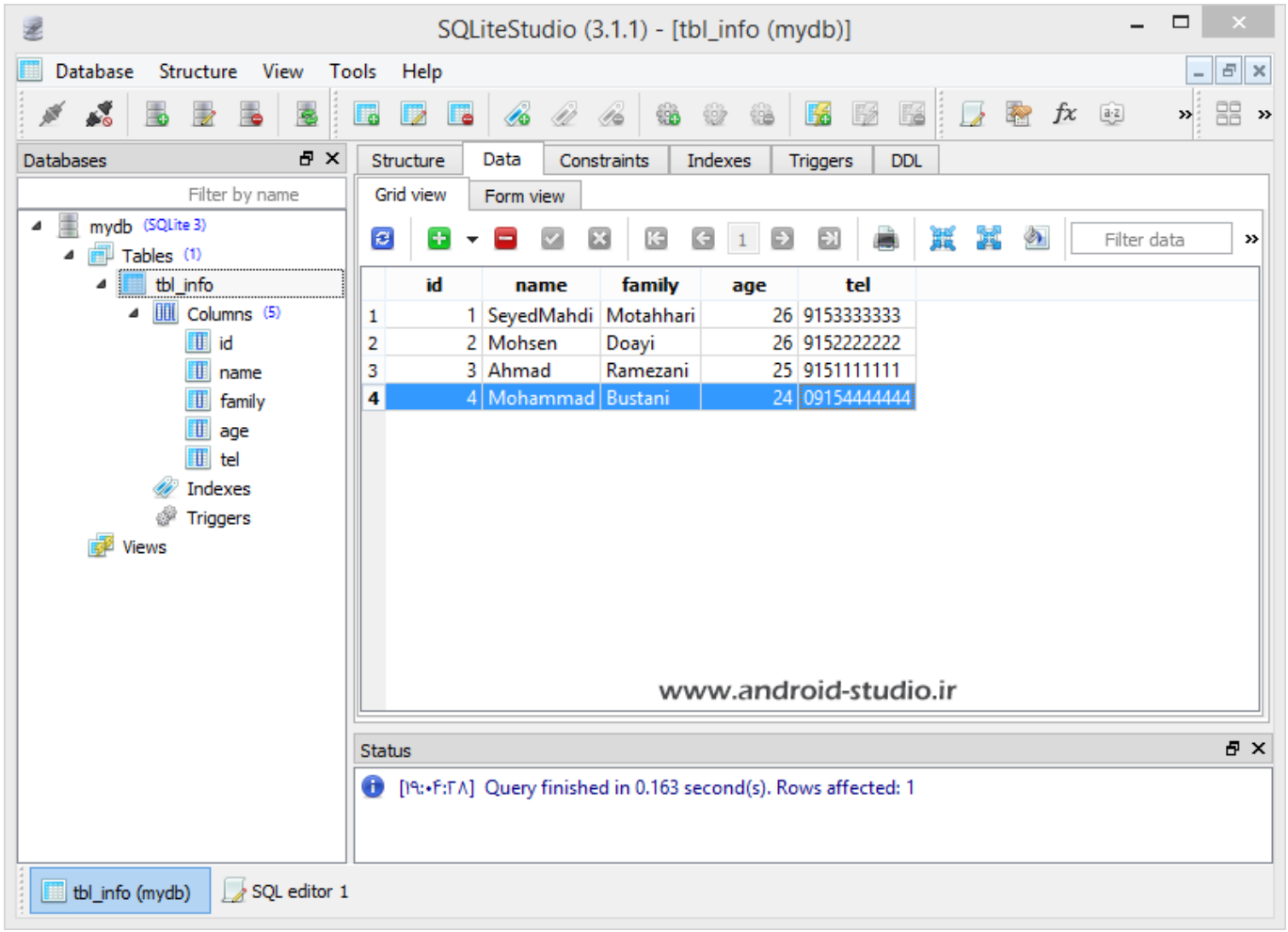
کوئری با موفقیت اجرا شد و پیغام Rows affected:1 نشان می دهد یک سطر مورد تغییر قرار گرفته که در اینجا تغییر شامل اضافه شدن بود.

نکته : در کوئری id به صورت NULL وارد شد تا شناسه به صورت خودکار به این ردیف اختصاص یابد.

نکته : باز هم تاکید می کنم تعداد فاصله ها، کوچک یا بزرگ بودن حروف و یک خطی/چند خطی بودن کوئری تاثیری در نتیجه ندارد. تنها اینکه لازم است بین کلمات بدنه کوئری مانند INSERT و INTO یا VALUES و () حداقل یک فاصله ایجاد شود.

نکته : مقادیر رشته ای مانند TEXT و VARCHAR می بایست داخل ' ' (تک کوتیشن) قرار گیرد.

مجدد به دیتابیس برمیگردم (گزینه های پایین/سمت چپ نرم افزار امکان سوئیچ کردن بین SQL editor و دیتابیس را فراهم کرده است). در برگه Data با بروزرسانی اطلاعات جدول توسط گزینه Refresh table data، ردیفی که به واسطه کوئری اضافه کرده بودم مشاهده می شود:



مشاهده می کنید عدد صفر ابتدای تلفن همراه به درستی نمایش داده شده.

کوئری بعدی که آزمایش می کنم Update هست.

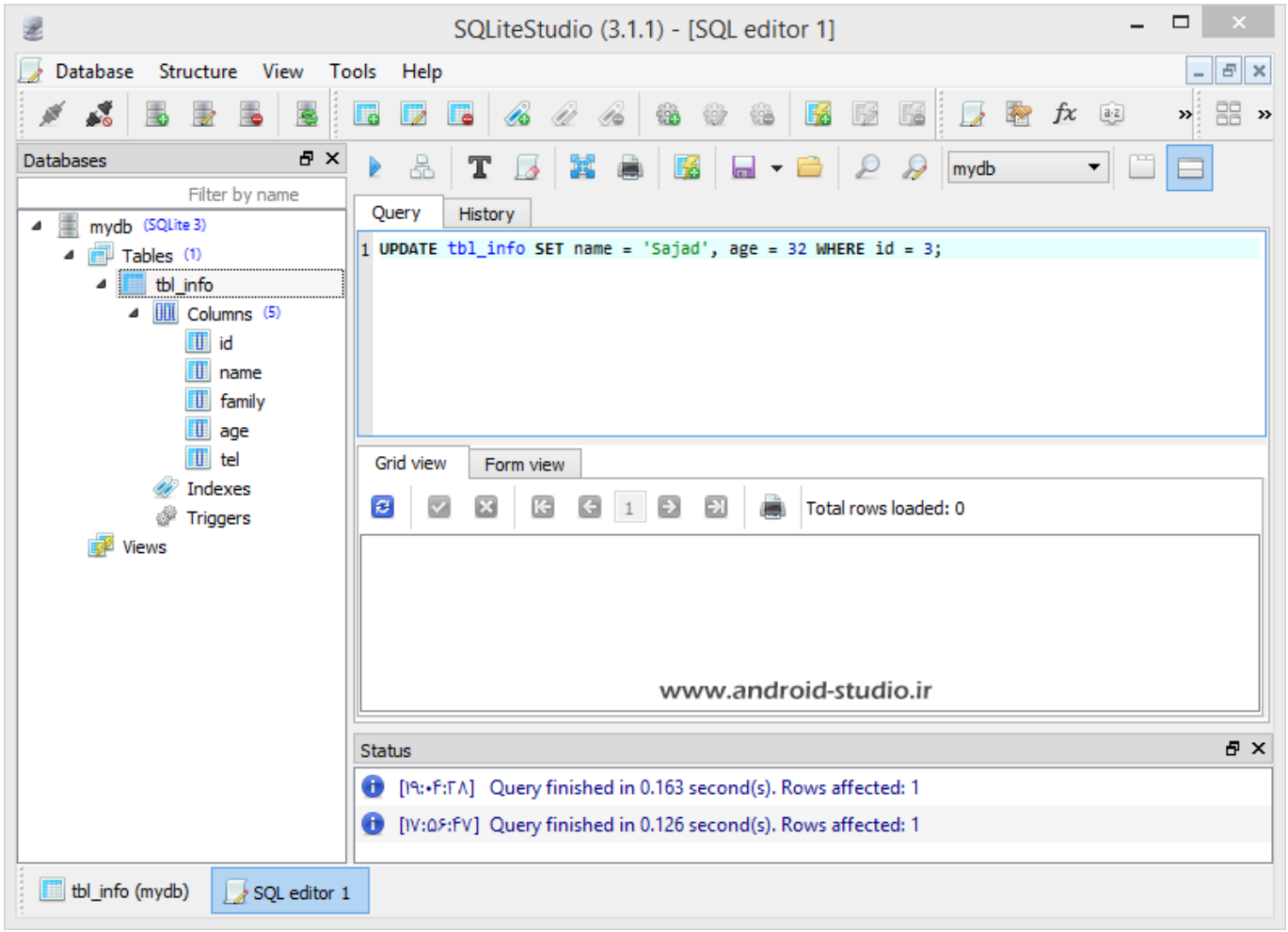
Syntax کوئری Update:

```
UPDATE table_name SET column1 = value1, column2 = value2...., columnN = valueN WHERE [condition];
```

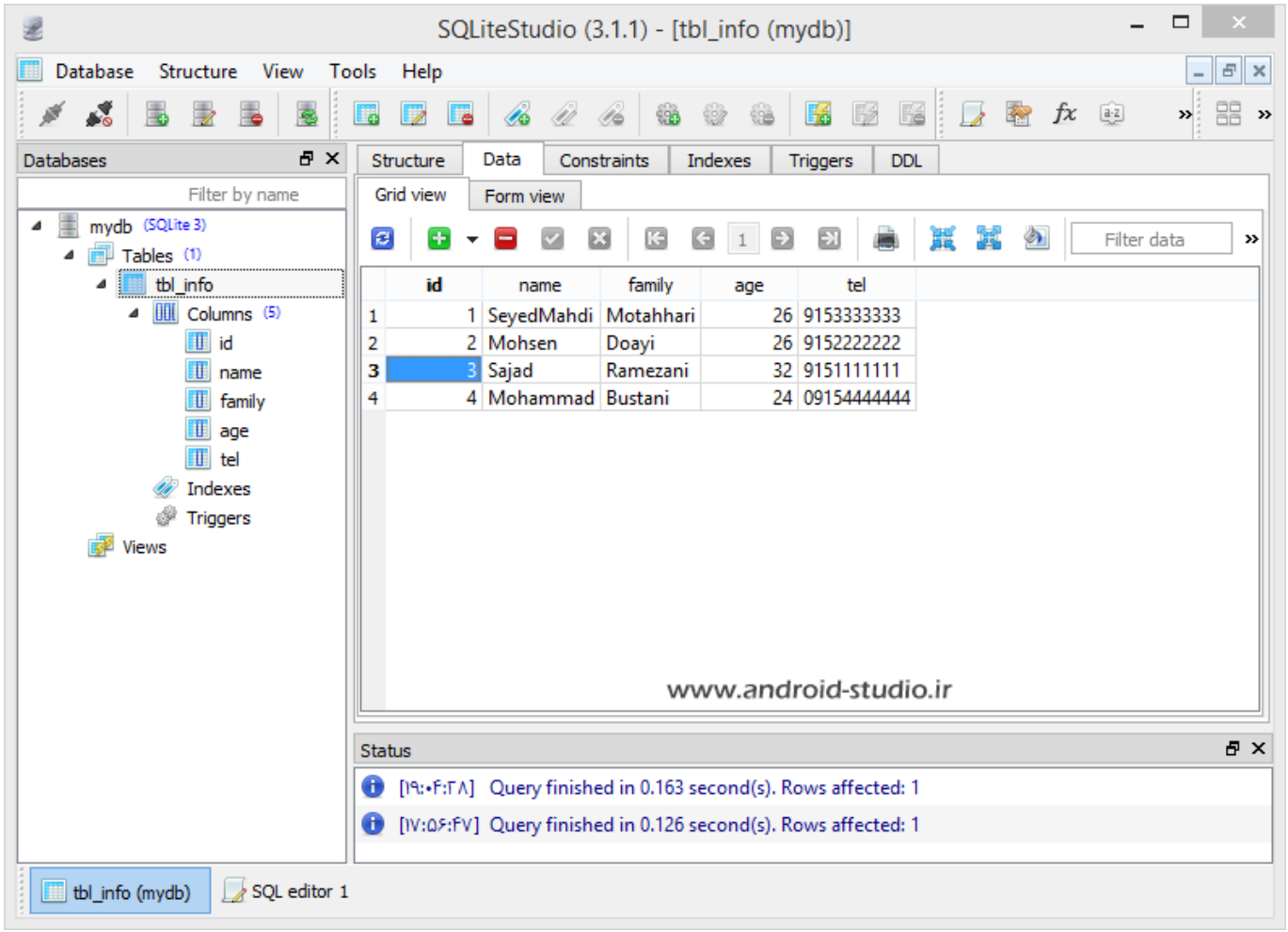
می خواهم نام شناسه ۳ را از Ahmad به Sajad و سن را هم از ۲۵ به ۳۲ تغییر دهم. مجدد از طریق گزینه SQL Editor 1 به بخش کوئری رفته و دستور زیر را وارد می کنم:

```
UPDATE tbl_info SET name = 'Sajad', age = 32 WHERE id = 3;
```

پس از اجرا پیغام تایید دریافت کردم که با بروزرسانی جدول، تغییرات مشاهده می شود:



دستورات کاملا واضح هستند. WHERE id = 3 یعنی ردیف/ردیفهایی که دارای این شرط (یعنی شناسه ۳) هستند.



کوئری Select هم برای انتخاب و نمایش ردیف هایی با مشخصاتی خاص بکار می رود و بر خلاف دو کوئری قبل، تغییری در جدول ایجاد نمی کند. به عنوان مثال می خواهم فقط لیست افرادی نمایش داده شود که سن آنها ۲۶ سال است.

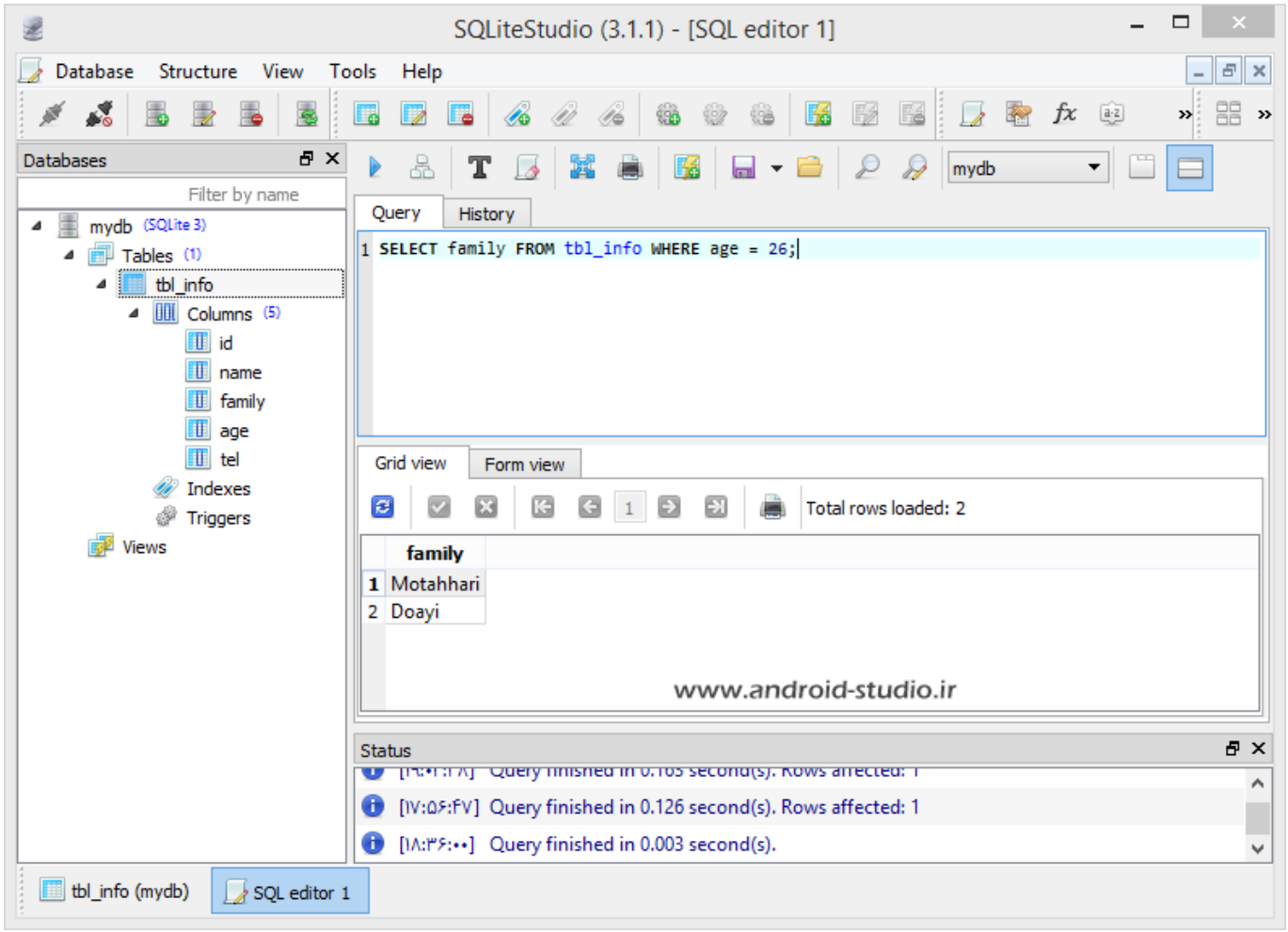
Syntax کوئری Select به اینصورت است:

```
SELECT column1, column2, columnN FROM table_name WHERE [condition];
```

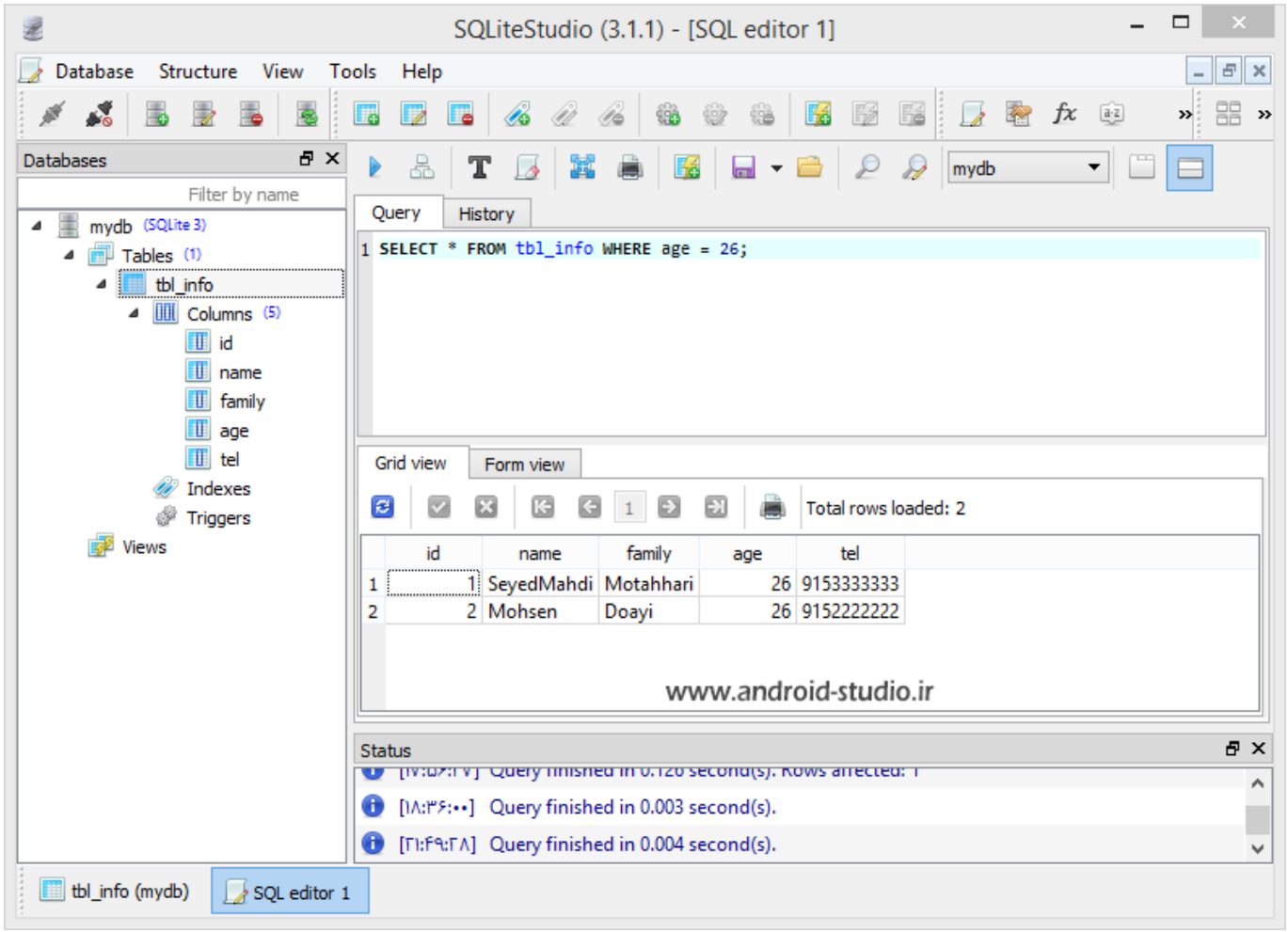
کوئری مدنظر من:

```
SELECT family FROM tbl_info WHERE age = 26;
```

همانطور که در پاراگراف بالا اشاره شد، دستور Select صرفا برای انتخاب اطلاعات است که این اطلاعات در همان برگه Query editor نمایش داده می شود:



من ستون family از جدول tbl_info با شرط مقدار ۲۶ برای ستون age را در کوئری قرار دادم که در قسمت Grid view دو نام خانوادگی به عنوان خروجی نمایش داده شده است. ممکن است در مواقعی لازم باشد تمامی ستون ها (در اینجا اطلاعات کامل شخص) انتخاب شود که کافیست به جای تعیین ستون یا ستون های خاص (مانند family در مثال بالا) کاراکتر * جایگزین شود. در سینتکس SQL علامت ستاره به معنی "همه" است و نیاز نیست تک تک ستون ها به صورت دستی قید شود:



مشاهده می کنید در خروجی، تمامی ستون های مشخصات افراد با سن ۲۶ سال نمایش داده شده است.

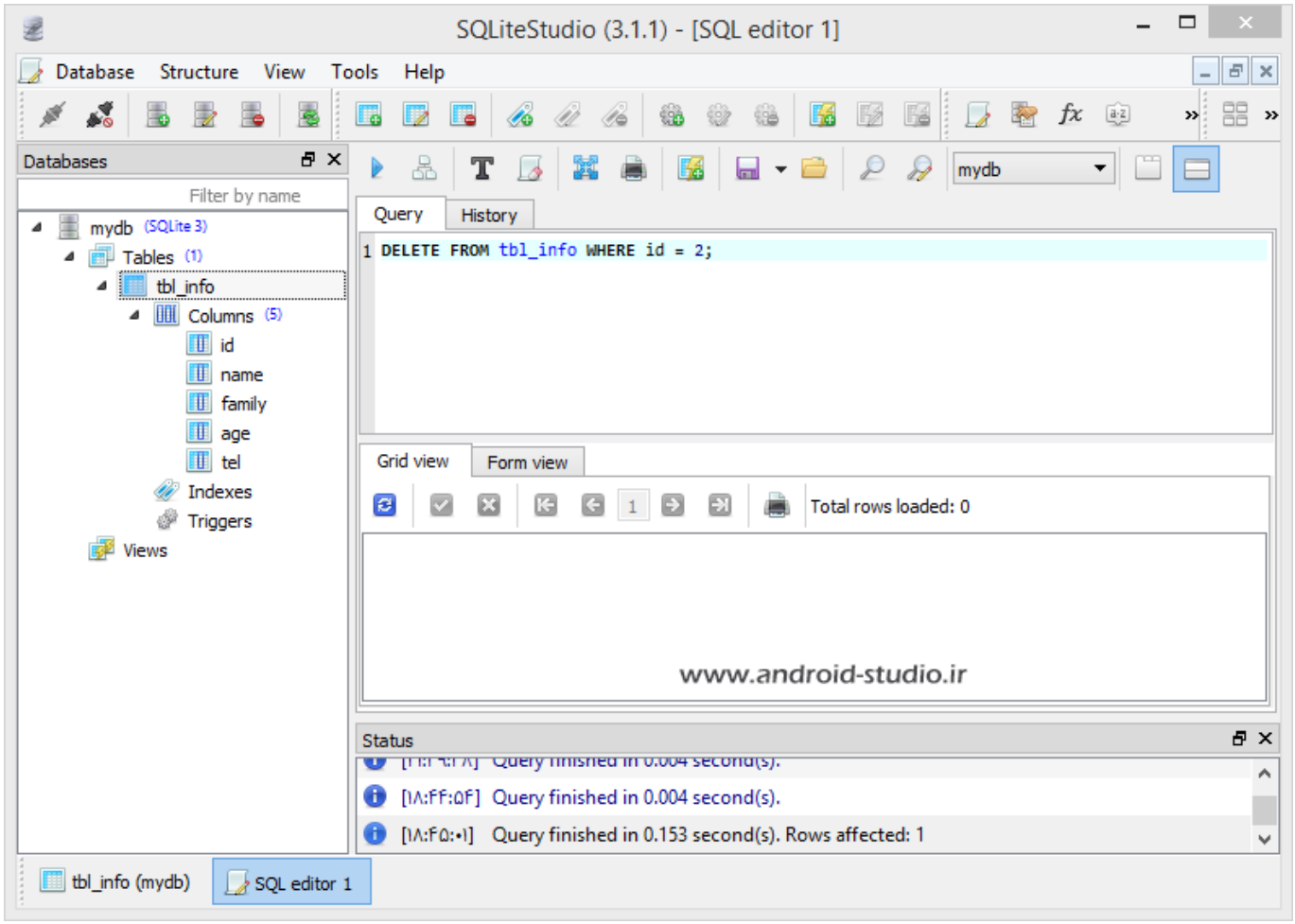
و اما آخرین کوئری که در این قسمت بررسی می کنیم دستور DELETE است که مشخصا برای حذف داده ها بکار می رود.

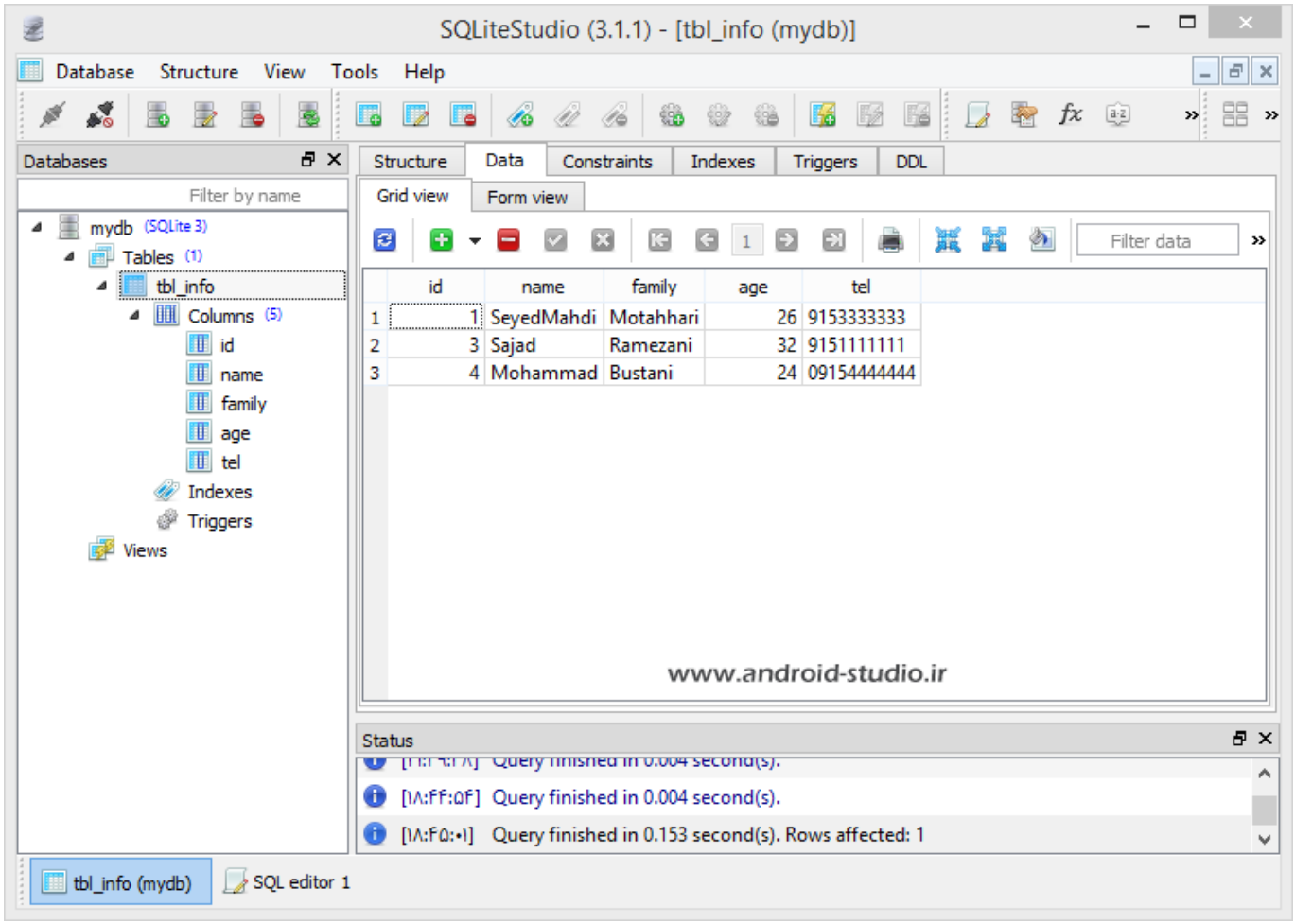
سینتکس کوئری Delete:

```
DELETE FROM table_name WHERE [condition];
```

به عنوان مثال می خواهم اطلاعات شخصی که با شناسه شماره ۲ در دیتابیس ثبت شده را حذف کنم:

```
DELETE FROM tbl_info WHERE id = 2;
```


SQLiteStudio (3.1.1) - [SQL editor 1]
Database
Structure
View
Tools
Help
Databases
Filter by name
mydb (SQLite 3)
Tables (1)
tbl_info
Columns (5)
id
name
family
age
tel
Indexes
Triggers
Views
mydb
Query
History
1 DELETE FROM tbl_info WHERE id = 2;
Grid view
Form view
Total rows loaded: 0
www.android-studio.ir
Status
[۱۸:۴۴:۵۴] Query finished in 0.004 second(s).
[۱۸:۴۵:۰۱] Query finished in 0.153 second(s). Rows affected: 1
tbl_info (mydb)
SQL editor 1

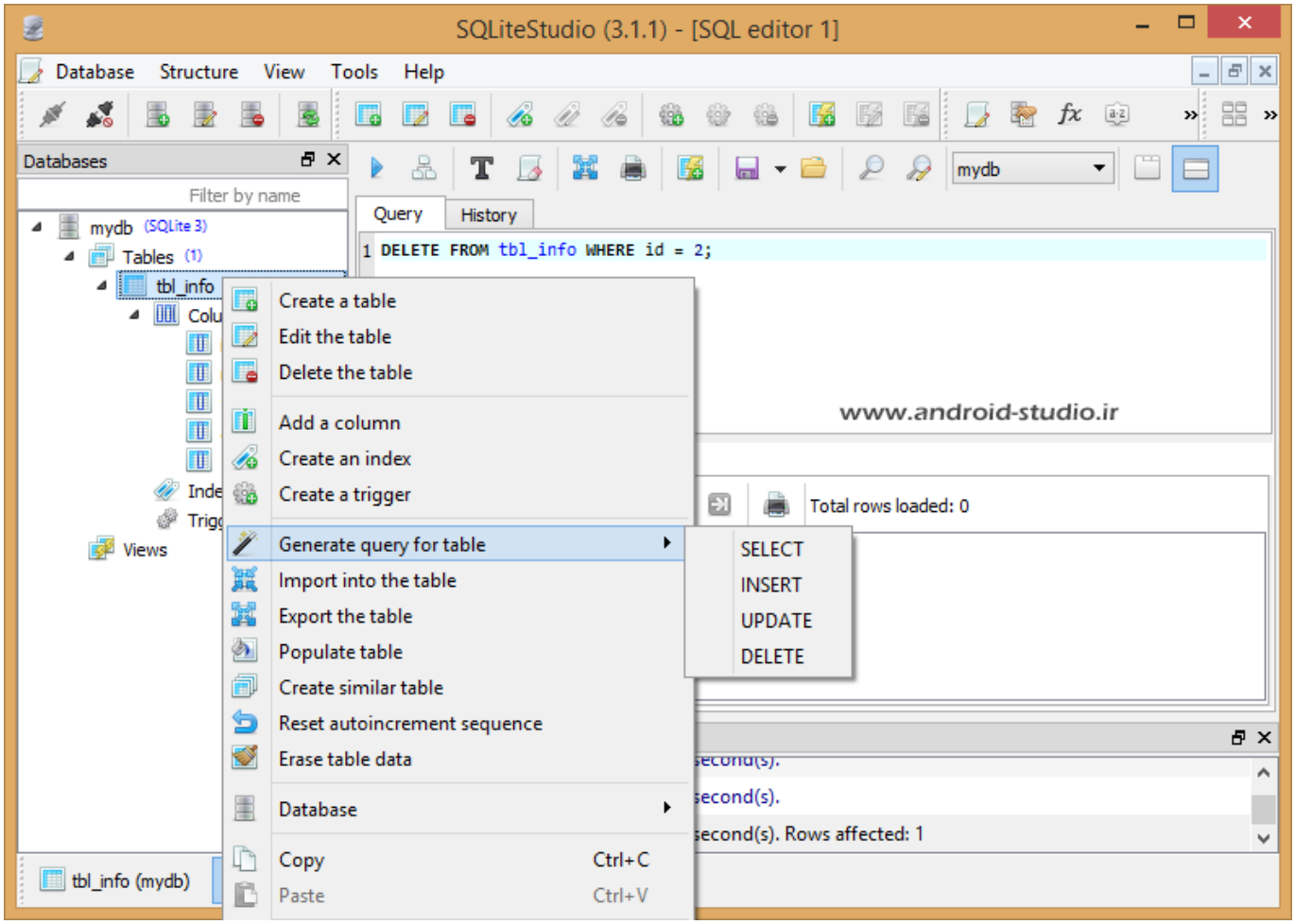


ردیف با شناسه ۲ از جدول حذف شد.

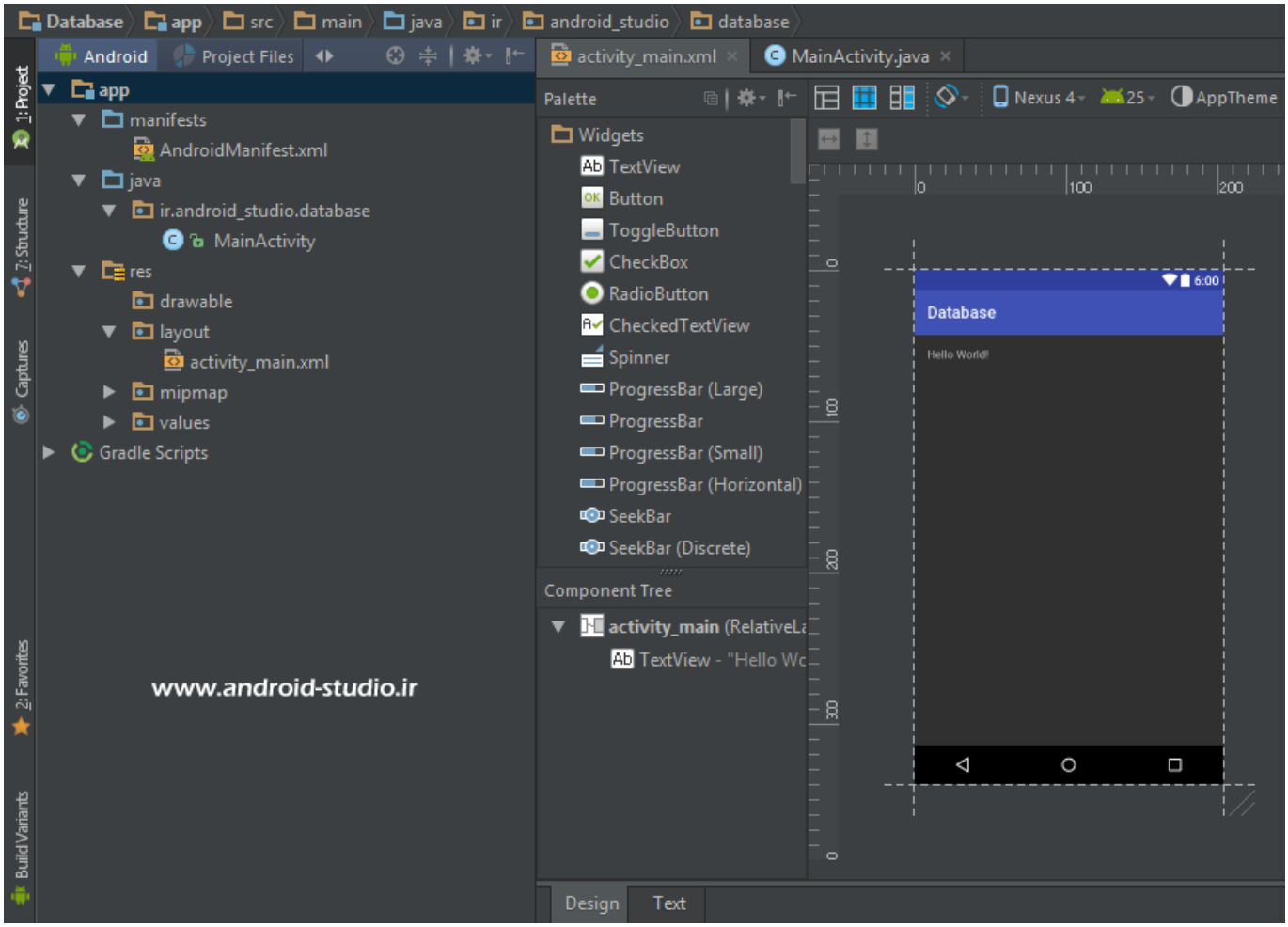
فعلا در همین حد آشنایی با مبحث دیتابیس برای ما کفایت می کند. با این حال اگر مایل بودید اطلاعات بیشتری در این خصوص کسب کنید لینک های مفیدی در وب پیدا خواهید کرد. از جمله:

https://www.tutorialspoint.com/sqlite/index.htm

قبل از خروج از SQLiteStudio هم بهتر است چند دقیقه ای وقت گذاشته و با سایر گزینه ها و امکانات این نرم افزار و یا ابزار مشابهی که استفاده می کنید آشنا شوید. برای مثال با راست کلیک روی نام جدول و گزینه generate query for table به صورت خودکار چهار عمل اصلی که در بالا معرفی شد در قسمت Query editor برای شما آماده استفاده می باشد.



خب! حالا سراغ اندروید استودیو و ساخت پروژه جدید اندرویدی می رویم. من یک پروژه جدید با نام Database و با API Level 10 و یک اکتیویتی از نوع Empty Activity ایجاد کردم که در زمان تهیه این آموزش، آخرین API یی هست که بنا به اطلاعات گوگل، از ۱۰۰% دیوایس ها پشتیبانی می کند.



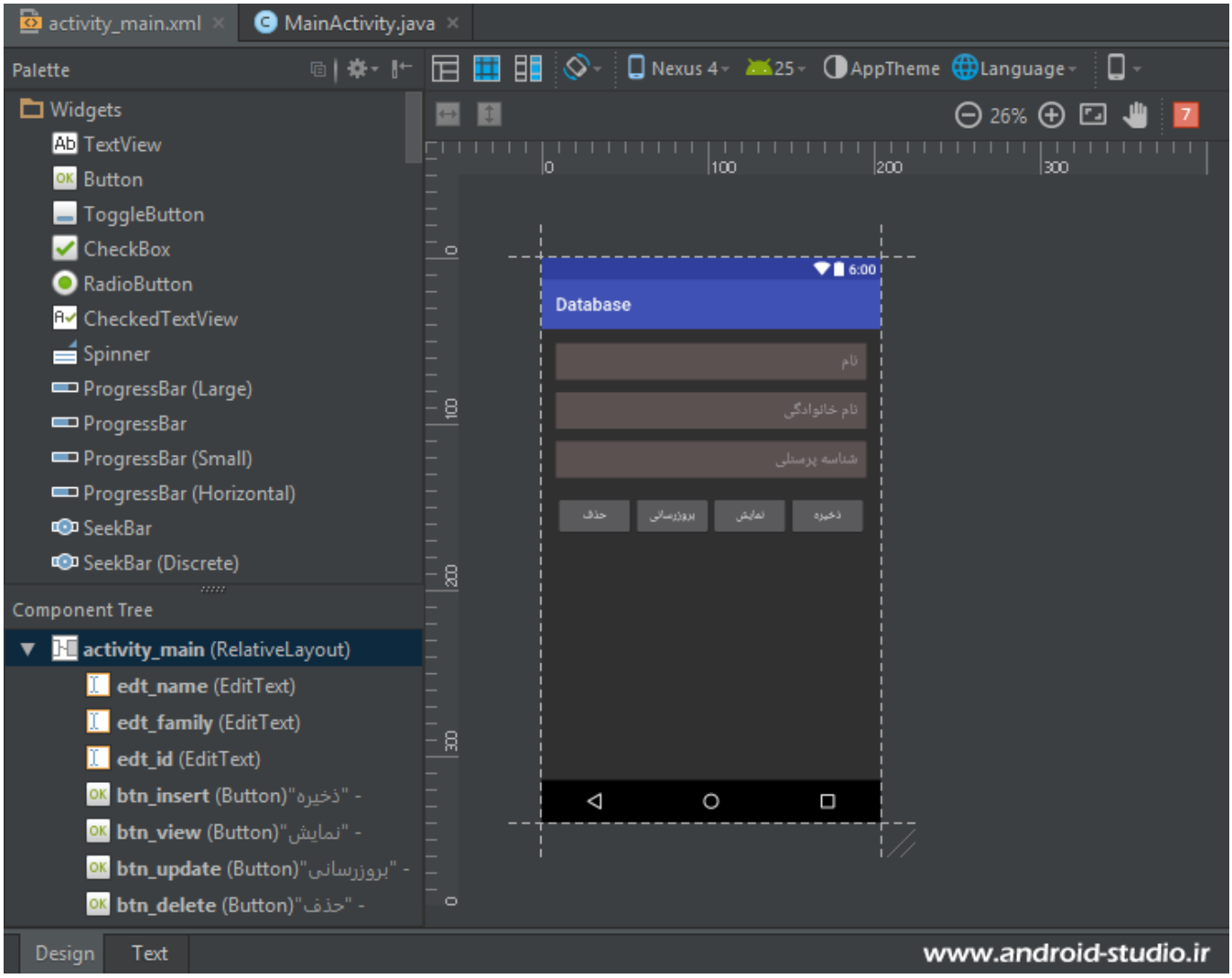
همانطور که در تصویر مشاهده می کنید پوشه های AndroidTest و Test درون پوشه Java را حذف کرده ام. در این پروژه من نیازی به این پوشه ها ندارم (اگر با یک بار انتخاب پوشه ها و delete کردن، کاملا حذف نشد مجدد پوشه ها را انتخاب و delete کنید).

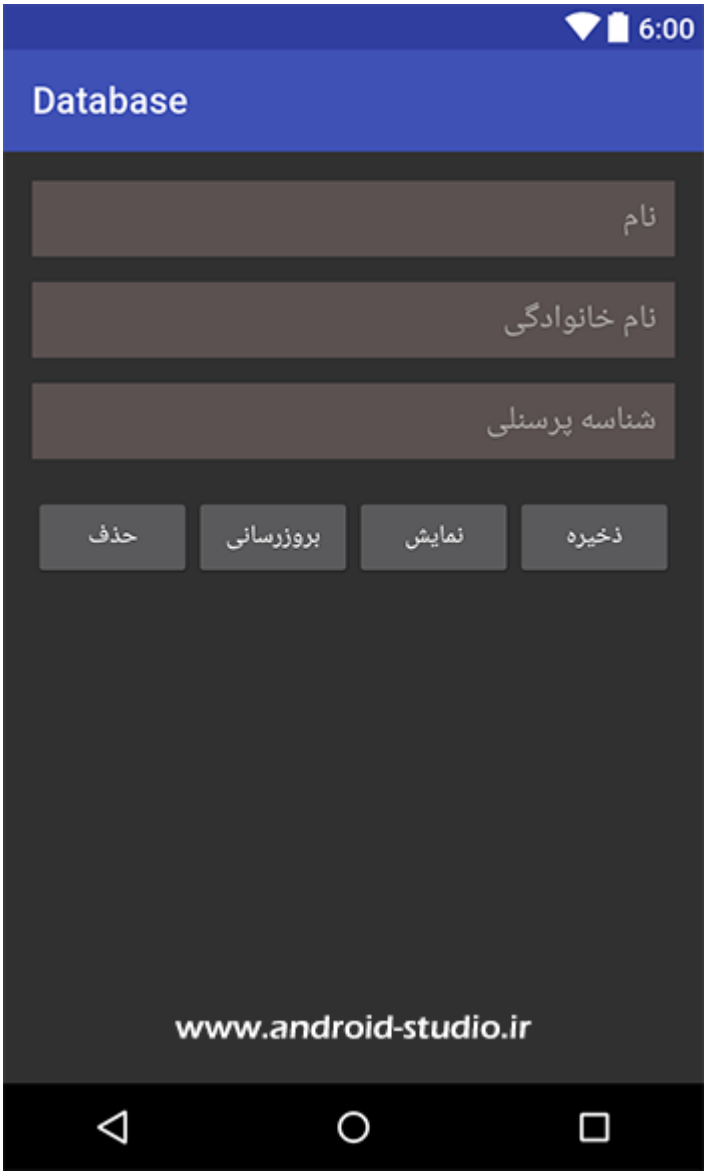
پروژه را در قالب یک اپلیکیشن کاربردی پیش می بریم. یک واحد تولیدی را در نظر می گیریم که تمامی پرسنل آن، یک شماره پرسنلی اختصاصی بر روی لباسشان درج شده و مدیر واحد قصد دارد روی موبایل اندرویدی خود، اپلیکیشنی داشته باشد که بتواند اطلاعات کامل کارکنان را ثبت و در مواقع لازم با وارد کردن کد پرسنلی درج شده بر روی لباس پرسنل، به مشخصات وی دسترسی داشته باشد. البته مدنظرتان باشد پروژه ای که در این آموزش می سازیم با اپلیکیشنی که در دنیای واقعی تحویل مشتری داده می شود تفاوت بسیاری دارد و من از مواردی مثل طراحی استاندارد ظاهر برنامه و رعایت حداکثری استانداردها در کد نویسی (مانند فرخوانی رشته های متنی از strings.xml و...) به جهت خلاصه شدن مبحث خودداری می کنم.

در قدم اول اجزای رابط کاربری را در اکتیویتی MainActivity قرار می دهم که به راحتی ویجت ها را از Palette به روی اکتیویتی کشیده و رها می کنم. ویجت های مدنظر من شامل سه EditText و چهار

Button هستند که از EditText ها برای دریافت و یا نمایش و Button ها برای ثبت، نمایش، بروزرسانی و حذف اطلاعات استفاده خواهد شد. البته که بهتر بود برای نمایش اطلاعات به صورت جداگانه از TextView استفاده می شد اما هدف ما در این مبحث صرفا آشنایی با دیتابیس است و هرچه خلاصه تر باشد درک آن نیز ساده تر خواهد بود. در حالت Design هم می توان ویجت ها را به تعداد مورد نیاز کپی کرد. کافیست ویجت مدنظر را انتخاب کرده، Copy و Paste نمایید (و یا ساده تر، کلید های ترکیبی Ctrl+C و Ctrl+V). ممکن است در مواردی از جمله در RelativeLayout ها هنگام ایجاد کپی از یک ویجت، در قسمت پیش نمایش گرافیکی تغییر را مشاهده نکنید که لازم است ویجت جدید را از روی ویجت قبل جابجا کنید (لیست ویجت ها در قسمت Component tree قابل مشاهده است).

دو EditText نام و نام خانوادگی را از نوع معمولی و EditText مربوط به شماره پرسنلی را از نوع Number انتخاب کردم:




6:00
Database
نام
نام خانوادگی
شناسه پرسنلی
ذخیره
نمایش
بروزرسانی
حذف
www.android-studio.ir

سورس activity_main.xml :

```
<?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:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="ir.android_studio.database.MainActivity">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:ems="10"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:padding="10dp"
        android:id="@+id/edt_name"
        android:background="#5b5151"
        android:hint="نام" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:ems="10"
        android:layout_below="@+id/edt_name"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="14dp"
        android:padding="10dp"
        android:id="@+id/edt_family"
        android:background="#5b5151"
        android:hint="نام خانوادگی" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:ems="10"
        android:layout_below="@+id/edt_family"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="14dp"
        android:padding="10dp"
        android:id="@+id/edt_id"
        android:background="#5b5151"
        android:hint="شناسه پرسنلی" />

    <Button
        android:text="ذخیره"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edt_id"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_marginTop="19dp"
        android:id="@+id/btn_insert" />

    <Button
        android:text="نمایش"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/btn_insert"
        android:layout_toLeftOf="@+id/btn_insert"
        android:layout_toStartOf="@+id/btn_insert"
        android:id="@+id/btn_view" />

    <Button
        android:text="بروزرسانی"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/btn_view"
        android:layout_toLeftOf="@+id/btn_view"
        android:layout_toStartOf="@+id/btn_view"
        android:id="@+id/btn_update" />

    <Button
        android:text="حذف"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/btn_update"
        android:layout_toLeftOf="@+id/btn_update"
        android:layout_toStartOf="@+id/btn_update"
        android:id="@+id/btn_delete" />

</RelativeLayout>
```

من در مجموع سه ویجت از نوع EditText و با id های edt_name، edt_family، edt_id و چهار Button با id های btn_insert، btn_view، btn_update و btn_delete ایجاد کردم که کاربرد هرکدام از نام آن مشخص است. سعی کنید نامگذاری (در xml و java و...) را با الگوی مشخص و معنی دار انجام دهید تا در حین تکمیل پروژه کمترین اشتباه را داشته باشید.

در قدم بعد و در کلاس MainActivity.java به تعداد نیاز چند نمونه از کلاسهای EditText و Button می سازم:

```
package ir.android_studio.database;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

    EditText edtName, edtFamily, edtID;
    Button btnInsert, btnView, btnUpdate, btnDelete;

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

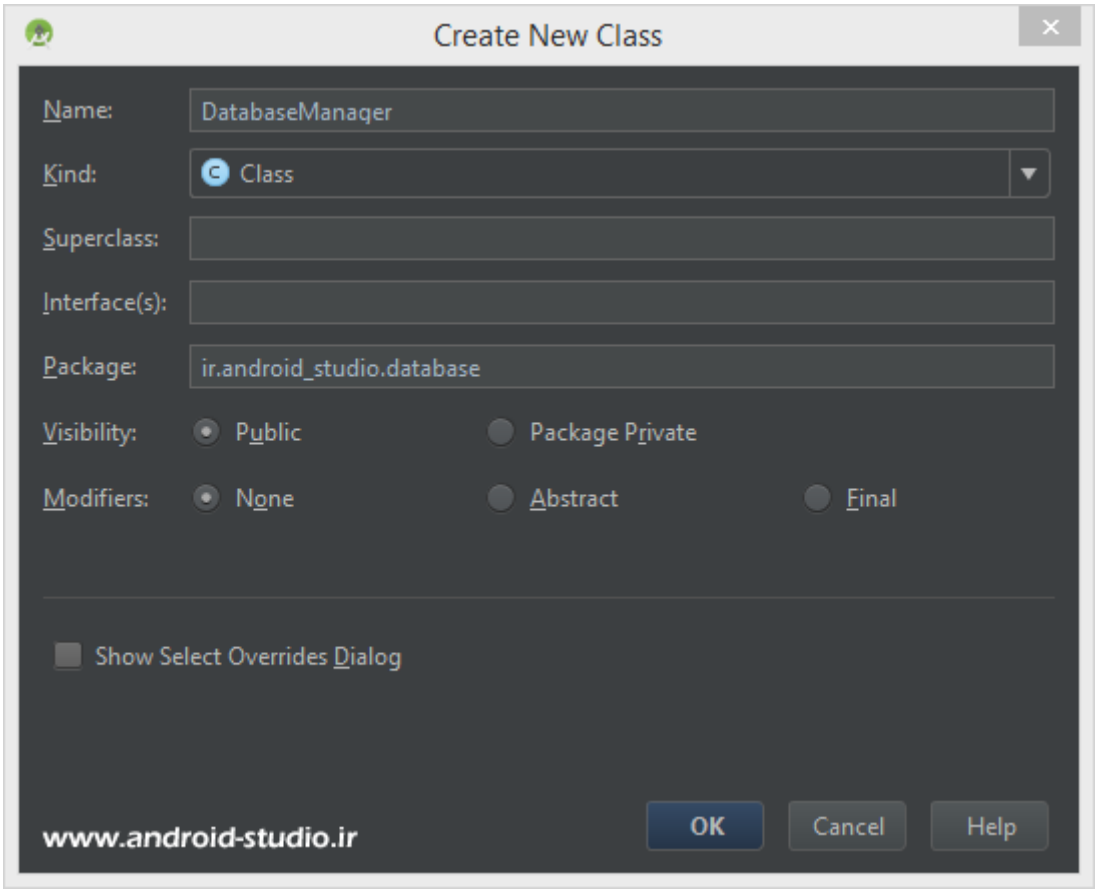
برای ساخت نمونه های هم نوع از یک کلاس (مانند EditText) به جهت کاهش حجم کد نهایی، آنها را

به صورت جداگانه تعریف نکردم و تنها با یک کاما پشت سر هم قرار دادم. حالا درون متد onCreate هر کدام از نمونه ها را به ویجت (کامپوننت) مربوطه متصل می کنم:

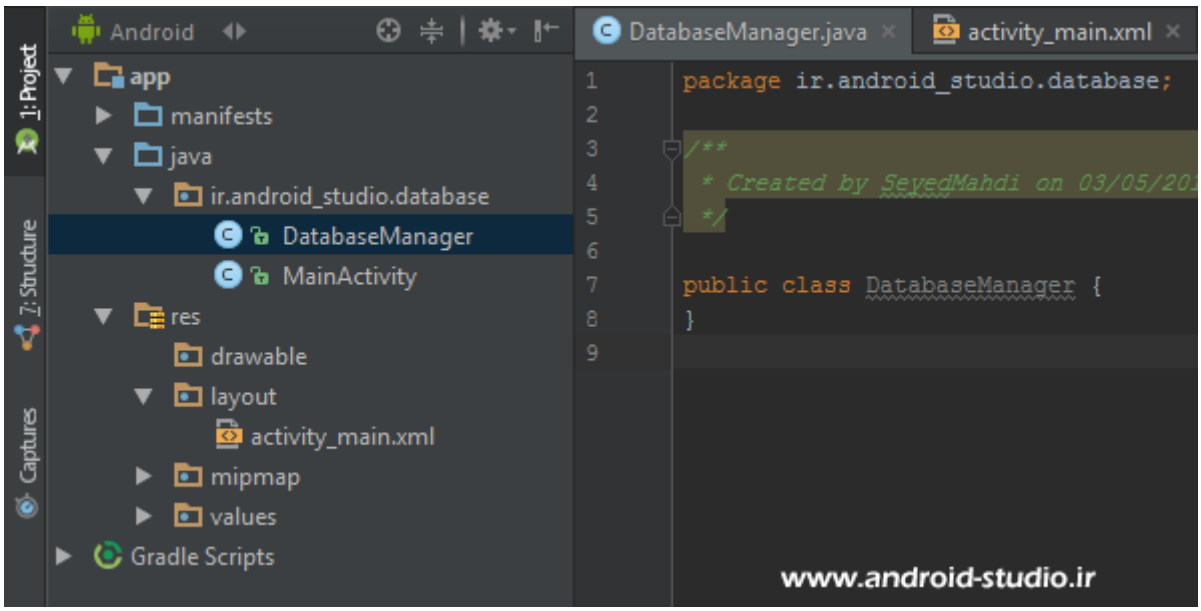
```
package ir.android_studio.database;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

    EditText edtName, edtFamily, edtID;
    Button btnInsert, btnView, btnUpdate, btnDelete;

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

        edtName = (EditText) findViewById(R.id.edt_name);
        edtFamily = (EditText) findViewById(R.id.edt_family);
        edtID = (EditText) findViewById(R.id.edt_id);
        btnInsert = (Button) findViewById(R.id.btn_insert);
        btnView = (Button) findViewById(R.id.btn_view);
        btnUpdate = (Button) findViewById(R.id.btn_update);
        btnDelete = (Button) findViewById(R.id.btn_delete);

    }
}
```

خب! فعلا کار ما با اکتیویتی تمام شد. حالا باید یک دیتابیس بسازیم، یک جدول درون دیتابیس ایجاد کنیم و متدهای مربوط به اعمال ذخیره، نمایش، بروزرسانی و حذف اطلاعات را بنویسیم.

اندروید یک کلاس کمکی با نام SQLiteOpenHelper تهیه کرده که برای مدیریت دیتابیس بکار می رود. من یک کلاس جاوا با نام **دلخواه** DatabaseManager به پروژه اضافه می کنم. به این منظور درون پوشه java و روی نام پکیج (ir.android_studio.database) راست کلیک کرده، گزینه New سپس Java Class و درون پنجره باز شده، نام کلاس را وارد می کنم:



Visibility کلاس می بایست public باشد که در حالت پیش فرض همین نوع انتخاب شده و در نهایت OK می کنم:

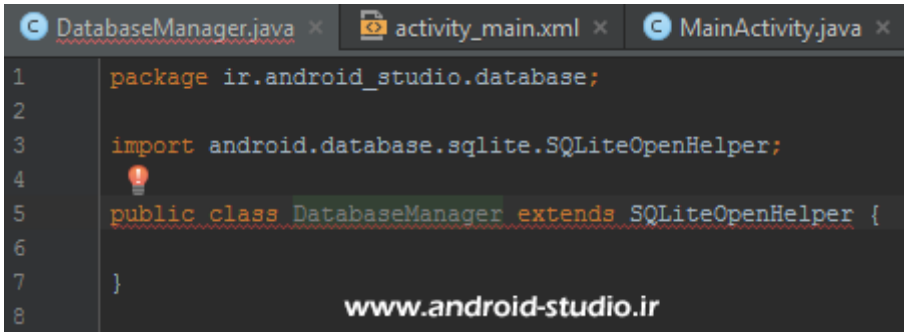


کلاس DatabaseManager.java به پروژه اضافه شد. کامنت بالای کلاس را حذف می کنم. توجه داشته باشید لزوما همه اخطارهایی که اندروید استودیو اعلام می کند به معنی خطا نیست. در تصویر

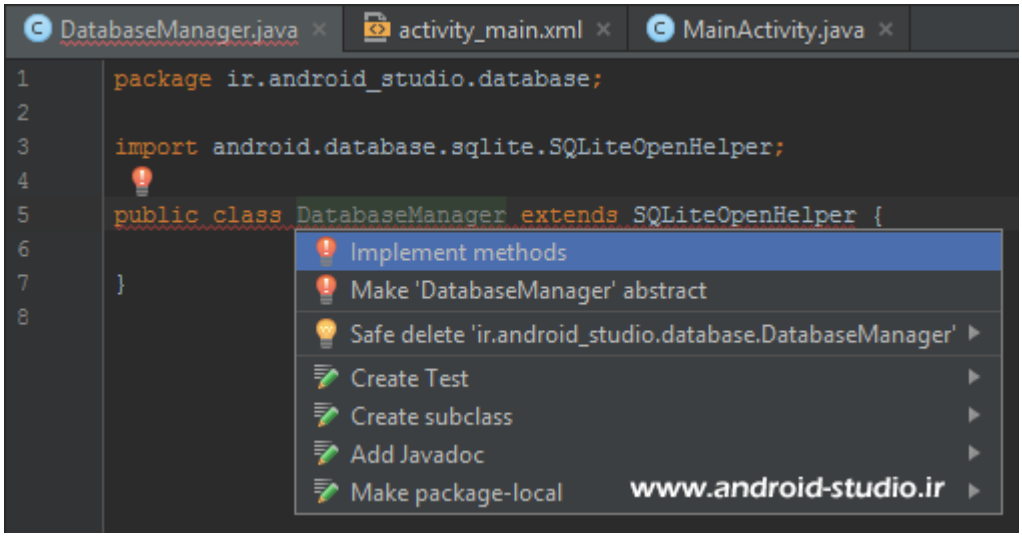
بالا زیر DatabaseManager خط کشیده شده که با نگه داشتن نشانگر روی آن، پیغام اخطار نمایش داده می شود که Class 'DatabaseManager' is never used هست، یعنی این کلاس هیچ کجای پروژه استفاده نشده که مشکلی نیست. ضمن اینکه خطاها با رنگ قرمز مشخص می شوند. حالا کافیست این کلاس از کلاس SQLiteOpenHelper ارث بری کند تا بتوانیم دیتابیس را مدیریت کنیم. یعنی کلاس DatabaseManager را extends می کنیم به SQLiteOpenHelper:

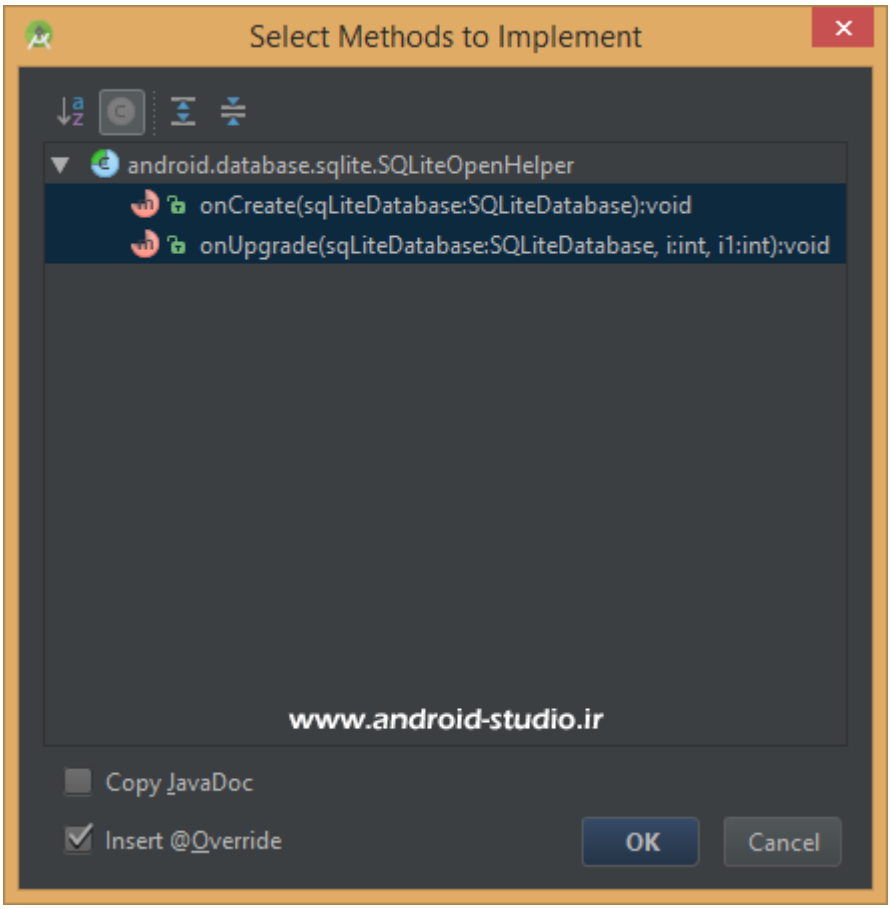
```
package ir.android_studio.database;

import android.database.sqlite.SQLiteOpenHelper;
public class DatabaseManager extends SQLiteOpenHelper {
}
```

در واقع DatabaseManager یک SubClass (زیر کلاس) از SQLiteOpenHelper است. مشاهده می کنید با انجام عمل ارث بری، android.database.sqlite.SQLiteOpenHelper به صورت خودکار import شد (اگر ایمپورت انجام نشد پیغام مربوطه را مشاهده می کنیم که با زدن Alt+Enter ایمپورت صورت می گیرد). حالا یک ارور داریم:



با بردن نشانگر روی بدنه کلاس متن ارور نمایش داده می شود با این مضمون که متد ()onCreate باید به این کلاس اضافه (یا اصطلاحا implement) شود. با کلیک روی آیکون لامپ قرمز یا زدن Alt+Enter متدهای لازم را می توان به صورت خودکار اضافه کرد و نیاز به نوشتن دستی آنها نیست:





در این پنجره دو متد ()onCreate و onUpgrade وجود دارد که به صورت پیش فرض هردو انتخاب شده هستند و من هر دو را تایید می کنم:

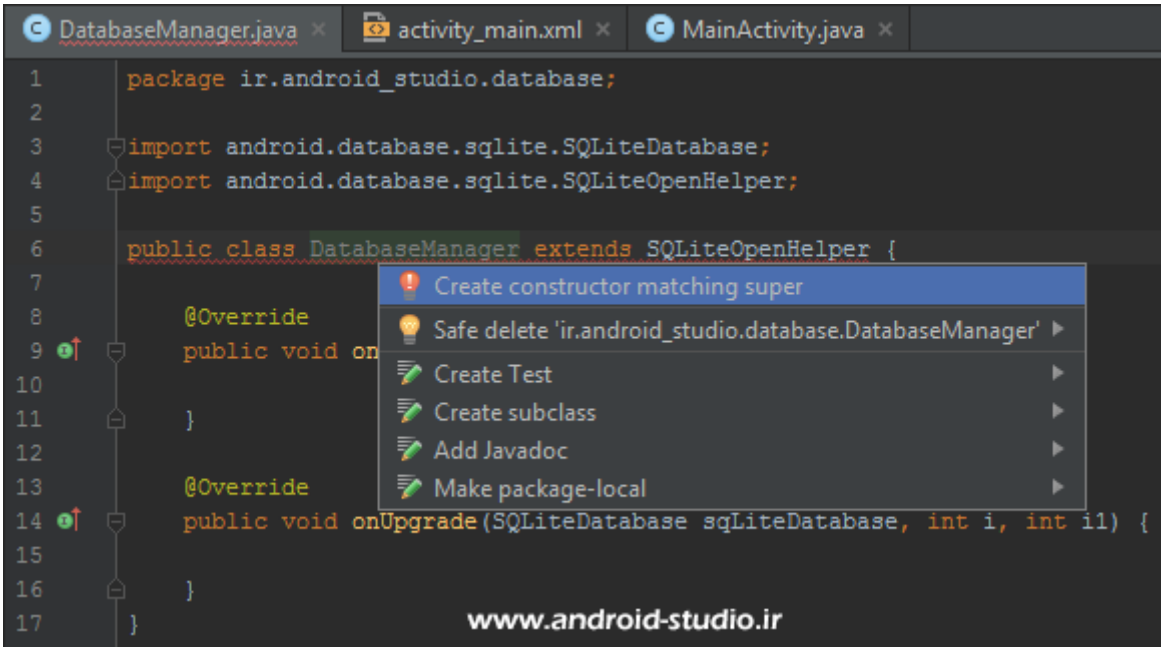
```
package ir.android_studio.database;

import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseManager extends SQLiteOpenHelper {

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {

    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {

    }
}
```

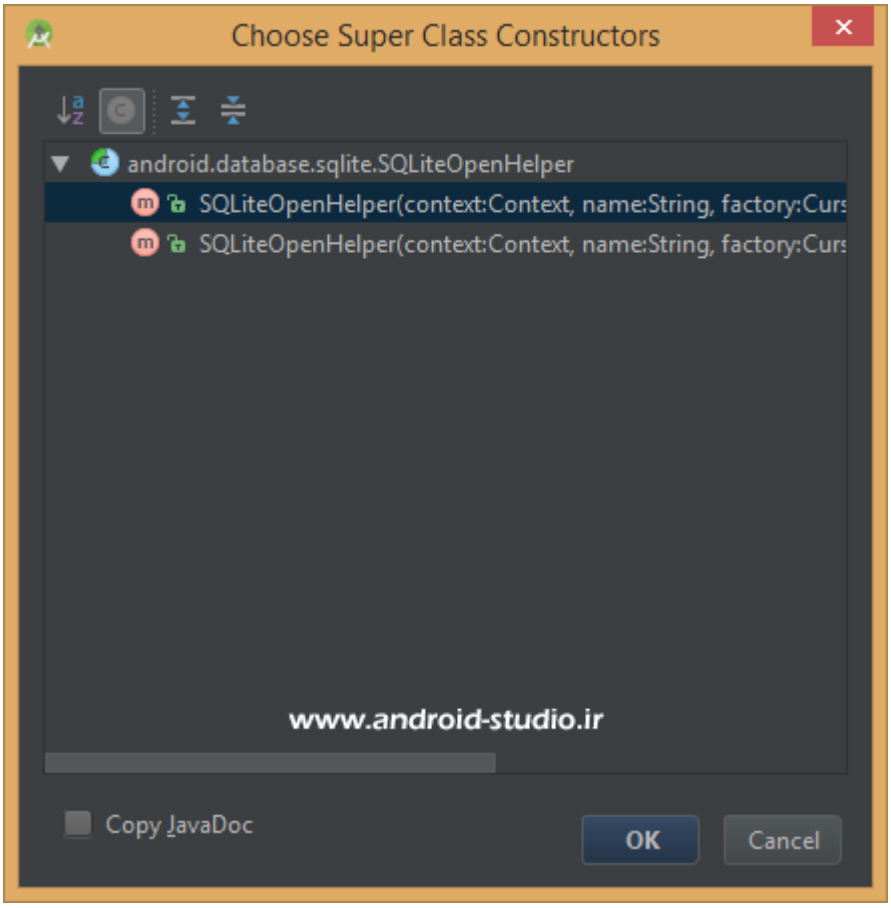
دو متد با پارامتر ورودی SQLiteDatabase و نام sqLiteDatabase داخل کلاس ایمپلمنت شد. متد onCreate() برای ساخت جدول دیتابیس و متد onUpgrade() برای بروزرسانی ساختار جدول (حذف یا اضافه کردن ستون) بکار می روند.

اما هنوز اندروید استودیو برای این کلاس یک ارور دیگر نمایش می دهد:

There is no default constructor available in 'android.database.sqlite.SQLiteOpenHelper'

یعنی کلاسی که از SQLiteOpenHelper مشتق (ارث بری) شده، علاوه بر دو متد فوق، به یک سازنده (Constructor) نیز نیاز دارد که عمل ایجاد فایل دیتابیس توسط سازنده صورت می گیرد:





دو گزینه برای انتخاب داریم که اندروید استودیو گزینه اول را انتخاب کرده و من هم همین سازنده را تایید می کنم. سازنده دوم یک error Handler اضافه دارد که مورد نیاز ما نیست.

محتوای فعلی کلاس:

```
package ir.android_studio.database;

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

public class DatabaseManager extends SQLiteOpenHelper {

    public DatabaseManager(Context context, String name, SQLiteDatabase.CursorFactory factory,
int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {

    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {

    }
}
```

متد سازنده همیشه همنام با کلاس خودش است. من سعی می کنم نامگذاری های پیش فرض را تغییر دهم و همچنین در هر متد از نامهای متفاوتی استفاده کنم تا کمتر باعث سردرگمی شما شود. اما مسلما زمانی که کدها را درک کردید، نیازی نیست در نامگذاری ها وسواس بخرج دهید. به عنوان مثال نام پارامتر Context را از context به cnt تغییر می دهم.

در متد سازنده بجز پارامتر Context مابقی را حذف می کنم زیرا نیازی به آنها نداریم:

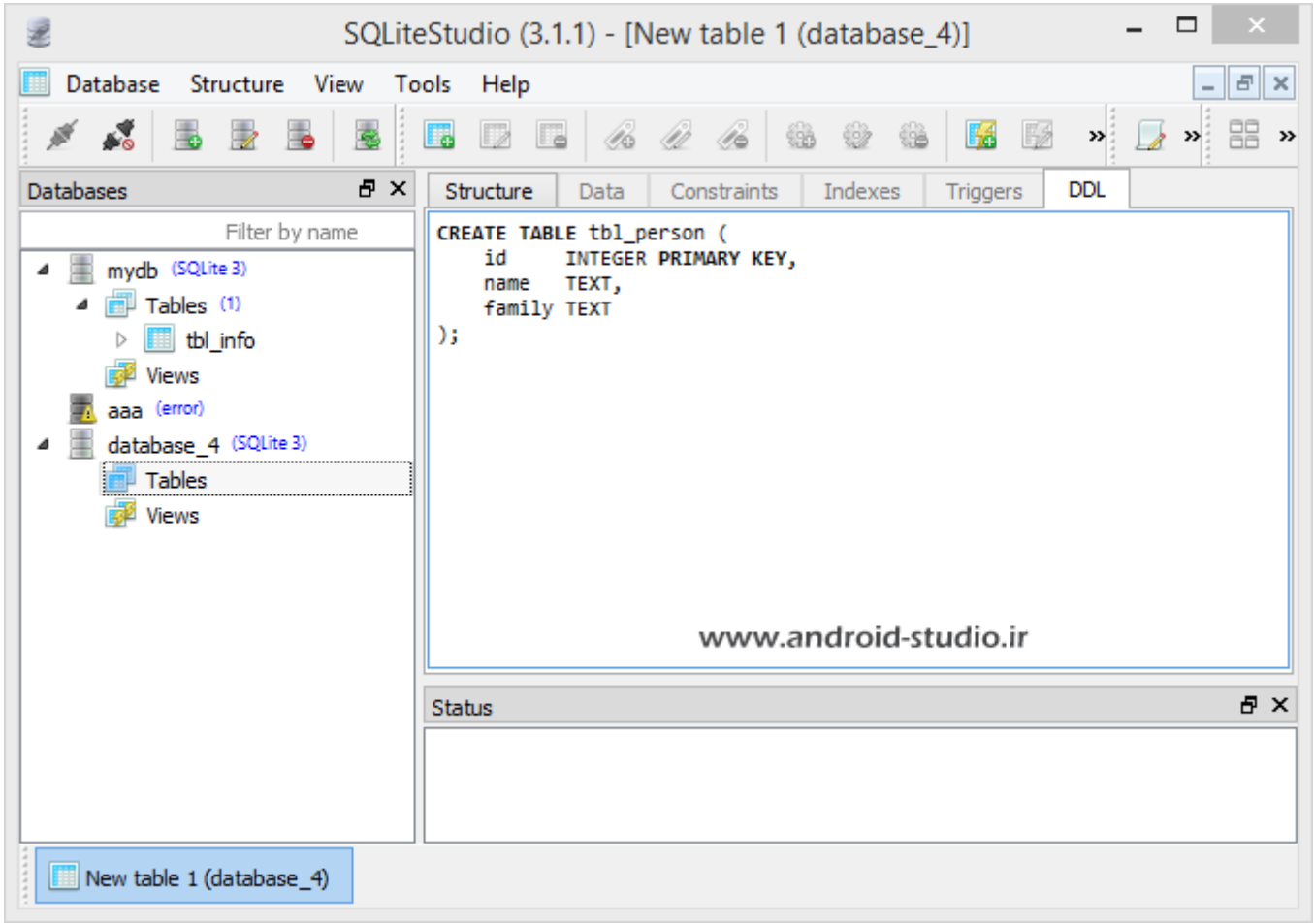
```
public DatabaseManager(Context cnt) {
    super(cnt, name, factory, version);
}
```

داخل ()super چهار پارامتر داریم. اولی همان نام نمونه ای است که از Context ساخته ایم یعنی cnt. دومی نام دیتابیس است. مورد سوم یعنی factory را null قرار می دهیم. مورد آخر ورژن دیتابیس را تعیین می کند. به صورت زیر تکمیل می کنم:

```
package ir.android_studio.database;

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

public class DatabaseManager extends SQLiteOpenHelper {

    private static final String DatabaseName = "myinfo.db";
    private static final int Version = 1;

    public DatabaseManager(Context cnt) {
        super(cnt, DatabaseName, null, Version);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {

    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {

    }
}
```

متغییرهای DatabaseName و Version را private static final تعریف کردم تا خصوصی بوده (در همین کلاس در دسترس باشد) و همچنین نهایی باشد (یعنی قابل ارث بری نباشد). نام دیتابیس می بایست دلخواه و با پسوند db. تعیین گردد که من myinfo.db را انتخاب کردم. ورژن هم به صورت عددی تعیین می شود. کاربر ورژن برای زمانی است که قصد دارید در دیتابیس تغییری ایجاد کنید تا این تغییرات هنگام بروزرسانی اپلیکیشن توسط کاربر بر روی دیتابیس اعمال شود (مانند حذف یا اضافه کردن

یک ستون به جدول). هر بار که نیاز به بروزرسانی دیتابیس شد عدد ورژن باید یک واحد به آن اضافه شود تا سیستم عامل در هنگام نصب بروزرسانی اپلیکیشن متوجه تغییر در ساختار دیتابیس شده و آن را اعمال نماید. بنابراین در حال حاضر عدد ۱ را در نظر می گیرم. مرحله بعد تکمیل متد onCreate() است. مایلم ابتدا نام پارامتر SQLiteDatabase را به cdb تغییر دهم. حالا می بایست داخل این متد، کوئری مربوط به ساخت جدول را فراخوانی کنیم. در ابتدای مبحث با کوئری ها و همچنین ابزاری مانند SQLiteStudio آشنا شدیم. توصیه می کنم برای کاهش اشتباهات، به جای نوشتن دستی کوئری، ابتدا جدول مدنظر را در نرم افزار مدیریت دیتابیس ساخته و در نهایت کوئری که در خروجی نمایش می دهد را به پروژه اندرویدی خود منتقل کنید. با این کار مطمئنیم که دستوری را کم، زیاد و یا غلط ننوشته ایم. SQLiteStudio را باز کرده و جدولم را می سازم:

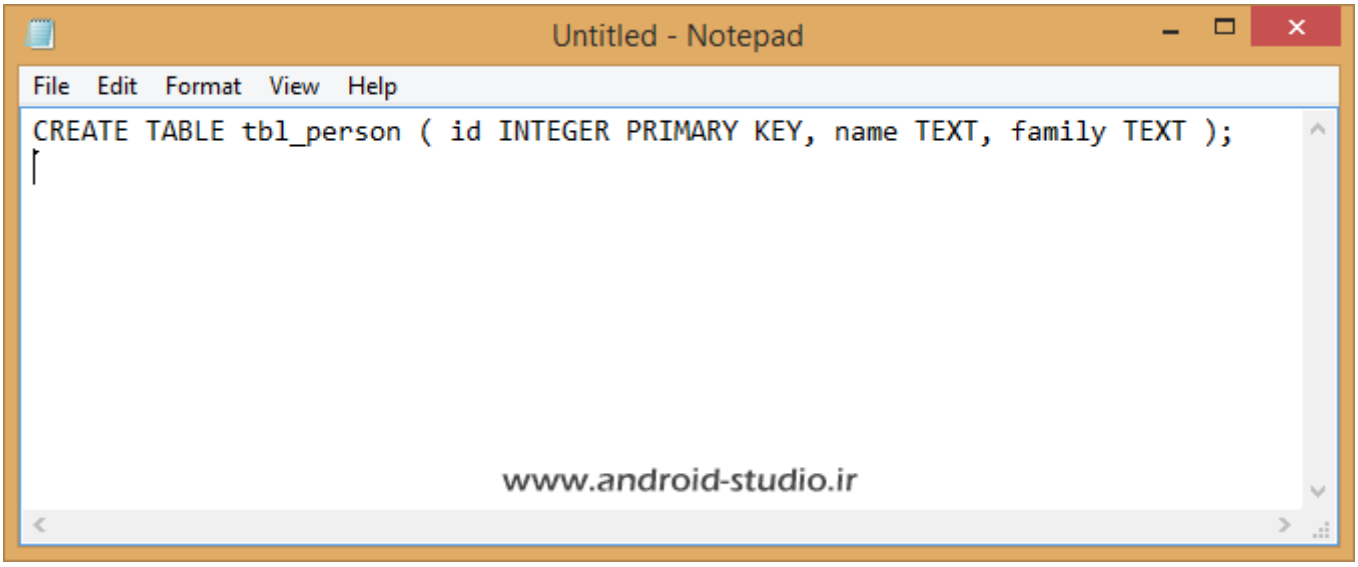


یک جدول با نام tbl_person و سه ستون به نام های id، name، family که id از جنس integer و primary key و ستون های name و family از جنس text هستند. قبلا اشاره کرده بودم از نظر بهینه بودن بهتر است فیلدهایی مانند نام که تعداد کاراکتر کمی را در آنها قرار می دهیم از نوع varchar تعیین کنیم، با این حال تفاوت چشمگیر نیست و در این پروژه از نوع text استفاده کردم. ضمنا قرار است

شناسه توسط کاربر و به صورت دستی وارد شود بنابراین Auto increment نیست. حالا کوئری را کپی کرده و درون یک متغیر از نوع String که نام cQuery را برای آن در نظر گرفته ام قرار می دهم:

```
@Override
public void onCreate(SQLiteDatabase cdb) {

    String cQuery = "CREATE TABLE tbl_person (\n" +
            "    id     INTEGER PRIMARY KEY,\n" +
            "    name   TEXT,\n" +
            "    family TEXT\n" +
            ");\n";

}
```

هر خط از کوئری بین دابل کوتیشن قرار گرفته که IDE به طور خودکار انتهای هر خط یک \n (که کار Enter را انجام می دهد) و یک + برای اتصال رشته ها به یکدیگر اضافه کرده است. من برای مرتب بودن کد، قبل از کپی کد به اندروید استودیو، درون NotePad فاصله های اضافی را حذف کرده و همچنین دستورات را در یک خط قرار می دهم:



حالا مجدد کوئری را درون cQuery قرار می دهم:

```
@Override
public void onCreate(SQLiteDatabase cdb) {

    String cQuery = "CREATE TABLE tbl_person ( id INTEGER PRIMARY KEY, name TEXT, family TEXT
);";

}
```

توجه : از ۲ سمی کالن (;) یکی مربوط به انتهای کوئری بوده و دیگری انتهای متغیر cQuery

با توجه به اینکه در چندجای پروژه نیاز به نام ستون ها داریم، بهتر است نام ستونها و جدول را نیز در متغیرهایی تعریف کنیم تا در کل کلاس، متغیرها را فراخوانی کرده و بازهم خطاهای احتمالی مربوط به غلط نوشتاری یا تغییر نامها در آینده و عدم اعمال تغییر در سراسر پروژه را کاهش دهیم:

```
package ir.android_studio.database;

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

public class DatabaseManager extends SQLiteOpenHelper {

    private static final String DatabaseName = "myinfo.db";
    private static final int Version = 1;
    private static final String TableName = "tbl_person";
    private static final String dID = "id";
    private static final String dName = "name";
    private static final String dFamily = "family";

    public DatabaseManager(Context cnt) {

        super(cnt, DatabaseName, null, Version);

    }

    @Override
    public void onCreate(SQLiteDatabase cdb) {

        String cQuery = "CREATE TABLE tbl_person ( id INTEGER PRIMARY KEY, name TEXT, family
TEXT );";

    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {

    }
}
```

سه متغیر با نام های dID، dName و dFamily به کلاس اضافه کردم. حالا متغیرها را درون کوئری جایگذاری می کنم:

```
@Override
public void onCreate(SQLiteDatabase cdb) {

    String cQuery = "CREATE TABLE " + TableName + " ( " + dID + " INTEGER PRIMARY KEY, " +
dName + " TEXT, " + dFamily + " TEXT );";

}
```

در نگاه اول جایگذاری مقداری گیج کننده است. اما با کمی دقت و مقایسه کوئری قدیم و جدید، تفاوت

را درک خواهید کرد. به عنوان مثال tbl_person با " + TableName + " جایگزین شده که در نهایت حاصل آن " + "tbl_person" + " است. حواستان به فضاهای خالی بین کلمات باشد. به عنوان مثال اگر

"CREATE TABLE "

را به صورت

"CREATE TABLE"

بنویسیم، نتیجه CREATE TABLEtbl_person می شود.

حالا باید کوئری را اجرا کنیم. متد execSQL() این وظیفه را به عهده دارد. exec مخفف execute به معنی اجرا می باشد. در نهایت متد onCreate() به این شکل تکمیل می شود:

```
@Override
public void onCreate(SQLiteDatabase cdb) {

    String cQuery = "CREATE TABLE " + TableName + " ( " + dID + " INTEGER PRIMARY KEY, " +
dName + " TEXT, " + dFamily + " TEXT );";
    cdb.execSQL(cQuery);

}
```

استفاده از Log کمک زیادی برای بررسی اتفاقات پشت صحنه می کند. برای مثال میخواهم موقع اجرای اپلیکیشن در Logcat ببینم آیا متد onCreate اجرا شده یا نه. قبلا با لاگ آشنا شدیم.

اندروید استودیو در تکمیل کدها توسعه دهنده را راهنمایی می کند. برای لاگ یک String به عنوان tag و یک String دیگر با عنوان message از ما خواسته شده. مزیت استفاده از تگ این است که در Logcat به راحتی می توان تگ را فیلتر کرده و فقط نتایجی که مربوط به آن فیلتر باشد را مشاهده کنیم.

توجه : متد سازنده و همچنین ()onCreate فقط یکبار و بعد از نصب اپلیکیشن اجرا می شود. مگر اینکه قصد ایجاد تغییر در جدول را داشته باشیم (تغییر عدد ورژن و اضافه کردن کوئری مربوطه در متد ()onUpgrade).

در این مبحث کاری به متد ()onUpgrade نداریم. در قدم بعد باید متدهای مربوط به ثبت، نمایش، بروزرسانی و حذف اطلاعات دیتابیس را اضافه کنیم. یک کلاس جدید میسازم که نام آن را Person به معنی شخص در نظر می گیرم. در این کلاس به تعداد ستون های جدول دیتابیس، متغییر میسازم:

```
package ir.android_studio.database;

public class Person {

    public String pID;
    public String pName;
    public String pFamily;

}
```

سه متغیر از جنس String و با نامهای pID، pName و pFamily.

نکته : ممکن است بپرسید در دیتابیس id از جنس int تعریف شده چرا اینجا pID را String تعریف کردم؟ اینجا مهم نیست متغیر از چه نوعی باشد. فقط باید کنترل کنیم کاربر اجازه وارد کردن کاراکترهای غیر عددی نداشته باشد که به راحتی با ویژگی inputType در رابط کاربری آنرا حل کرده ام.

به کلاس DatabaseManager برمیگردم. اولین متدی که به کلاس اضافه می کنم مربوط به ثبت اطلاعات است. یعنی دریافت داده ها از کاربر و ذخیره آنها در دیتابیس. بعد از متد ()onUpgrade یک متد public از نوع void با نام **دلخواه** insertPerson می سازم (اگر دلخواه بودن نامگذاری ها را درک کرده اید به یاداوریهای مکرر بنده خرده نگیرید). پارامتر ورودی این متد را از جنس Person و نام ipers تعیین می کنم:

```
public void insertPerson(Person iprs) {

}
```

قبل از ذخیره، بروزرسانی، واکشی و یا حذف اطلاعات لازم است یک اتصال به دیتابیس ایجاد شود. برای اتصال به دیتابیس دو متد در اختیار داریم:

- getReadableDatabase : که برای خواندن اطلاعات از دیتابیس استفاده می شود.

- getWritableDatabase : که برای نوشتن/خواندن اطلاعات استفاده می شود.

واضح است در این پروژه از متد اول برای نمایش اطلاعات به کاربر و متد دوم برای ذخیره، بروزرسانی و حذف اطلاعات استفاده خواهد شد.

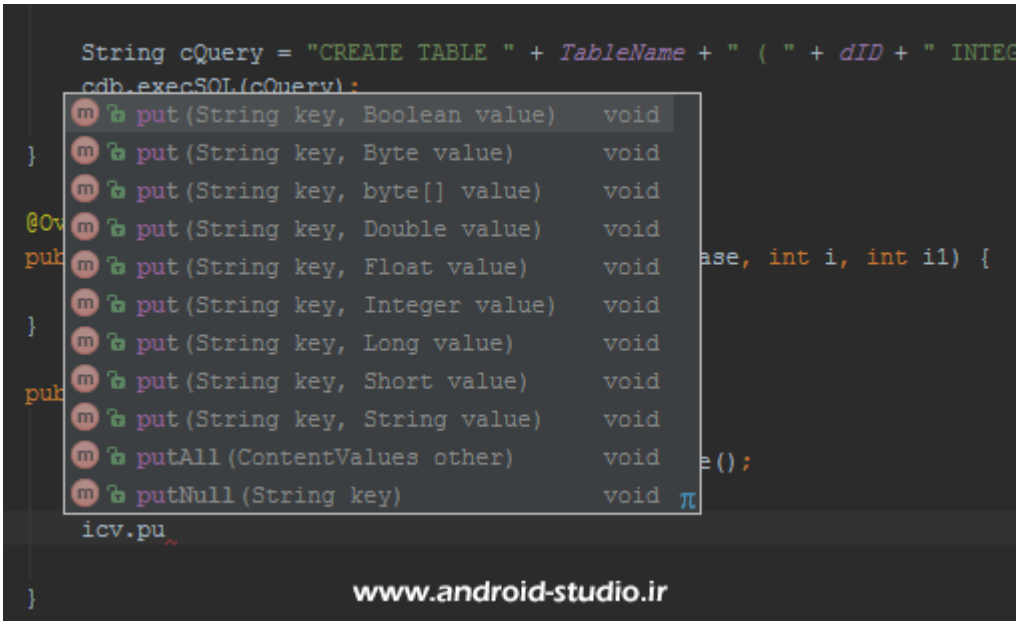
getWritableDatabase() را به اینصورت در متد insertPerson بکار می برم:

```
public void insertPerson(Person iprs) {

    SQLiteDatabase idb = this.getWritableDatabase();

}
```

ابتدا یک شیء از کلاس SQLiteDatabase با نام idb ساختم و برابر this.getWritableDatabase() قرار دادم. This یعنی در همین کلاس که از SQLiteOpenHelper مشتق شده، getWritableDatabase() فراخوانی شود و حاصل که ایجاد یک کانکشن نوشتنی بر روی دیتابیس است درون idb قرار گیرد. حالا برای انتقال داده ها به دیتابیس و یا بلعکس آن انتقال داده از دیتابیس به اکتیویتی قابل مشاهده برای کاربر، دو راه داریم. راه اول همان کوئری هست. یعنی مشابه آنچه در متد onCreate برای ایجاد جدول انجام دادیم. اما راه دوم استفاده از متد ContentValues است که بجز برای متد نمایش (واکشی) اطلاعات از دیتابیس که از کوئری استفاده می شود، مابقی را با ContentValues انجام می دهم. یک شیء با نام icv از این متد میسازم:

```
public void insertPerson(Person iprs) {

    SQLiteDatabase idb = this.getWritableDatabase();
    ContentValues icv = new ContentValues();

}
```

حالا با دستور put اطلاعاتی که کاربر در EditText ها وارد می کند را به دیتابیس پاس می دهم.



لیستی که با تایپ کردن تعدادی از حروف put (یا در سایر مواقع) نمایش داده می شود به عنوان راهنماست تا توسعه دهنده فورا ببیند این متد چه ورودی هایی می تواند داشته باشد و این که من از لیست کدام put را انتخاب کنم مهم نیست.

```
icv.put(dID, iprs.pID);
```

پارامتر اول dID است که مربوط به ستون id دیتابیس است. پارامتر دوم pID از iprs است که از کلاس Person ساخته بودیم. یعنی اطلاعات از iprs.pID دریافت و به dID منتقل می شود. به همین ترتیب برای دو فیلد دیگر (نام و نام خانوادگی) نیز تکرار می کنم:

```
icv.put(dID, iprs.pID);
icv.put(dName, iprs.pName);
icv.put(dFamily, iprs.pFamily);
```

در نهایت توسط متد insert اطلاعاتی که به وسیله put دریافت شده در دیتابیس ذخیره می شود:

```
idb.insert(TableName, null, icv);
```

در قسمت اول نام جدول، قسمت دوم چون مورد نیاز نیست مقدار null و قسمت آخر هم icv که از ContentValues ساخته بودیم. در نهایت کانکشنی که روی دیتابیس ایجاد کرده بودم را با متد close() می بندم:

```
idb.close();
```

علت اینکه لازم است کانکشن را ببندیم، اشغال بخشی (هرچند کم) از حافظه دستگاه می باشد. بنابراین برای اینکه اپلیکیشن از این جهت بهینه باشد و کمترین منابع را استفاده کند، بعد از اتمام کار با دیتابیس کانکشن را قطع می کنیم. کد نهایی insertPerson:

```
public void insertPerson(Person iprs) {

    SQLiteDatabase idb = this.getWritableDatabase();
    ContentValues icv = new ContentValues();
    icv.put(dID, iprs.pID);
    icv.put(dName, iprs.pName);
    icv.put(dFamily, iprs.pFamily);
    idb.insert(TableName, null, icv);
    idb.close();

}
```

سراغ اکتیویتی می روم. قبل از هر چیز لازم است یک شیء از کلاس DatabaseManager درون اکتیویتی ایجاد کنم تا به متدهای آن دسترسی داشته باشم. داخل بدنه MainActivity یک شیء با نام dbm ایجاد کرده:

DatabaseManager dbm;

و سپس dbm را داخل متد onCreate به اینصورت new می کنم:

dbm = new DatabaseManager(this);

کد MainActivity تا اینجای کار:

```
package ir.android_studio.database;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

    EditText edtName, edtFamily, edtID;
    Button btnInsert, btnView, btnUpdate, btnDelete;
    DatabaseManager dbm;

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

        edtName = (EditText) findViewById(R.id.edt_name);
        edtFamily = (EditText) findViewById(R.id.edt_family);
        edtID = (EditText) findViewById(R.id.edt_id);
        btnInsert = (Button) findViewById(R.id.btn_insert);
        btnView = (Button) findViewById(R.id.btn_view);
        btnUpdate = (Button) findViewById(R.id.btn_update);
        btnDelete = (Button) findViewById(R.id.btn_delete);
        dbm = new DatabaseManager(this);

    }
}
```

حالا باید درون اکتیویتی، کاری کنیم با لمس دکمه "ذخیره" اطلاعات وارد شده توسط کاربر به کلاس Person منتقل شود. متد setOnClickListener را برای دکمه btnInsert می سازم:

```
btnInsert.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

    }
});
```

سپس یک متغیر از جنس int و دو متغیر از جنس String داخل متد onClick اضافه می کنم تا داده های وارد شده توسط کاربر درون این متغیرها ریخته شود:

```
btnInsert.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        String mID = edtID.getText().toString();
        String mName = edtName.getText().toString();
        String mFamily = edtFamily.getText().toString();

    }
});
```

حالا یک نمونه (شیء) از Person میسازم تا داده هایی که داخل این متغیرها ریخته شده را به کلاس Person انتقال دهم:

```
btnInsert.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        String mID = edtID.getText().toString();
        String mName = edtName.getText().toString();
        String mFamily = edtFamily.getText().toString();
        Person iperson = new Person();
        iperson.pID = mID;
        iperson.pName = mName;
        iperson.pFamily = mFamily;

    }
});
```

حالا باید اطلاعات موجود در iperson را به insertPerson بفرستیم تا در دیتابیس ذخیره شود:

```
btnInsert.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        int mID = Integer.parseInt(edtID.getText().toString());
        String mName = edtName.getText().toString();
        String mFamily = edtFamily.getText().toString();
        Person iperson = new Person();
        iperson.pID = mID;
        iperson.pName = mName;
        iperson.pFamily = mFamily;
        dbm.insertPerson(iperson);

    }
});
```

dbm یک نمونه از DatabaseManager بود. dbm.insertPerson(iperson) یعنی iperson را که شامل mID، mName و mFamily می باشد را به متد insertPerson کلاس DatabaseManager بفرست. بهتر است برای این قسمت شرطی تعریف کنیم تا اگر کاربر فیلدها را تکمیل نکرده باشد پیغامی مبنی بر لزوم تکمیل اطلاعات خواسته شده دریافت کند و تا قبل از تکمیل هر سه فیلد، داده ای به سمت دیتابیس ارسال نشود:

```
btnInsert.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        String mID = edtID.getText().toString();
        String mName = edtName.getText().toString();
        String mFamily = edtFamily.getText().toString();

        if (TextUtils.isEmpty(mID) || TextUtils.isEmpty(mName) || TextUtils.isEmpty(mFamily))
{

   Toast.makeText(MainActivity.this, "شود تکمیل باید فیلدها تمامی", Toast.LENGTH_LONG).show();

        }

        else {

            Person iperson = new Person();
            iperson.pID = mID;
            iperson.pName = mName;
            iperson.pFamily = mFamily;
            dbm.insertPerson(iperson);
            Toast.makeText(MainActivity.this, "شد ذخیره موفقیت با اطلاعات",
Toast.LENGTH_LONG).show();

        }
    }
});
```

یک if else اضافه کردم. مقابل if و درون پرانتز شرطی تعریف کردم که هر سه فیلد ورودی را چک می کند که حداقل یک کاراکتر باید در هر کدام از فیلدهای ورودی وارد شده و empty (خالی) نباشد:

```
TextUtils.isEmpty(mID) || TextUtils.isEmpty(mName) || TextUtils.isEmpty(mFamily)
```

TextUtils.isEmpty(mID) چک می کند که آیا از EditText با شناسه mID حداقل یک کاراکتر دریافت می کند یا خیر. و نتیجه را برمیگرداند. من این دستور را برای هر سه فیلد ورودی تکرار کرده ام. || به معنی "یا" می باشد. یعنی mID یا mName یا mFamily در هرکدام شرط برقرار بود... . (|| دو خط عمودی است در کنار یکدیگر. عموما با کلید "\" روی کیبورد مشترک است. کنار کلید "}").

البته برای بررسی خالی بودن یا نبودن ورودی راههای دیگری مانند متد ()equals هم هست اما من ترجیح دادم از TextUtils استفاده کنم.

حالا اگر این شروط (یا حداقل یکی از آنها برقرار بود، دستور داخل {آکولاد} اول اجرا در غیر اینصورت {آکولاد} بعد از else اجرا خواهد شد. برای نمایش پیغام روی صفحه از Toast استفاده کردم. این متد برای نمایش پیغام شناور کوتاه مدت بکار می رود:

```
Toast.makeText()
```

بعد از makeText درون پرانتز ۳ ورودی نیاز دارد. اولی Context بوده که در اینجا MainActivity.this می گذاریم (یعنی هدف ما همین اکتیویتی جاری هست). در قسمت دوم پیغامی که قصد داریم برای کاربر نمایش داده شود و در قسمت نهایی مدت زمان نمایش پیغام را تعیین می کنیم. دو حالت LENGTH_LONG و LENGTH_SHORT قابل انتخاب هست که از نام آنها پیداست LONG مدت بیشتری پیغام را روی صفحه نگه می دارد. من از LONG استفاده کردم اما شما یک بار هم با SHORT تست کنید تا تفاوت را مشاهده کنید. در نهایت با ()show. به برنامه اعلام می کنم قصد نمایش پیغام را دارم:

```
Toast.makeText(MainActivity.this, "تمامی فیلدها باید تکمیل شود", Toast.LENGTH_LONG).show();
```

در نتیجه اگر کاربر هرکدام از فیلدها را خالی رها کند و دکمه ذخیره را لمس کند، پیغام "تمامی فیلدها باید تکمیل شود" روی صفحه ظاهر خواهد شد و عمل ثبت اطلاعات در دیتابیس انجام نمی شود. مشابه این پیغام هم در قسمت دوم عبارت شرطی اضافه کردم تا اگر اطلاعات با موفقیت ثبت شد پیغام "اطلاعات با موفقیت ثبت شد" ظاهر شود.

خب! الان برای اولین اجرای پروژه آماده ایم. قبل از اجرا به متدها Log اضافه کردم تا عملکرد برنامه را در قسمت Logcat بررسی کنم:

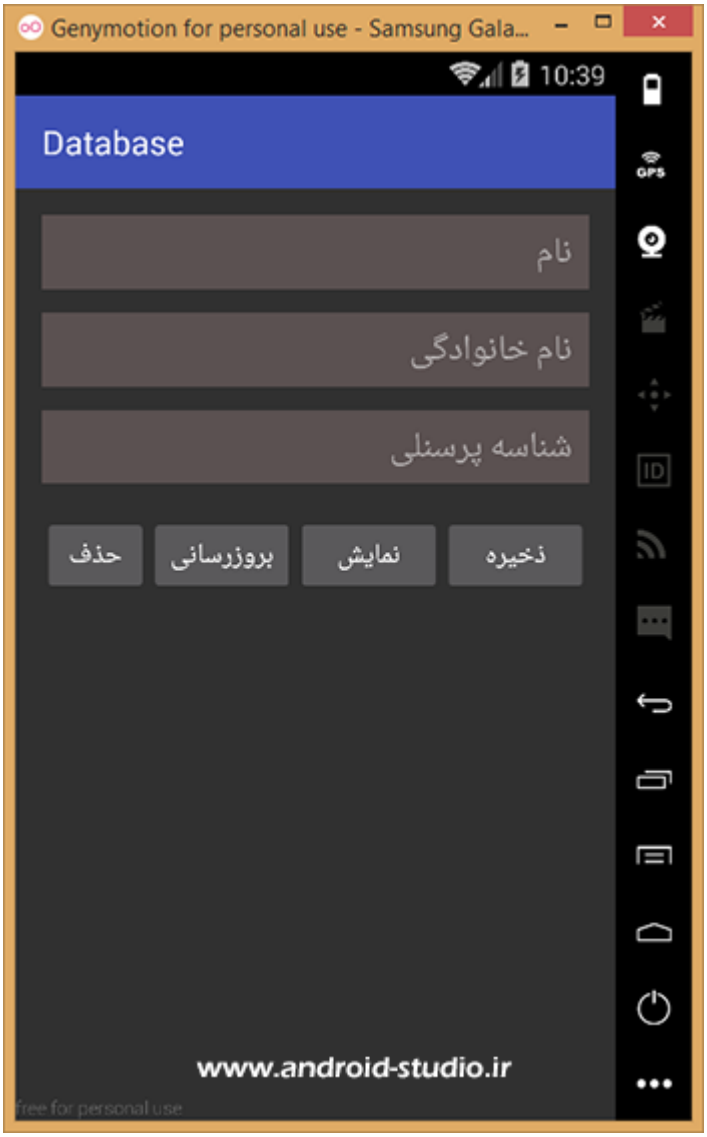
کلاس DatabaseManager.java:

```
package ir.android_studio.database;

import android.content.ContentValues;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class DatabaseManager extends SQLiteOpenHelper {

    private static final String DatabaseName = "myinfo.db";
    private static final int Version = 1;
    private static final String TableName = "tbl_person";
    private static final String dID = "id";
    private static final String dName = "name";
    private static final String dFamily = "family";

    public DatabaseManager(Context cnt) {

        super(cnt, DatabaseName, null, Version);
        Log.i("Mahdi", "Database Created!");

    }

    @Override
    public void onCreate(SQLiteDatabase cdb) {

        String cQuery = "CREATE TABLE " + TableName + " ( " + dID + " INTEGER PRIMARY KEY, " +
dName + " TEXT, " + dFamily + " TEXT );";
        cdb.execSQL(cQuery);
        Log.i("Mahdi", "Table Created!");

    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {

    }

    public void insertPerson(Person iprs) {

        SQLiteDatabase idb = this.getWritableDatabase();
        ContentValues icv = new ContentValues();
        icv.put(dID, iprs.pID);
        icv.put(dName, iprs.pName);
        icv.put(dFamily, iprs.pFamily);
        idb.insert(TableName, null, icv);
        idb.close();
        Log.i("Mahdi", "insertPerson Method");

    }
}
```

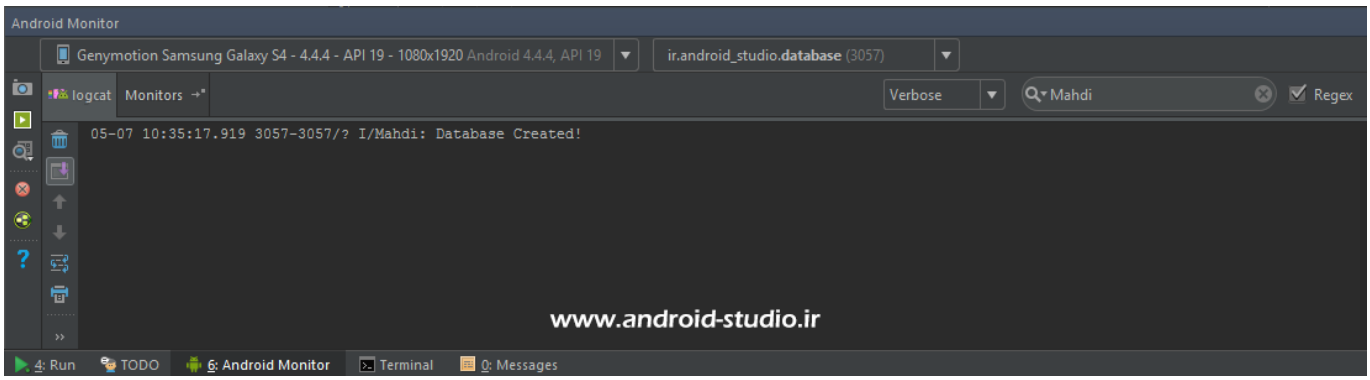
کلاس MainActivity.java:

```
package ir.android_studio.database;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RelativeLayout;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    EditText edtName, edtFamily, edtID;
    Button btnInsert, btnView, btnUpdate, btnDelete;
    DatabaseManager dbm;

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

        edtName = (EditText) findViewById(R.id.edt_name);
        edtFamily = (EditText) findViewById(R.id.edt_family);
        edtID = (EditText) findViewById(R.id.edt_id);
        btnInsert = (Button) findViewById(R.id.btn_insert);
        btnView = (Button) findViewById(R.id.btn_view);
        btnUpdate = (Button) findViewById(R.id.btn_update);
        btnDelete = (Button) findViewById(R.id.btn_delete);
        dbm = new DatabaseManager(this);

        btnInsert.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                Log.i("Mahdi", "insertButton Clicked!");
                String mID = edtID.getText().toString();
                String mName = edtName.getText().toString();
                String mFamily = edtFamily.getText().toString();

                if (TextUtils.isEmpty(mID) || TextUtils.isEmpty(mName) ||
TextUtils.isEmpty(mFamily)) {

                    Toast.makeText(MainActivity.this, "شود تکمیل باید فیلدها تمامی",
Toast.LENGTH_LONG).show();

                }

                else {

                    Person iperson = new Person();
                    iperson.pID = mID;
                    iperson.pName = mName;
                    iperson.pFamily = mFamily;
                    dbm.insertPerson(iperson);
                    Toast.makeText(MainActivity.this, "شد ذخیره موفقیت با اطلاعات",
Toast.LENGTH_LONG).show();
                    Log.i("Mahdi", "Data inserted!");

                }
            }
        });

    }
}
```

پروژه را اجرا می کنم:

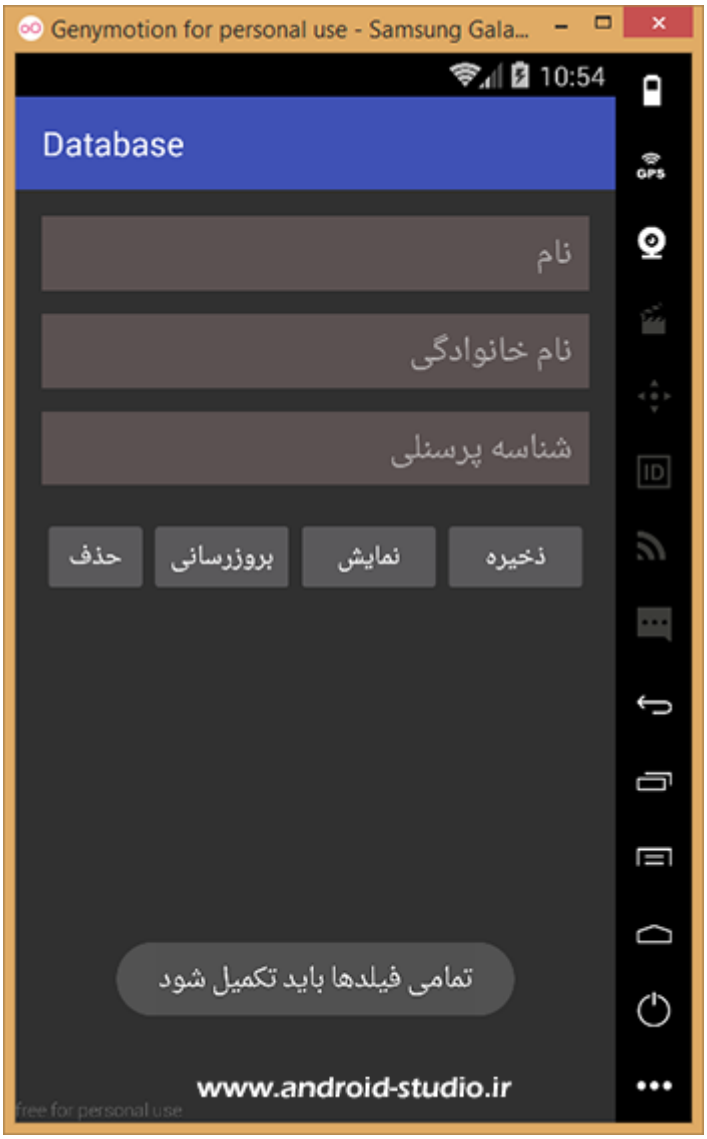


Logcat را هم باز می کنم:

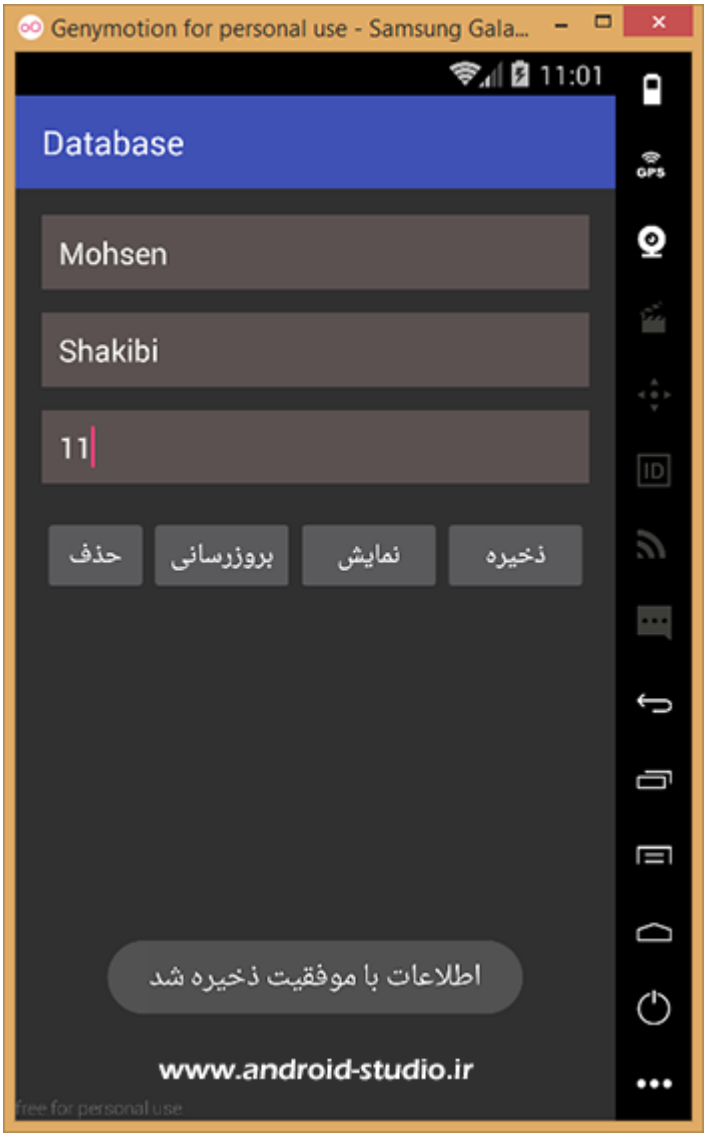


در قسمت Device دقت کنید دیوایس در حال اجرا انتخاب شده باشد نه دیوایسی که قبلا استفاده کرده اید. در قسمت فیلتر من تگ Mahdi را وارد کردم تا فقط لاگ هایی نمایش داده شود که خودم تعریف

کرده ام. مشاهده می کنید بلافاصله بعد از اجرا لاگ Database Created چاپ شد. یعنی متد سازنده اجرا شده است. قبل از اینکه درون فیلدها چیزی بنویسم یک بار دکمه ذخیره را کلیک می کنم. اگر کد من مشکلی نداشته باشد انتظار دارم پیغام "تمامی فیلدها باید تکمیل شود" ظاهر شود که نتیجه نیز همین بود:



لاگ مربوط به دکمه نیز چاپ شد. حالا فیلدها را تکمیل و مجدد تایید می کنم:

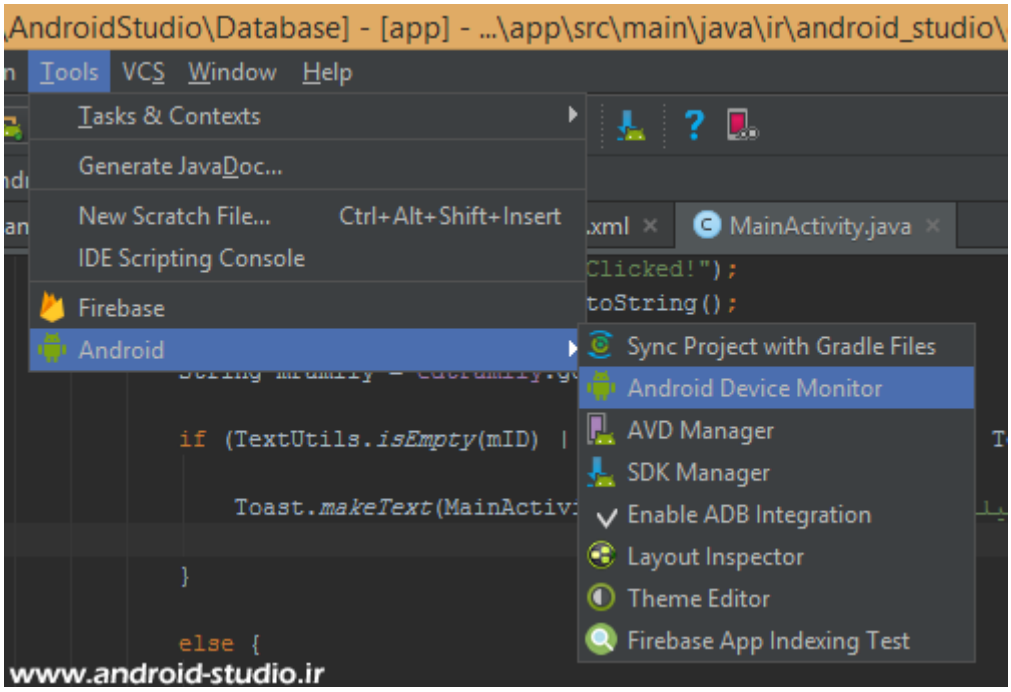


این بار پیغام "اطلاعات با موفقیت ذخیره شد" روی صفحه ظاهر گردید. به لاگها دقت کنید.بعد از لمس دکمه، ابتدا متد ساخت جدول اجرا شده، سپس insertPerson و در نهایت اطلاعات ثبت شده است. اگر پروژه را Stop و مجدد Run کنید این بار باید لاگ مربوط به سازنده را ببینید اما ساخت جدول نه. چون در هر بار اجرا اندروید چک می کند اگر دیتابیس قبلا ساخته نشده آنرا بسازد اما وقتی دیتابیس و جدول قبلا ساخته شده اند، متد onCreate که برای ایجاد جدول بود مجدد اجرا و بررسی نخواهد شد.

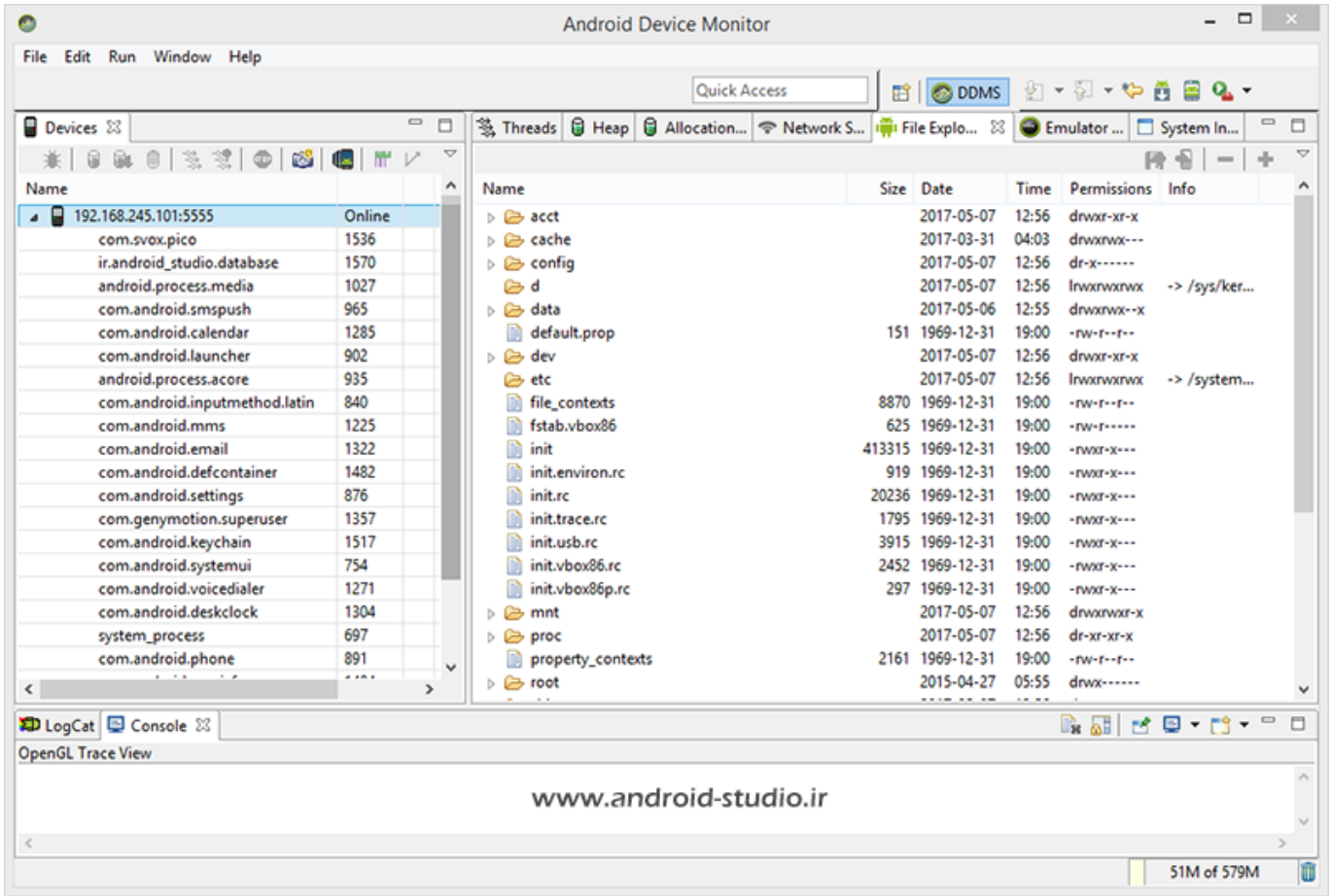
نکته: در مواردی ممکن است متد سازنده دیتابیس نیز هنگام ثبت اولین داده اجرا شود و نه هنگام اولین اجرای پروژه.

شاید مایل باشید به فایل دیتابیسی که روی دیوایس ساخته شده دسترسی داشته باشید یا به مشکل بر بخورید و لازم باشد چک بکنید آیا دیتابیس واقعا ایجاد شده یا نه. (البته در تستی که من انجام دادم دسترسی به فایلها فقط در شبیه سازها میسر است و روی دیوایس حقیقی در حالت عادی این دسترسی امکانپذیر نیست).

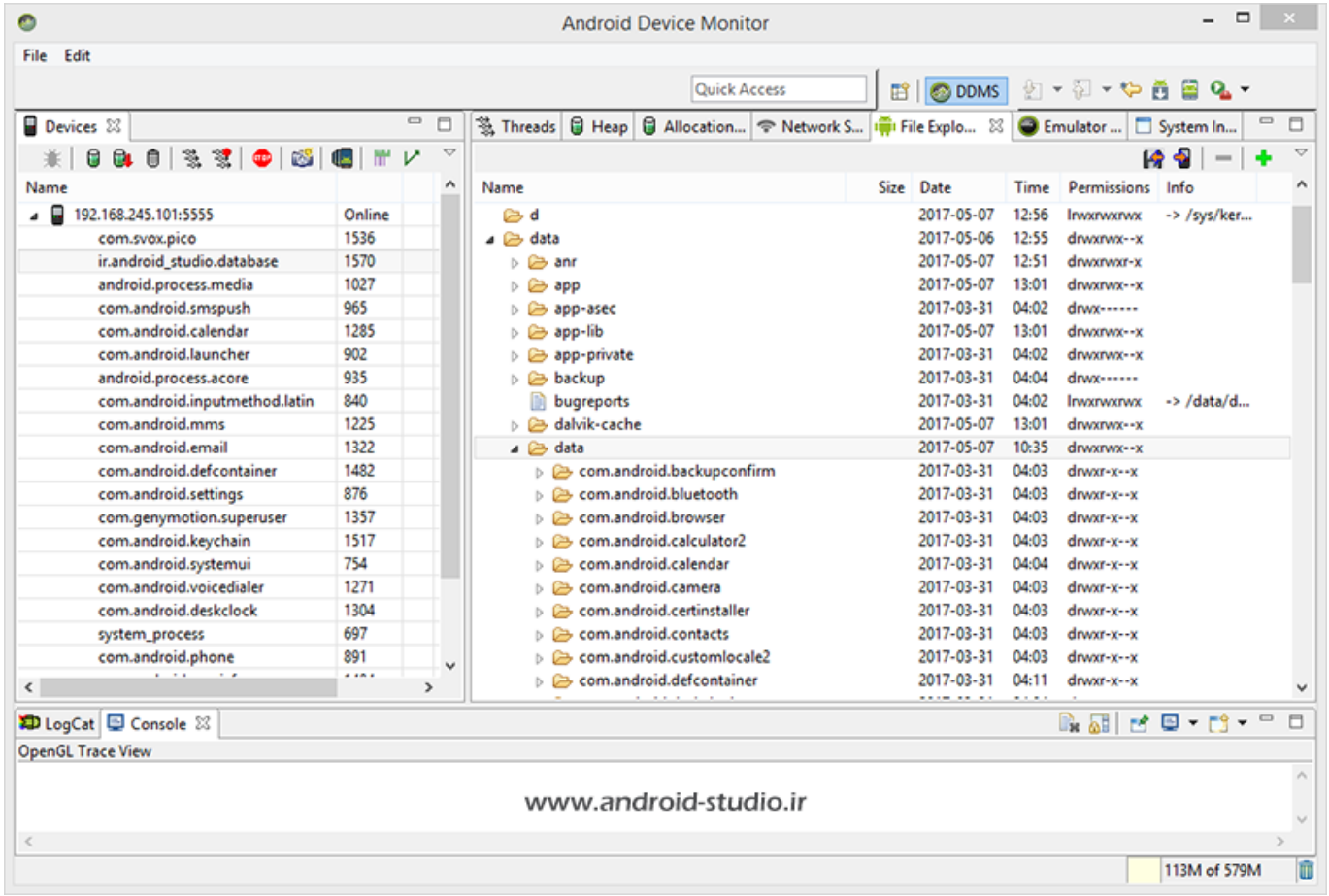
به بخش Android Device Monitor می روم:

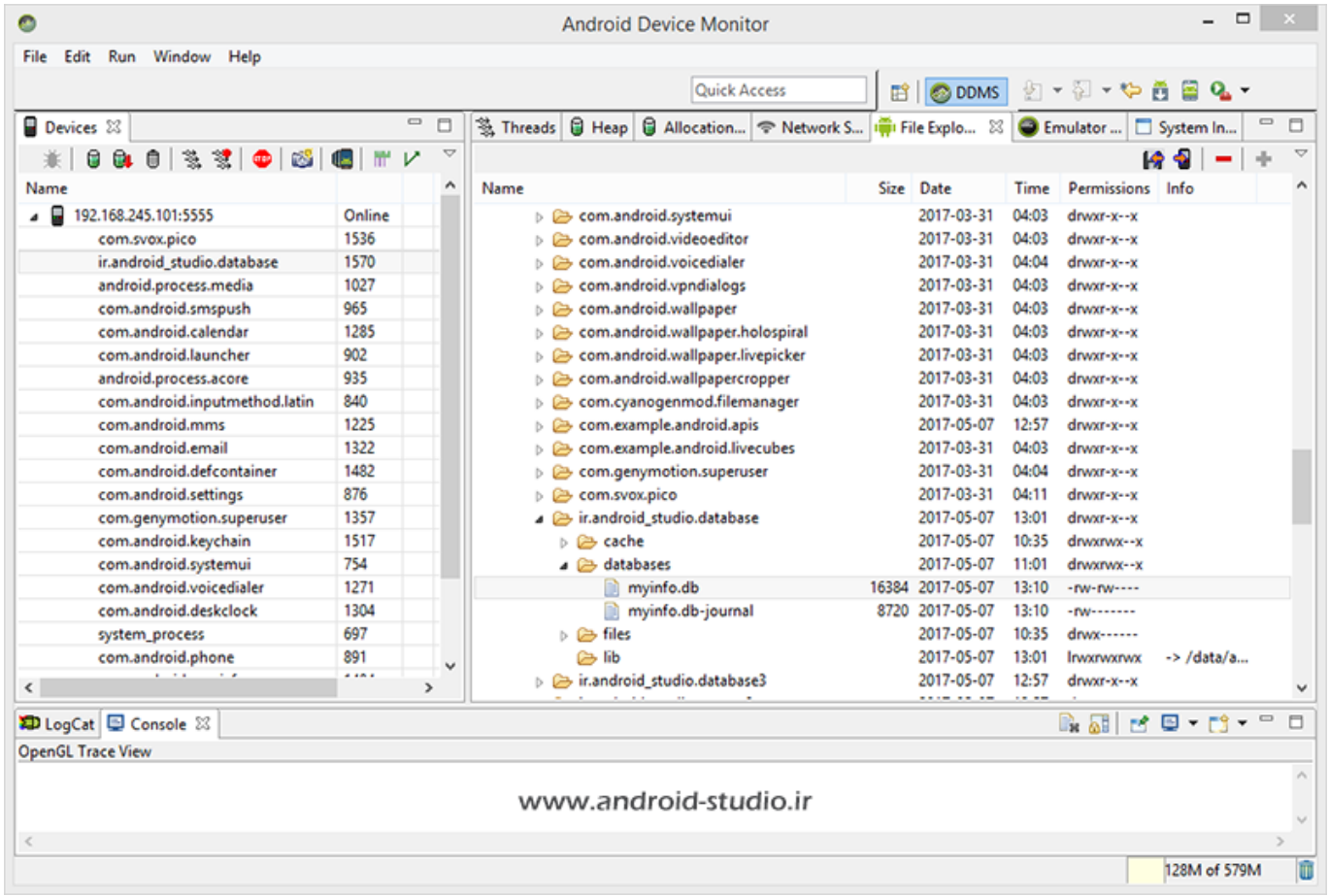


با پیغام Following debug... اگر مواجه شدید تایید کنید. باز شدن پنجره جدید ممکن است چند ثانیه طول بکشد و علاوه بر این احتمالا این پنجره در پشت اندروید استودیو باز شود و ابتدا متوجه باز شدن آن نشوید.

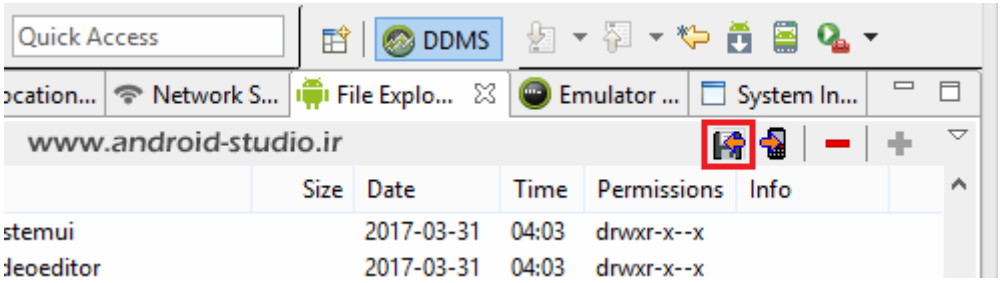


در سمت چپ لیست اپلیکیشن های نصب شده روی دیوایس را مشاهده می کنید که می بایست اپلیکیشن خود را انتخاب کنیم. پس از انتخاب اپلیکیشن، در سمت راست و برگه File Explorer به مسیر data > data رفته و پوشه مربوط به اپلیکیشن خودم را باز می کنم:

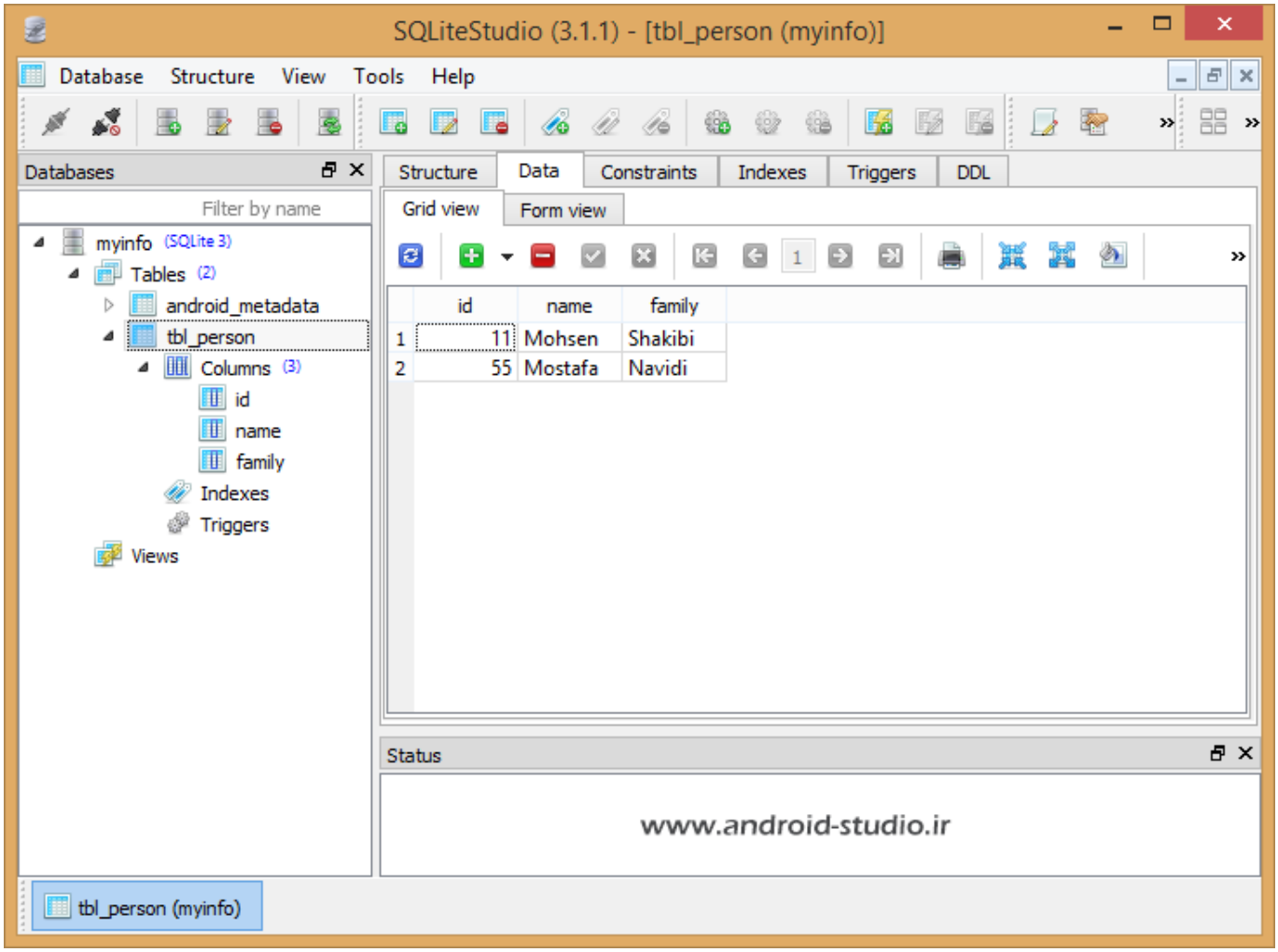




پوشه databases و داخل آن فایل myinfo.db وجود دارد. اگر این پوشه ایجاد نشده یعنی پروژه مشکلی داشته که نتوانسته دیتابیس را بسازد. می خواهم فایل دیتابیس را روی دسکتاپ ذخیره کرده و با SQLiteStudio باز کنم. آیکون ذخیره فایل را انتخاب می کنم:



فایل ذخیره شد. حالا آن را باز می کنم. در SQLiteStudio در مسیر Database > add a database و سپس در قسمت File ، با انتخاب آیکون پوشه، دیتابیس را انتخاب می کنم.



تا اینجای کار مشکلی نداشتیم.

اگر در حین تکمیل پروژه خطایی دریافت می کنید به دقت آنرا مطالعه کنید. در اکثر مواقع خطاها دلایل ساده ای مثل اشتباه تایپی یا جا افتادن یک کاراکتر دارند. یا اگر هنگام ران کردن پروژه ارور گرفتید، لاگ را بررسی کنید. به خصوص خط های قرمز رنگ. ممکن است تعداد خطوط زیاد باشد اما با یک بررسی ساده در بیشتر مواقع اشکال را پیدا می کنید. حتی محل وقوع خطا و شماره خطی که در آن خطا رخ داده را اعلام می کند. در نهایت اگر شخصا به نتیجه نرسیدید درباره آن سرچ کنید. در اکثر موارد با سرچ قسمتی از خطا و بدون اضافه کردن توضیحات اضافه انگلیسی و فارسی، به راه حلهای متعددی در وب سایتهای مختلف از جمله stackoverflow.com می رسید. مشکلی که الان شما گرفتارش شدید صدها و بلکه هزاران نفر قبل از شما هم با آن برخورد کرده اند. پس نیازی نیست برای هر خطا و مشکلی سراغ اشخاص و انجمن ها بروید، آنرا مطرح کنید و ساعتها یا روزها منتظر پاسخ بمانید!

سراغ متد بعدی می روم. دکمه دوم "نمایش" است که با وارد کردن شناسه توسط کاربر در فیلد مربوطه و زدن دکمه نمایش، نام و نام خانوادگی مرتبط با آن شناسه از دیتابیس خوانده و به کاربر نمایش داده

شود. یک متد جدید با نام getPerson و از نوع Person می سازم که ورودی آن از نوع String و نام gID می باشد:

```
public Person getPerson(String gID) {


}
```

ابتدا یک نمونه از متد Person با نام gPrs درون متد ایجاد می کنم. با توجه به توضیحات قبل، در این متد به getReadableDatabase نیاز داریم. یک نمونه هم از SQLiteDatabase با نام gdb می سازم:

```
public Person getPerson(String gID) {

    Person gPrs = new Person();
    SQLiteDatabase gdb = this.getReadableDatabase();

}
```

برای دریافت اطلاعات از دیتابیس از کوئری استفاده می کنم:

```
String gQuery = "SELECT * FROM tbl_person  WHERE id = gID";
```

ابتدا متد کامل شده را نشان می دهم و در ادامه توضیح می دهم:

```
public Person getPerson(String gID) {

    Person gPrs = new Person();
    SQLiteDatabase gdb = this.getReadableDatabase();
    String gQuery = "SELECT * FROM " + TableName + " WHERE " + dID + "=" + gID;
    Cursor gCur = gdb.rawQuery(gQuery, null);
    if (gCur.moveToFirst()) {
        gPrs.pName = gCur.getString(1);
        gPrs.pFamily = gCur.getString(2);
    }
    Log.i("Mahdi", "getPerson Method");
    return gPrs;

}
```

در خط gQuery مانند کوئری ساخت جدول، نام جدول و ستون id را از متغیرها فراخوانی کردم. به واسطه ارتباطی که توسط gdb به دیتابیس ایجاد شده، متد rawQuery دستور gQuery را اجرا کرده و حاصل (سطر یا سطرهایی که با شرط ما همخوانی دارند) را درون gCur می ریزد. حالا درون عبارت شرطی if و با استفاده از moveToFirst حاصلی که درون gCur ریخته شده را خط به خط برمی گرداند (که اینجا چون شناسه هر شخص یکتاست فقط یک سطر داریم) . در نهایت ستون نام و نام خانوادگی سطری

که برگردانده شده را در کلاس Person قرار می دهد. gCur.getString(1) یعنی رشته داخل اندیس شماره ۱ سطر (یعنی ستون نام) را داخل متغیر pName بریز. همچنین gCur.getString(2) برای اندیس شماره ۲ یعنی نام خانوادگی این عمل را تکرار می کند.

نکته : شماره اندیس ها از صفر شروع می شود. بنابراین اندیس ستون id برابر صفر، اندیس نام برابر ۱ و اندیس نام خانوادگی برابر ۲ می باشد.

در نهایت Person را return می کنم. اگر return انجام نشود زیر آکولاد "}" انتهای متد ارور Missing return statement می گیرید.

نکته : بعد از return نمی توانید دستوری به متد اضافه کنید. به عبارتی هرچه بعد از خط return بنویسید اصلا اجرا نمی شود! بنابراین لاگ را قبل از return اضافه کردم.

به سراغ MainActivity رفته و رویداد کلیک را برای دکمه نمایش تعریف می کنم.

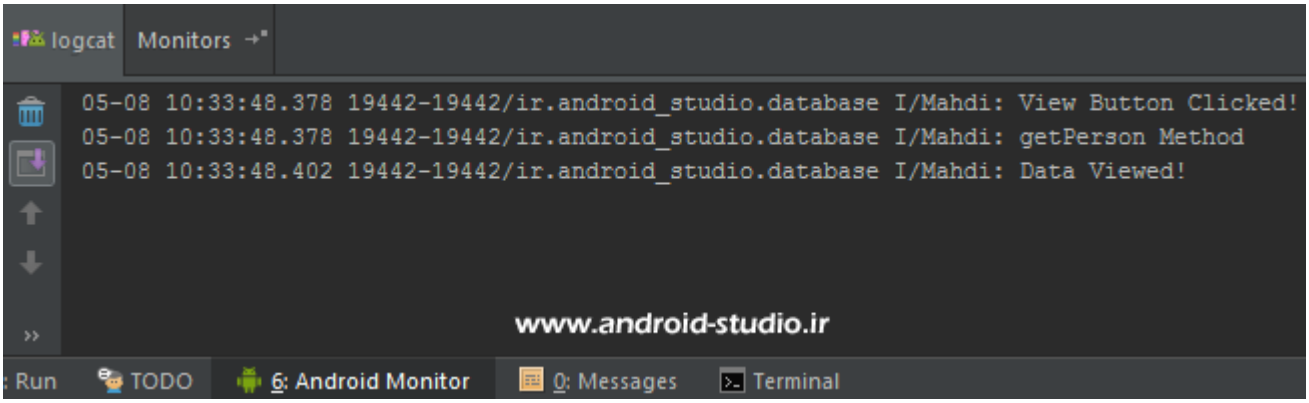
```
btnView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        Log.i("Mahdi", "View Button Clicked!");
        String vID = edtID.getText().toString();
        Person vPer = dbm.getPerson(vID);
        edtName.setText(vPer.pName);
        edtFamily.setText(vPer.pFamily);
        Log.i("Mahdi", "Data Viewed!");

    }
});
```

ابتدا یک متغیر از نوع String با نام vID ساختم تا شناسه ای که کاربر وارد کرده را ذخیره کند. در خط بعد یک شیء جدید از Person با نام vPer ایجاد کردم که حاصل dbm.getPerson(vID) یعنی نام و نام خانوادگی مرتبط با شناسه وارد شده درون آن قرار می گیرد. در نهایت توسط متد setText اطلاعات را از vPer به EditText ها منتقل کردم.

مجدد پروژه را اجرا و شناسه ۱۱ که قبلا ذخیره کرده بودم را وارد می کنم:


Genymotion for personal use - Samsung Gala...
10:27
Database
نام
نام خانوادگی
11
ذخیره
نمایش
بروزرسانی
حذف
www.android-studio.ir
free for personal use


Genymotion for personal use - Samsung Gala...
10:31
Database
Mohsen
Shakibi
11
حذف
بروزرسانی
نمایش
ذخیره
www.android-studio.ir
free for personal use


logcat
Monitors
05-08 10:33:48.378 19442-19442/ir.android_studio.database I/Mahdi: View Button Clicked!
05-08 10:33:48.378 19442-19442/ir.android_studio.database I/Mahdi: getPerson Method
05-08 10:33:48.402 19442-19442/ir.android_studio.database I/Mahdi: Data Viewed!
www.android-studio.ir
Run
TODO
6: Android Monitor
0: Messages
Terminal

:DatabaseManager

```
package ir.android_studio.database;

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

public class DatabaseManager extends SQLiteOpenHelper {

    private static final String DatabaseName = "myinfo.db";
    private static final int Version = 1;
    private static final String TableName = "tbl_person";
    private static final String dID = "id";
    private static final String dName = "name";
    private static final String dFamily = "family";

    public DatabaseManager(Context cnt) {

        super(cnt, DatabaseName, null, Version);
        Log.i("Mahdi", "Database Created!");

    }

    @Override
    public void onCreate(SQLiteDatabase cdb) {

        String cQuery = "CREATE TABLE " + TableName + " ( " + dID + " INTEGER PRIMARY KEY, " +
dName + " TEXT, " + dFamily + " TEXT );";
        cdb.execSQL(cQuery);
        Log.i("Mahdi", "Table Created!");

    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {

    }

    public void insertPerson(Person iprs) {

        SQLiteDatabase idb = this.getWritableDatabase();
        ContentValues icv = new ContentValues();
        icv.put(dID, iprs.pID);
        icv.put(dName, iprs.pName);
        icv.put(dFamily, iprs.pFamily);
        idb.insert(TableName, null, icv);
        idb.close();
        Log.i("Mahdi", "insertPerson Method");

    }

    public Person getPerson(String gID) {

        Person gPrs = new Person();
        SQLiteDatabase gdb = this.getReadableDatabase();
        String gQuery = "SELECT * FROM " + TableName + " WHERE " + dID + "=" + gID;
        Cursor gCur = gdb.rawQuery(gQuery, null);
        if (gCur.moveToFirst()) {
            gPrs.pName = gCur.getString(1);
```

```
            gPrs.pFamily = gCur.getString(2);
        }
        Log.i("Mahdi", "getPerson Method");
        return gPrs;

    }
}
```

:MainActivity

```
package ir.android_studio.database;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;

import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RelativeLayout;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    EditText edtName, edtFamily, edtID;
    Button btnInsert, btnView, btnUpdate, btnDelete;
    DatabaseManager dbm;

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

        edtName = (EditText) findViewById(R.id.edt_name);
        edtFamily = (EditText) findViewById(R.id.edt_family);
        edtID = (EditText) findViewById(R.id.edt_id);
        btnInsert = (Button) findViewById(R.id.btn_insert);
        btnView = (Button) findViewById(R.id.btn_view);
        btnUpdate = (Button) findViewById(R.id.btn_update);
        btnDelete = (Button) findViewById(R.id.btn_delete);
        dbm = new DatabaseManager(this);

        btnInsert.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                Log.i("Mahdi", "insertButton Clicked!");
                String mID = edtID.getText().toString();
                String mName = edtName.getText().toString();
                String mFamily = edtFamily.getText().toString();

                if (TextUtils.isEmpty(mID) || TextUtils.isEmpty(mName) ||
TextUtils.isEmpty(mFamily)) {

                    Toast.makeText(MainActivity.this, "شود تکمیل باید فیلدها تمامی",
Toast.LENGTH_LONG).show();

                }

                else {

                    Person iperson = new Person();
                    iperson.pID = mID;
```

```
                    iperson.pName = mName;
                    iperson.pFamily = mFamily;
                    dbm.insertPerson(iperson);
                    Toast.makeText(MainActivity.this, "شد ذخیره موفقیت با اطلاعات",
Toast.LENGTH_LONG).show();
                    Log.i("Mahdi", "Data inserted!");

                }
            }
        });

        btnView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                Log.i("Mahdi", "View Button Clicked!");
                String vID = edtID.getText().toString();
                Person vPer = dbm.getPerson(vID);
                edtName.setText(vPer.pName);
                edtFamily.setText(vPer.pFamily);
                Log.i("Mahdi", "Data Viewed!");

            }
        });

    }
}
```

نوبت به گزینه بروزرسانی می رسد. داخل کلاس دیتابیس متد جدیدی با نام updatePerson می سازم.

متد تکمیل شده:

```
public void updatePerson(Person uprs) {

    SQLiteDatabase udb = this.getWritableDatabase();
    ContentValues ucv = new ContentValues();
    ucv.put(dName, uprs.pName);
    ucv.put(dFamily, uprs.pFamily);
    udb.update(TableName, ucv, dID + " = ?", new String[] {String.valueOf(uprs.pID)});
    Log.i("Mahdi", "updatePerson Method");

}
```

۴ خط ابتدا مشابه متد insertPerson است که نیاز به توضیح مجدد ندارد. در ادامه توسط متد update عمل بروزرسانی اطلاعات انجام می شود به اینصورت که ابتدا شناسه ای که کاربر وارد کرده بررسی می شود و در صورت وجود سطری با این شناسه، دو ستون دیگر یعنی نام و نام خانوادگی که کاربر وارد کرده جایگزین اطلاعات قبلی می گردد. متد update چهار پارامتر ورودی دارد که اولین آن نام جدول، دومی نمونه ی ساخته شده از ContentValues، مورد سوم رشته ای با ساختار کلی "id = ?" که id همان ستون شناسه دیتابیس بوده و پارامتر چهارم به صورت

new String[] {id}

که توسط String.valueOf(uprs.pID) شناسه دریافتی از کاربر بدست می آید.

متد مربوط به دکمه بروزرسانی در MainActivity نیز مشابه دکمه ذخیره بوده و نیاز به توضیح ندارد:

```
btnUpdate.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        String uID = edtID.getText().toString();
        String uName = edtName.getText().toString();
        String uFamily = edtFamily.getText().toString();

        Person uperson = new Person();
        uperson.pID = uID;
        uperson.pName = uName;
        uperson.pFamily = uFamily;
        dbm.updatePerson(uperson);

    }
});
```

:DatabaseManager

```
package ir.android_studio.database;

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

public class DatabaseManager extends SQLiteOpenHelper {

    private static final String DatabaseName = "myinfo.db";
    private static final int Version = 1;
    private static final String TableName = "tbl_person";
    private static final String dID = "id";
    private static final String dName = "name";
    private static final String dFamily = "family";

    public DatabaseManager(Context cnt) {

        super(cnt, DatabaseName, null, Version);
        Log.i("Mahdi", "Database Created!");

    }

    @Override
    public void onCreate(SQLiteDatabase cdb) {

        String cQuery = "CREATE TABLE " + TableName + " ( " + dID + " INTEGER PRIMARY KEY, " +
dName + " TEXT, " + dFamily + " TEXT );";
        cdb.execSQL(cQuery);
        Log.i("Mahdi", "Table Created!");

    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {

    }
```

```
    public void insertPerson(Person iprs) {

        SQLiteDatabase idb = this.getWritableDatabase();
        ContentValues icv = new ContentValues();
        icv.put(dID, iprs.pID);
        icv.put(dName, iprs.pName);
        icv.put(dFamily, iprs.pFamily);
        idb.insert(TableName, null, icv);
        idb.close();
        Log.i("Mahdi", "insertPerson Method");

    }

    public Person getPerson(String gID) {

        Person gPrs = new Person();
        SQLiteDatabase gdb = this.getReadableDatabase();
        String gQuery = "SELECT * FROM " + TableName + " WHERE " + dID + "=" + gID;
        Cursor gCur = gdb.rawQuery(gQuery, null);
        if (gCur.moveToFirst()) {
            gPrs.pName = gCur.getString(1);
            gPrs.pFamily = gCur.getString(2);
        }
        Log.i("Mahdi", "getPerson Method");
        return gPrs;

    }

    public void updatePerson(Person uprs) {

        SQLiteDatabase udb = this.getWritableDatabase();
        ContentValues ucv = new ContentValues();
        ucv.put(dName, uprs.pName);
        ucv.put(dFamily, uprs.pFamily);
        udb.update(TableName, ucv, dID + " = ?", new String[] {String.valueOf(uprs.pID)});
        Log.i("Mahdi", "updatePerson Method");

    }

}
```

:MainActivity

```
package ir.android_studio.database;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RelativeLayout;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    EditText edtName, edtFamily, edtID;
    Button btnInsert, btnView, btnUpdate, btnDelete;
    DatabaseManager dbm;

    @Override
```

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

        edtName = (EditText) findViewById(R.id.edt_name);
        edtFamily = (EditText) findViewById(R.id.edt_family);
        edtID = (EditText) findViewById(R.id.edt_id);
        btnInsert = (Button) findViewById(R.id.btn_insert);
        btnView = (Button) findViewById(R.id.btn_view);
        btnUpdate = (Button) findViewById(R.id.btn_update);
        btnDelete = (Button) findViewById(R.id.btn_delete);
        dbm = new DatabaseManager(this);

        btnInsert.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                Log.i("Mahdi", "insertButton Clicked!");
                String mID = edtID.getText().toString();
                String mName = edtName.getText().toString();
                String mFamily = edtFamily.getText().toString();

                if (TextUtils.isEmpty(mID) || TextUtils.isEmpty(mName) ||
TextUtils.isEmpty(mFamily)) {

                    Toast.makeText(MainActivity.this, "شود تکمیل باید فیلدها تمامی",
Toast.LENGTH_LONG).show();

                }

                else {

                    Person iperson = new Person();
                    iperson.pID = mID;
                    iperson.pName = mName;
                    iperson.pFamily = mFamily;
                    dbm.insertPerson(iperson);
                    Toast.makeText(MainActivity.this, "شد ذخیره موفقیت با اطلاعات",
Toast.LENGTH_LONG).show();
                    Log.i("Mahdi", "Data inserted!");

                }
            }
        });

        btnView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                Log.i("Mahdi", "View Button Clicked!");
                String vID = edtID.getText().toString();
                Person vPer = dbm.getPerson(vID);
                edtName.setText(vPer.pName);
                edtFamily.setText(vPer.pFamily);
                Log.i("Mahdi", "Data Viewed!");

            }
        });

        btnUpdate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                String uID = edtID.getText().toString();
                String uName = edtName.getText().toString();
                String uFamily = edtFamily.getText().toString();

                Person uperson = new Person();
```

```
                uperson.pID = uID;
                uperson.pName = uName;
                uperson.pFamily = uFamily;
                dbm.updatePerson(uperson);

            }
        });

        btnDelete.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                String delID = edtID.getText().toString();
                Person ddprs

            }
        });

    }
}
```

در نهایت به دکمه حذف می رسیم. متد این عمل را با نام deletePerson و از جنس boolean ایجاد می سازم تا اگر شناسه ای که کاربر وارد کرده در دیتابیس موجود نباشد، برنامه کرش نکند و هم اینکه پیغامی روی صفحه ظاهر گردد.

```
public boolean deletePerson(Person dprs) {

    SQLiteDatabase ddb = this.getWritableDatabase();
    long dResult = ddb.delete(TableName, dID + "=?", new String[] {String.valueOf(dprs.pID)});

    Log.i("Mahdi", "deletePerson Method");

    if (dResult == 0)
        return false;
    else
        return true;

}
```

متغیری با نام dResult از جنس long ایجاد کردم (زیرا متد delete مقدار long برمیگرداند). در مقابل توسط متد delete عمل حذف سطر از دیتابیس انجام می شود. این متد ۳ پارامتر می گیرد که مشابه متد update بوده با این تفاوت که در اینجا ContentValues نقشی ندارد. در ادامه اگر dResult مقدار صفر برگرداند یعنی عمل حذف انجام نشده و در غیر اینصورت عمل حذف بدرستی انجام پذیرفته است.

```
if (dResult == 0)
    return false;
else
    return true;
```

در مواقعی که شروط مختصری مانند return داریم، لزومی به استفاده از آکولاد (مانند زیر) نیست و با حذف آن، کد کوتاهتری خواهیم داشت.

```
if (dResult == 0)
{
    return false;
}
else
{
    return true;
}
```

متد دکمه حذف در اکتیویتی را به اینصورت نوشتم:

```
btnDelete.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        String delID = edtID.getText().toString();
        Person dperson = new Person();
        dperson.pID = delID;
        boolean del = dbm.deletePerson(dperson);

        if (del == true) {
            Toast.makeText(MainActivity.this, "اطلاعات حذف شد", Toast.LENGTH_LONG).show();
        }
        else {
            Toast.makeText(MainActivity.this, "در حذف اطلاعات مشکلی وجود دارد",
Toast.LENGTH_LONG).show();
        }

    }
});
```

به تصویر دقت کنید:

در قسمتی که del را برابر true تعیین کردم، رنگ پس زمینه آن متفاوت است که با بردن نشانگر روی آن، نکته ای را یاداوری می کند به این مضمون که نیازی به ذکر true نیست و del با del==true مفهوم یکسانی دارد. من هم به این توصیه عمل می کنم!

حالا پروژه را اجرا می کنم. شناسه ۱۱ که قبلا ذخیره شده بود را وارد و روی دکمه حذف کلیک می کنم:

پیغام حذف اطلاعات روی صفحه ظاهر شد یعنی برنامه مشکلی ندارد. اگر می خواهید مطمئن شوید این شناسه حذف شده دکمه نمایش را بزنید. اگر مشخصات شخص نمایش داده نشد یعنی حذف شده.

آیتم هایی که ابتدای کار مدنظر داشتیم با موفقیت به پروژه اضافه شد. حالا احساس می کنم پروژه من یک آیتم ضروری دیگر هم نیاز دارد. اینکه کاربر (مدیر واحد تولیدی) تعداد پرسنل ثبت شده را مشاهده کند. برای این کار کافیست تعداد سطرهای دیتابیس را بدست بیاوریم. خروجی مدنظر یک عدد صحیح است پس متد را از جنس int و نام personCount در کلاس دیتابیس می سازم:

```
public int personCount() {

    String gQuery = "SELECT * FROM " + TableName;
    SQLiteDatabase gdb = this.getReadableDatabase();
    Cursor gCur = gdb.rawQuery(gQuery, null);
    int cResult = gCur.getCount();
    return cResult;

}
```

در خط اول که مربوط به کوئری SELECT می باشد، شرطی تعیین نشده زیرا قصد داریم تمامی سطرهای جدول محاسبه شود. خط ۲ و ۳ نیاز به توضیح ندارد. در خط چهارم توسط getCount که از متدهای کلاس Cursor می باشد، حاصلی که درون gCur ریخته شده (تعداد سطرها) را درون متغیر cResult میریزد. در نهایت cResult را return می کنم.

ابتدا یک TextView به اکتیویتی اضافه می کنم. البته چون این ویجت فقط عدد را نشان می دهد، یک TextView دیگر نیز کنار آن قرار می دهم تا عبارت "تعداد پرسنل:" را نمایش دهد. برای ویجت نمایش عدد، آی دی txt_count در نظر گرفته و آنرا با نام txtCount در کلاس اکتیویتی مقداردهی می کنم. حالا کافیست داخل متد onCreate کلاس MainActivity عدد حاصل را از personCount گرفته و درون txtCount بریزم:

```
int sCount = dbm.personCount();
txtCount.setText(Integer.toString(sCount));
```

در کد بالا من خروجی personCount را داخل متغیر sCount ریخته و سپس توسط متد setText به ویجت هدف انتقال داده ام. به این ترتیب هربار که اپلیکیشن اجرا شود، تعداد پرسنلی که درون دیتابیس ذخیره شده نشان داده می شود. اما هنوز یک مشکل باقی مانده. کد مربوط به دریافت تعداد پرسنل از دیتابیس و نمایش آن در رابط کاربری فقط یکبار و هنگام اجرای اپلیکیشن اجرا می شود و کاربر اگر اطلاعات جدیدی ثبت و یا اطلاعات قبلی را حذف نماید، این عدد بروز نمی شود (مگر آنکه اپلیکیشن را بسته و مجدد اجرا کند). پس لازم است این کد در متدهای مربوط به دکمه های ثبت و حذف هم اضافه شود. اما راه بهینه تر این است که درون اکتیویتی یک متد جدید در کنار متد onCreate اضافه شود که شامل این کد بوده و در نهایت تنها لازم است متد یکبار درون بدنه اصلی onCreate و یکبار هم در متدهای دکمه های ذخیره و حذف فراخوانی شود. یک متد از نوع void و نام setCount ایجاد کردم:

```
public void setCount() {

    int sCount = dbm.personCount();
    txtCount.setText(Integer.toString(sCount));

}
```

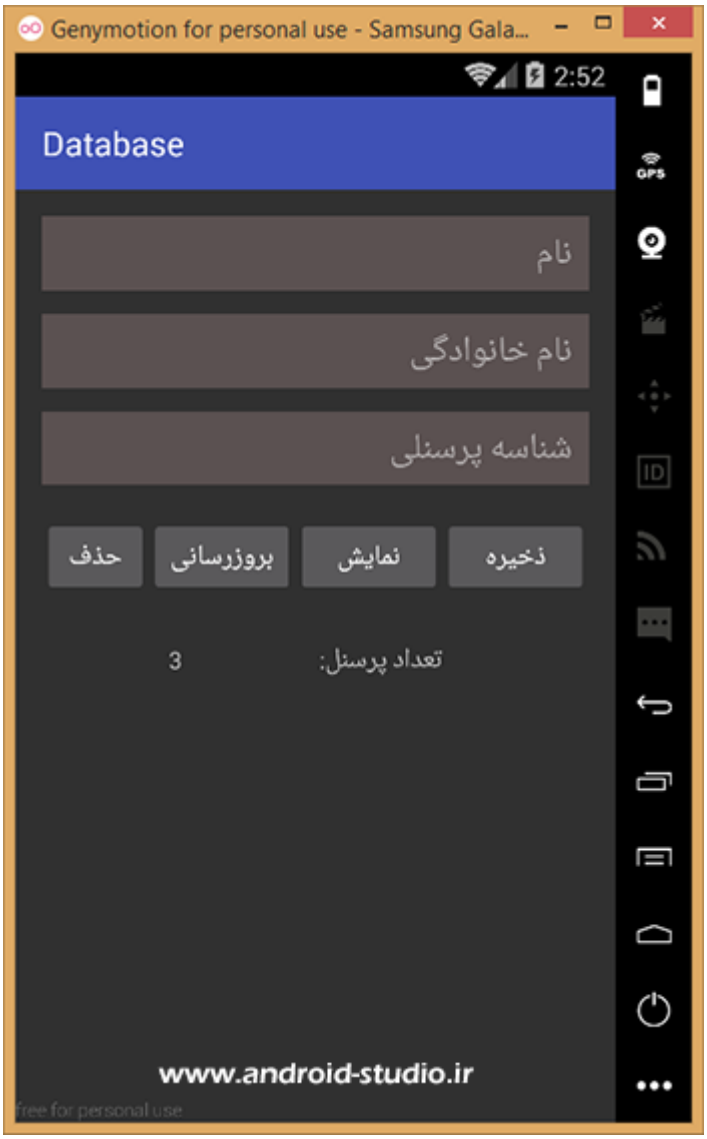
و در نهایت:

```
package ir.android_studio.database;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    EditText edtName, edtFamily, edtID;
    Button btnInsert, btnView, btnUpdate, btnDelete;
    TextView txtCount;
    DatabaseManager dbm;

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

        edtName = (EditText) findViewById(R.id.edt_name);
        edtFamily = (EditText) findViewById(R.id.edt_family);
        edtID = (EditText) findViewById(R.id.edt_id);
        btnInsert = (Button) findViewById(R.id.btn_insert);
        btnView = (Button) findViewById(R.id.btn_view);
        btnUpdate = (Button) findViewById(R.id.btn_update);
        btnDelete = (Button) findViewById(R.id.btn_delete);
        txtCount = (TextView) findViewById(R.id.txt_count);
        dbm = new DatabaseManager(this);
        setCount();

        btnInsert.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                Log.i("Mahdi", "insertButton Clicked!");
                String mID = edtID.getText().toString();
                String mName = edtName.getText().toString();
                String mFamily = edtFamily.getText().toString();

                if (TextUtils.isEmpty(mID) || TextUtils.isEmpty(mName) ||
TextUtils.isEmpty(mFamily)) {

                    Toast.makeText(MainActivity.this, "شود تکمیل باید فیلدها تمامی",
Toast.LENGTH_LONG).show();

                }

                else {
```

```
                    Person iperson = new Person();
                    iperson.pID = mID;
                    iperson.pName = mName;
                    iperson.pFamily = mFamily;
                    dbm.insertPerson(iperson);
                    Toast.makeText(MainActivity.this, "شد ذخیره موفقیت با اطلاعات",
Toast.LENGTH_LONG).show();
                    setCount();
                    Log.i("Mahdi", "Data inserted!");

                }
            }
        });

        btnView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                Log.i("Mahdi", "View Button Clicked!");
                String vID = edtID.getText().toString();
                Person vPer = dbm.getPerson(vID);
                edtName.setText(vPer.pName);
                edtFamily.setText(vPer.pFamily);
                Log.i("Mahdi", "Data Viewed!");

            }
        });

        btnUpdate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                String uID = edtID.getText().toString();
                String uName = edtName.getText().toString();
                String uFamily = edtFamily.getText().toString();

                Person uperson = new Person();
                uperson.pID = uID;
                uperson.pName = uName;
                uperson.pFamily = uFamily;
                dbm.updatePerson(uperson);

            }
        });

        btnDelete.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                String delID = edtID.getText().toString();
                Person dperson = new Person();
                dperson.pID = delID;
                boolean del = dbm.deletePerson(dperson);

                if (del) {
                    Toast.makeText(MainActivity.this, "شد حذف اطلاعات",
Toast.LENGTH_LONG).show();
                    setCount();
                }
                else {
                    Toast.makeText(MainActivity.this, "دارد وجود اشکالی اطلاعات حذف در",
Toast.LENGTH_LONG).show();
                }

            }
        });

    }
```

```
    public void setCount() {

        int sCount = dbm.personCount();
        txtCount.setText(Integer.toString(sCount));

    }

}
```

پروژه را اجرا می کنم:



در ابتدای اجرا، تعداد پرسنل به درستی نمایش داده شد و با هربار ثبت پرسنل جدید یا حذف یک پرسنل، عدد نیز کم یا زیاد شد.

توجه : سورس پروژه درون پوشه Exercises قرار داده شده است.

با ارائه انتقادات و پیشنهادات خود، ما را در ارائه آموزش های بهتر یاری فرمائید.

با توجه به اینکه آموزشهای پایه با قیمت ناچیزی برای تامین هزینه های سایت و تهیه آموزش های بعدی عرضه می شود، به اشتراک گذاری این فایل با دیگران خلاف اخلاق خواهد بود.

www.android-studio.ir