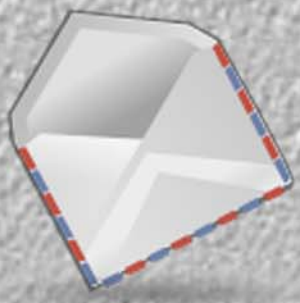
کتاب آموزشی

SQLite

پایگاه اطلاع رسانی تربیت معلّم

www.mttc88.blogfa.com



mttc88@gmail.com

فهرست

مقدمه

SQL مخفف عبارت Structured Query Language یک زبان توسعه یافته برای ارتباط با پایگاه داده است. درک دستورات این زبان به خاطر نزدیکی زیاد آنها به زبان محاوره‌ای انگلیسی خیلی ساده است. به طوری که حتی اگر با قواعد آن هم آشنا نباشید از طریق ترجمه دستور می توانید تا حدود زیادی خروجی دستور را درک کنید.

به عنوان مثال:

```
SELECT * FROM mybook WHERE name="Ali";
```

ترجمه لغت به لغت جمله بالا میشود : انتخاب کن هر چیزی را از جدول mybook جایی که نام = Ali باشد. این جمله یک دستور به زبان SQL هست که به برنامه می گوید در فیلد نام جدول mybook پایگاه داده مورد نظر بگردد و رکوردی را که موجودی فیلد نام آن با عبارت Ali برابر بود را برگرداند.

فیلد چیست؟ رکورد چیست؟ منظور از پایگاه داده و جدول چیست؟

فرض کنید تعداد زیادی کتاب در کتابخانه شخصی ما وجود دارد و قصد داریم آنها را مرتب و طبقه‌بندی کنیم. برای شروع ما بخشی از مشخصات یک کتاب را در جایی یادداشت میکنیم و بعد سعی میکنیم یادداشت‌ها را مرتب کنیم تا در نهایت کتابخانه ما مرتب شود. چه بخشی از مشخصات کتاب را برای مرتب‌سازی لازم داریم؟ بعنوان مثال: شماره کتاب - نام کتاب – نام نویسنده – موضوع کتاب وقتی در مورد مرحله قبل به نتیجه قطعی رسیدیم یک جدول رسم می‌کنیم و برای جدول تعدادی ستون تعیین میکنیم.

مثلا ستون اول شماره کتاب – ستون دوم نام کتاب – ستون سوم نام نویسنده و ستون چهارم موضوع کتاب

شماره کتاب	نام کتاب	نام نویسنده	موضوع کتاب

بعد سطر به سطر اطلاعات موردنظر کتابها را در ستون‌های جدول وارد میکنیم. حالا فرض کنید ما قصد داریم بعضی از کتابها را به دیگران امانت بدهیم. برای اینکه بدانیم کتابها را به چه کسی و با چه شرایطی امانت دادیم جدول دیگری شبیه جدول کتابها رسم میکنیم و در ستون‌ها، مشخصات امانت گیرنده و موضوع را یادداشت میکنیم.

مثلا ستون اول شماره کتاب – ستون دوم نام کتاب – ستون سوم نام امانت گیرنده – ستون چهارم تاریخ بازگشت – ستون پنجم آدرس یا شماره تماس امانت گیرنده و ...

شماره کتاب	نام کتاب	نام امانت گیرنده	تاریخ بازگشت	آدرس و شماره تلفن امانت گیرنده

و بعد سطر به سطر مشخصات هر کتابی که به امانت داده می شود را یادداشت میکنیم.

حالا ما دوتا جدول داریم که ارتباط مستقیمی با همدیگر و موجودی کتابهای ما دارن. بر اساس این جداول ما می توانیم کتابها را طبقه‌بندی کنیم و از موجودی و کسری کتابها اطلاع کامل داشته باشیم.

ممکن است برای اینکه بخواهیم طبقه‌بندی بهتری انجام بدهیم جداول دیگری را هم طراحی و اطلاعاتی را درج کنیم.

در پایان این جداول را در یک پرونده قرار میدهیم و آن پرونده را در محل امنی نگهداری می‌کنیم. حالا یک پایگاه داده (بانک اطلاعاتی)داریم .

پرونده، بانک اطلاعات (**database** – پایگاه داده) ماست. جداول داخل پرونده، جداول (**table**) بانک اطلاعاتی ما هستند. ستونهای هر جدول، فیلدها (**field**) و سطرهای هر جدول که حاوی اطلاعات بخصوصی راجع به یک کتاب یا یک امانت گیرنده کتاب هستند هم رکوردهای (**record**) بانک اطلاعاتی ما را تشکیل میدهند .

با توجه به این توضیحات یکبار دیگر به این جمله نگاه کنید:

```
SELECT * FROM mybook WHERE name="Ali";
```

آیا می توانید منظور این دستور را به طور کامل درک کنید؟

سایر دستورات **SQL** هم به همین سادگی قابل درک و اجرا هستند. و البته سعی بر این هست که کاربردی‌ترین و اصلی‌ترین دستورات **SQL** برای برقراری ارتباط با پایگاه داده و پلاگین **SQLite** و روش استفاده از آن مطرح شود.

برای شروع کار با پایگاه‌های داده ابتدا باید یک دیتابیس ایجاد کرد. در مرحله بعد باید درون این دیتابیس خالی جداول مورد نظر را ایجاد کرد و برای هر جدول فیلدهای آن را تعریف کرد. هر فیلد خصوصیاتی دارد که بر اساس این خصوصیات قادر هست اطلاعات بخصوصی بگیرد. مثلا : اگر خصوصیت یک فیلد عددی تعیین شده باشد نمی شود داخل این فیلد یک عبارت متنی قرارداد.

حالا می شود از این پایگاه داده استفاده کرد. به این معنی که می شود اطلاعات مورد نظر را به صورت رکورد به جداول اضافه کرد. می شود رکوردهای ثبت شده را ویرایش یا حذف کرد ، در اطلاعات ثبت شده جستجو کرد و گزارش‌های لازم را دریافت کرد و ... هر کدام از عملیات بالا برای اجرا دستوراتی دارند که در ادامه آشنا خواهیم شد.

ساخت دیتابیس (Database)

در SQL برای ایجاد یک دیتابیس از دستور CREATE DATABASE استفاده می شود. شکل کلی این دستور به این صورت هست.

CREATE DATABASE name;

در قسمت name می توا نید نامی را برای دیتابیس تعیین کنید. نام دیتابیس باید یک عبارت بهم پیوسته باشد اما می تواند دارای زیر خط (_) هم باشد . مثلا : librarydatabase یا مثلا : my_library

کاراکتر (;) در انتهای جمله نشان دهنده پایان دستور است. گاهی اوقات ممکن است دستورات را در چند خط بنویسید اما تا زمانی که از کاراکتر (;) در انتهای دستور استفاده نکرده باشید تمامی عبارت‌های این چند خط جزو یک دستور محسوب خواهند شد

با دستور بالا یک دیتابیس خالی ایجاد شده و حالا می شود جداول را درون دیتابیس ساخت. برای ساخت جداول باید دیتابیس مورد نظر را اجرا و بارگذاری کرد.

بارگذاری دیتابیس (Datebase)

دستور USE برای بارگذاری دیتابیس استفاده می شود. شکل کلی این دستور به این صورت هست.

USE name;

در قسمت name باید نام پایگاه داده ی خود را وارد کنید.

برای حذف پایگاه داده هم می توانید از دستور DROP DATABASE استفاده کنید. شکل کلی این دستور به این صورت هست.

DROP DATABASE name;

در قسمت name باید نام دیتابیس خود را قرار بدید. دقت کنید که با حذف پایگاه داده تمامی جداول و اطلاعات درون آن حذف و غیر قابل بازگشت خواهند بود. پس در استفاده از این دستور دقت کنید. در مرحله بعد از ساخت دیتابیس لازم است که جداول مورد نظر را ایجاد کنید.

ساخت جدول در Datebase

دستور CREATE TABLE برای این منظور استفاده می شود. استفاده از این دستور نیازمند داشتن پارامترهایی هست. وقتی شما جدولی را ایجاد میکنید باید تعیین کنید که این جدول چه فیلدهایی دارد و هر فیلد چه خصوصیتی دارد. هر دیتابیس باید حداقل یک جدول و هر جدول باید حداقل یک فیلد داشته باشه و در هر جدول باید حداقل یک فیلد آن خصوصیت کلید اصلی (Primary Key) داشته باشه. هیچ دو رکوردی در جدول نمی تواند وجود داشته باشد که داده های کاملا یکسان با هم داشته باشند . بنابراین برای جلوگیری از این خطا که اطلاعات کاملا یکسان را دوبار در جدول وارد نکنیم باید یک شناسه خاص برای یکی از فیلدهای جدول تعیین کنیم. این شناسه خاص همان کلید اصلی است. فیلدی که خصوصیت کلید اصلی براش تعیین شده نمیتواند دو تا مقدار مشابه به هم داشته باشد .

مثلا اگر شما در جدول، فیلدی به نام ردیف داشته باشید و برای ردیف خصوصیت کلید اصلی را تعیین کرده باشید نمی توانید دو تا رکورد را در جدول وارد کنید که در فیلد ردیف عدد یکسان داشته باشن. این مثال را ببینید:

ردیف	موضوع	تاریخ
۱		
۲		
۳		
۳		

در این مثال در فیلد ردیف دوبار عدد ۳ درج شده و این یعنی خطا. می دانید علت خطا چیست؟

شکل کلی دستور CREATE TABLE به این صورت هست:

```
CREATE TABLE name (Field1 datatype, Field2 datatype, Field3 datatype );
```

در دستور بالا در قسمت name نام جدول مورد نظر را می‌نویسیم. بعد کاراکتر پرانتز را باز میکنیم تا فیلدهای جدول را همراه با خصوصیاتش بنویسیم. در خطوط بعدی نام فیلد مورد نظر و خصوصیت آن را تعیین می کنیم و در پایان هر فیلد کاراکتر ویرگول (,) را درج می‌کنیم. این کاراکتر نشان دهنده پایان یک فیلد و خصوصیات آن هست. در پایان، پرانتز را بسته و کاراکتر (;) به معنی پایان دستور درج می کنیم.
با ارسال این دستور درون دیتابیس شما یک جدول با فیلدهای مورد نظر ایجاد خواهد شد.
موقع ایجاد جداول باید نام و نوع فیلدها را تعیین کرد.

انواع داده در SQL

به طور کلی سه نوع داده اصلی در SQL وجود دارد: عددی، تاریخ و زمان و رشته‌ای. در نوع داده عددی (INTEGER – INT – FLOAT – ...) می شود اعداد صحیح یا اعشاری را وارد کرد. در نوع داده تاریخ و زمان (DATE – TIME – ...) می شود انواع مختلف تاریخ و زمان را وارد کرد.

دو نوع پرکاربرد : ۱- داده int شامل اعداد صحیح منفی و مثبت بدون اعشار - ۲- داده float اعداد اعشاری

در نوع داده رشته‌ای (CHAR – VARCHAR – BLOB – ...) می شود انواع داده های متنی با طول کاراکترهای ثابت و متغیر را وارد کرد.
بعضی کلمات کلیدی هم وجود دارد که موقع ایجاد یک فیلد استفاده می شود و خصوصیتی را برای یک فیلد تعیین میکند :

موقع استفاده از NOT NULL همه رکوردهای جدول در فیلدی که این خصوصیت تعیین شده باشد باید مقداری را بگیرند.
(NOT NULL به این معنی که فیلدی که این خصوصیت را داشته باشد نباید خالی باشد)
عبارت PRIMARY KEY پس از نام یک فیلد بیانگر این است که این فیلد کلید اصلی جدول ما است.

```
CREATE TABLE mybook ( id INTEGER PRIMARY KEY NOT NULL, name CHAR NOT NULL, subject CHAR NOT NULL, );
```

در کد بالا جدولی به نام mybook ایجاد خواهد شد با ۳ فیلد. که شماره کتاب، نام کتاب و موضوع کتاب را ثبت میکند .
فیلد اول به نام id که نوع داده عددی را در خودش ذخیره میکند و همچنین کلید اصلی جدول (PRIMARY) هم هست.
فیلد دوم به نام name که نوع داده متنی را در خودش ذخیره میکند .
و فیلد سوم به نام subject که باز نوع داده متنی را در خودش ذخیره میکند .
دقت کنید که همه این فیلدها خاصیت NULL NOT دارن و به این معناست که در این فیلدها باید مقداری درج شود.
در اینجا سعی شده کلمات کلیدی و انواع داده پرکاربرد در SQL مطرح شود،

درج اطلاعات در جدول

تا اینجا یادگرفتیم چطور یک دیتابیس بسازیم. چطور یک دیتابیس را بارگذاری کنیم و چطور در یک دیتابیس جدول مورد نظر با فیلدهای مشخصی را ایجاد کنیم. در مرحله بعد نوبت به ورود اطلاعات به جدول میرسد .
اطلاعات ما با استفاده از دستور INSERT INTO به صورت رکورد به جدول اضافه خواهند شد
مثال قبلی را در نظر بگیرید. جدولی به نام mybook که سه تا فیلد id و name و subject داشت.
با دستور INSERT INTO می شود اطلاعات مربوط به یک کتاب را به صورت یک رکورد در جدول مورد نظر ثبت کرد.
شکل کلی این دستور به این صورت هست:

```
INSERT INTO name (Field 1, Field 2 , Field 3, ...) VALUES (value 1, value 2, value 3, ...);
```

دستور بالا در قسمت name نام جدول مورد نظر را مینویسیم. در قسمت Field 1 , 2 , ۳ نام فیلدهای جدول و در قسمت value 1 , 2 , 3 مقادیر فیلدها را به ترتیب مینویسیم. در دستور بالا مقدار value 1 به Field 1 ، مقدار value 2 به Field 2 و ... نسبت داده می شود.
همچنین برای جداسازی فیلدها و مقادیر آنها از کاراکتر ویرگول (,) بعد از هر نام فیلد و مقدار فیلد استفاده شده. و در پایان دستور از کاراکتر (;) استفاده می شود.

مثال:

```
INSERT INTO mybook (id, name , subject) VALUES (1,"sql","learning sql" );
```

با دستور بالا یک رکورد به جدول **mybook** اضافه می شود که در فیلد **id** عدد ۱ ، در فیلد **name** که نام کتاب هست عبارت **sql** و در فیلد **subject** که موضوع کتاب هست عبارت **learning sql** درج خواهد شد.

id	name	subject
۱	sql	learning sql

به همین ترتیب میتوانیم رکوردهای دیگری را به جدول مورد نظر اضافه کنیم.

در دستور بالا دقت کنید که عبارت‌های رشته‌ای (متنی) درون یک جفت دابل کوتیشن (" ") قرار گرفتن مثل "sql" و "learning sql" برای ثبت رکورد در جدول **mybook** به این صورت هم می شود عمل کرد :

```
INSERT INTO mybook SET id = 1 , name = "sql" , subject = "learning sql" ;
```

جستجو و بازیابی داده ها در جدول

فرض کنید جدولی به نام **tell** با ۳ فیلد (**name**، **id** و **phone**) داریم و تعدادی رکورد در این جدول ثبت شده است.

id	name	phone
۱	Ali	۰۹۱۵۳۵۴۵۱۲۷۲۱
۲	Reza	۰۹۴۳۴۴۲۱۳۵۷۷۸
۳	Mohammad	۰۴۵۳۱۴۶۷۸۴۲۳
۴	Amirali	۰۹۴۵۳۵۴۵۸۴۵۳۳
۵	Komeyl	۰۶۵۴۳۱۵۴۸۲۱۲

قصد داریم در این جدول به دنبال رکوردی بگردیم و نتیجه را نمایش بدهیم. دستور **SELECT** برای این منظور استفاده می شود.
شکل کلی دستور به اینصورت هست:

```
SELECT (field name/s) FROM tablename;
```

در قسمت (**field name/s**) نام فیلد (فیلدهای) مورد نظر و در **tablename** نام جدول مورد نظر را وارد میکنیم.

مثال بالا را در نظر بگیرید :

```
SELECT name FROM  tell;
```

نتیجه دستور بالا انتخاب تمامی رکوردهای فیلد **name** جدول **tell** خواهد بود.

id	name	phone
۱	Ali	۰۹۱۵۳۵۴۵۱۲۷۲۱
۲	Reza	۰۹۴۳۴۴۲۱۳۵۷۷۸
۳	Mohammad	۰۴۵۳۱۴۶۷۸۴۲۳
۴	Amirali	۰۹۴۵۳۵۴۵۸۴۵۳۳
۵	Komeyl	۰۶۵۴۳۱۵۴۸۲۱۲

SELECT name,phone FROM tell;

نتیجه این دستور انتخاب تمامی رکوردهای فیلد **name** و فیلد **phone** جدول **tell** خواهد بود.

id	name	phone
۱	Ali	۰۹۱۵۳۵۴۵۱۲۷۲۱
۲	Reza	۰۹۴۳۴۴۲۱۳۵۷۷۸
۳	Mohammad	۰۴۵۳۱۴۶۷۸۴۲۳
۴	Amirali	۰۹۴۵۳۵۴۵۸۴۵۳۳
۵	Komeyl	۰۶۵۴۳۱۵۴۸۲۱۲

در دستور **SELECT** اگر به جای نام فیلد مورد نظر از کاراکتر (*) استفاده شود یعنی:

SELECT * FROM tablename;

تمامی فیلدهای رکوردهای جدول بدست می آید.

id	name	phone
۱	Ali	۰۹۱۵۳۵۴۵۱۲۷۲۱
۲	Reza	۰۹۴۳۴۴۲۱۳۵۷۷۸
۳	Mohammad	۰۴۵۳۱۴۶۷۸۴۲۳
۴	Amirali	۰۹۴۵۳۵۴۵۸۴۵۳۳
۵	Komeyl	۰۶۵۴۳۱۵۴۸۲۱۲

دستور **SELECT** را می شود به کمک شرط محدود به یک موقعیت خاص کرد و نتیجه دلخواه را بدست آورد. برای ایجاد شرط در دستور **SELECT** از عبارت **WHERE** استفاده می شود:

SELECT * FROM tell WHERE name="Reza";

در دستور بالا رکوردی که مقدار فیلد **name** آن برابر با عبارت **Reza** باشه (یعنی رکورد شماره ۲) بدست می آید .

id	name	phone
۱	Ali	۰۹۱۵۳۵۴۵۱۲۷۲۱
۲	Reza	۰۹۴۳۴۴۲۱۳۵۷۷۸
۳	Mohammad	۰۴۵۳۱۴۶۷۸۴۲۳
۴	Amirali	۰۹۴۵۳۵۴۵۸۴۵۳۳
۵	Komeyl	۰۶۵۴۳۱۵۴۸۲۱۲

یک مثال دیگر :

SELECT name,phone FROM tell WHERE phone = 065431548212;

در دستور بالا شرط (**phone = 065431548212**) بررسی و فقط مقدار فیلد **name** و **phone** از جدول **tell** که برابر شرط است (یعنی فیلد **name** و **phone** رکورد شماره ۵) بدست می آید .

id	name	phone
۱	Ali	۰۹۱۵۳۵۴۵۱۲۷۲۱
۲	Reza	۰۹۴۳۴۴۲۱۳۵۷۷۸
۳	Mohammad	۰۴۵۳۱۴۶۷۸۴۲۳
۴	Amirali	۰۹۴۵۳۵۴۵۸۴۵۳۳
۵	Komeyl	۰۶۵۴۳۱۵۴۸۲۱۲

در قسمت شرط جستجو می شود از عملگرهای = ، > ، < ، >= ، <= یا <> استفاده کرد.
برای تعمیم بازه جستجو در دستور **SELECT** از کلمات کلیدی مانند **NOT – OR – AND** استفاده کرد.

```
SELECT * FROM tell WHERE name="Ali" OR id=2;
```

در این دستور رکوردی که فیلد **name** برابر عبارت **Ali** باشه یا فیلد **id** برابر عدد ۲ باشه بدست می آید.

id	name	phone
۱	Ali	۰۹۱۵۳۵۴۵۱۲۷۲۱
۲	Reza	۰۹۴۳۴۴۲۱۳۵۷۷۸
۳	Mohammad	۰۴۵۳۱۴۶۷۸۴۲۳
۴	Amirali	۰۹۴۵۳۵۴۵۸۴۵۳۳
۵	Komeyl	۰۶۵۴۳۱۵۴۸۲۱۲

برای جستجو کلمات مشابه در شرط می شود از **LIKE** استفاده کرد.
به این مثال توجه کنید:

```
SELECT * FROM tell WHERE name LIKE "Ali";
```

در این دستور رکوردی که فیلد **name** آن شبیه به عبارت **Ali** باشه بدست می آید.
حالا این مثال را ببینید:

```
SELECT * FROM tell WHERE name LIKE "%Ali";
```

در این دستور قبل از عبارت **Ali** یک کاراکتر (٪) اضافه شده. وجود این کاراکتر در شرط باعث می شود که بازه جستجو گسترش پیدا کنه. به این معنی که هر رکوردی که در جدول آخرش به **Ali** ختم می شود انتخاب می شود. خروجی این دستور دوتا رکورد خواهد بود.
رکورد شماره ۱ و رکورد شماره ۴

id	name	phone
۱	Ali	۰۹۱۵۳۵۴۵۱۲۷۲۱
۲	Reza	۰۹۴۳۴۴۲۱۳۵۷۷۸
۳	Mohammad	۰۴۵۳۱۴۶۷۸۴۲۳
۴	Amirali	۰۹۴۵۳۵۴۵۸۴۵۳۳
۵	Komeyl	۰۶۵۴۳۱۵۴۸۲۱۲

از کاراکتر (٪) قبل یا بعد از عبارت مورد نظر برای شرط استفاده می شود تا به این وسیله کلمات مشابه هم انتخاب شود.
به این مثال دقت کنید:

```
SELECT * FROM tell WHERE name LIKE "%m%";
```

خروجی این دستور سه تا رکورد خواهد بود. رکورد شماره ۳ و ۴ و ۵ . در فیلد **name** این سه تا رکورد کاراکتر **m** وجود دارد. – **Mohammad Komeyl – Amirali**

id	name	phone
۱	Ali	۰۹۱۵۳۵۴۵۱۲۷۲۱
۲	Reza	۰۹۴۳۴۴۲۱۳۵۷۷۸
۳	Mohammad	۰۴۵۳۱۴۶۷۸۴۲۳
۴	Amirali	۰۹۴۵۳۵۴۵۸۴۵۳۳
۵	Komeyl	۰۶۵۴۳۱۵۴۸۲۱۲

ویرایش و بروزرسانی داده‌های پایگاه داده

ممکن است که قصد داشته باشید اطلاعاتی را در یک رکورد ویرایش یا اطلاعات جدیدتری را جایگزین کنید. مثال قبلی را در نظر بگیرید. فرض کنید شماره تلفن یکی از افرادی که در جدول **tell** درج شده اشتباه هست و باید تغییر کند.

در این مواقع از دستور **UPDATE** استفاده می شود.

شکل کلی این دستور به این صورت هست:

```
UPDATE  tablename  SET  Field 1 = value 1, Field 2 = value 2, Field 3 = value 3, ... WHERE your condition ;
```

در دستور بالا به جای **tablename** نام جدول مورد نظر، به جای **Field 1 – 2 – 3** و ... نام فیلدهای جدول، به جای **value 1 – 2 – 3** و ... مقادیر مورد نظر و در قسمت **condition your** شرط مورد نظر را برای دستور مینویسیم.

مثال :

```
UPDATE tell SET
        name = "Alireza"
        phone = 5435732168
WHERE id = 1;
```

قبل از ویرایش

d	nsme	phone
۱	Ali	۰۹۱۵۲۵۱۲۱۲۱
۲	Reza	۰۹۴۳۴۴۲۱۳۵۷۷۸
۳	Mohemmad	۰۶۸۳۱۴۶۷۸۴۲۳
۴	Amirali	۰۹۴۵۳۵۴۵۸۴۵۳۳
۵	Komeyl	۰۶۵۴۳۱۵۴۸۲۱۲

بعد از ویرایش

id	name	phone
۱	Alireza	۵۴۳۵۷۳۲۱۶۸
۲	Reza	۰۹۴۳۴۴۲۱۳۵۷۷۸
۳	Mohammad	۰۶۸۳۱۴۶۷۸۴۲۳
۴	Amiral	۰۹۳۵۲۵۳۵۹۳۵۲۲
۵	Komeyl	۰۶۵۴۳۱۵۴۸۲۱۲

حذف داده‌ها از پایگاه داده

برای حذف یک رکورد از جدول مورد نظر از دستور **DELETE** استفاده می شود.

شکل کلی این دستور به این صورت هست:

```
DELETE FROM tablename WHERE your condition ;
```

در دستور بالا به جای **tablename** نام جدول موردنظر و به جای **your condition** شرط موردنظر را وارد میکنیم.

```
DELETE FROM tell WHERE id = 3;
```

در دستور بالا رکورد شماره ۳ جدول **tell** حذف خواهد شد.

id	name	phone
۱	Alireza	۵۴۳۵۷۳۲۱۶۸
۲	Reza	۰۹۴۳۴۴۲۱۳۵۷۷۸
۴	Amirali	۰۹۴۵۳۵۴۵۸۴۵۳۳
۵	Komeyl	۰۶۵۴۳۱۵۴۸۲۱۲

دستور **DELETE** اگر به این صورت زیر استفاده شود : تمامی رکوردهای جدول حذف خواهند شد.

```
DELETE FROM tablename ;
```

اگر قصد داشته باشید یک جدول را به طور کامل از پایگاه داده حذف کنید از دستور **DROP TABLE** استفاده کنید:

```
DROP TABLE tablename ;
```

فرق دوتا دستور بالا در این هست که اگر با دستور **DELETE FROM tablename** تمام رکوردهای جدول مورد نظر را حذف کنیم. باز هم جدول ما باقی هست و می شود به کمک **INSERT INTO** رکوردهای جدیدی به جدول اضافه کرد.

اما اگر از **DROP TABLE** استفاده شود جدول به طور کامل از پایگاه داده حذف خواهد شد و برای استفاده مجدد باید به کمک **CREATE TABLE** ابتدا جدول جدیدی ایجاد و سپس آن استفاده کرد.

به دلیل غیر بازگشت بودن اطلاعاتی که حذف میشوند در استفاده از این دستورات با دقت عمل کنید.

ایجاد تغییرات در ساختار جدول

ممکن هست قصد داشته باشید ساختار جدولی که قبلا ایجاد کردید را تغییر بدهید. مثلا نام یک فیلد را تغییر بدید یا یک فیلد به جدول اضافه یا حذف کنید. یا مثلا نوع داده یک فیلد را از عددی به متنی تغییر بدید یا کلید اصلی یک جدول را از یک فیلد به فیلد دیگه ای منتقل کنید .

انجام هرگونه تغییر در ساختار جدول به کمک دستور **ALTER TABLE** امکان پذیراست.
شکل کلی این دستور به این صورت هست:

ALTER TABLE tablename alteration ;

در قسمت **tablename** نام جدول موردنظر و در قسمت **alteration** تغییر مورد نظر را درج می کنیم.

دستور **ALTER TABLE** با پارامترهای زیر تغییرات لازم را در ساختار جدول ایجاد میکند .

ADD field name and description [first/after field]

با پارامتر **ADD** می شود در محل مورد نظر فیلد جدیدی به جدول اضافه کرد. به جای عبارت **field name and description** نام و نوع ستون مورد نظر و به جای **first/after field** محل فیلد مورد نظر را وارد کنید.

MODIFY field description

با پارامتر **MODIFY** می شود خصوصیت فیلد مورد نظر در جدول را ویرایش کرد. به جای عبارت **field description** نام فیلد و خصوصیت مورد نظر را وارد کنید.

DROP filed

با پارامتر **DROP** می شود فیلدی را از جدول مورد نظر حذف کرد. به جای **field** نام فیلد مورد نظر را وارد کنید.

پارامترهای دیگری هم در دستور **ALTER TABLE** قابل استفاده هستند که مطالعه بیشتر در این زمینه را به شما واگذار میکنیم.

تا اینجا سعی بر این بوده که با کاربردی‌ترین و اصلی‌ترین دستورات **SQL** آشنا بشیم. البته کار با پایگاه داده مبتنی بر زبان **SQL** جزئیات زیادی دارد که اگر مایل باشید می توانید کتابهایی که در این زمینه وجود دارند را مطالعه کنید.

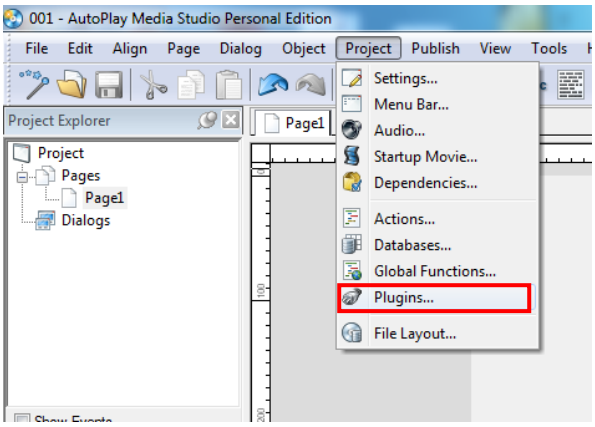
هدف اصلی ما راه اندازی یک پایگاه داده مبتنی بر پلاگین **SQLite** هست. بنابراین باید فرامین مورد نظر برای ساختن ویرایش و ارتباط با یک پایگاه داده را به این پلاگین ارسال و نتایج را دریافت کنیم.

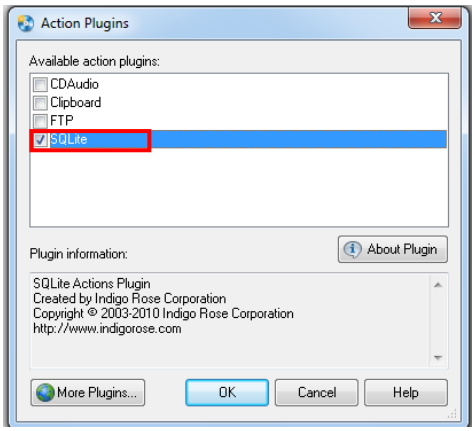
برای اینکار لازم است که دستوراتی که تا الان یادگرفتیم را به کمک توابع رشته ای و دستورات مربوط به پلاگین به **SQLite** ارسال و نتایج بازگشتی را به کمک لیست باکسها، رشته‌ها، آرایه‌ها و... بررسی و نمایش بدهیم.

ساخت برنامه ثبت نام در Autoplay media studio 8.0

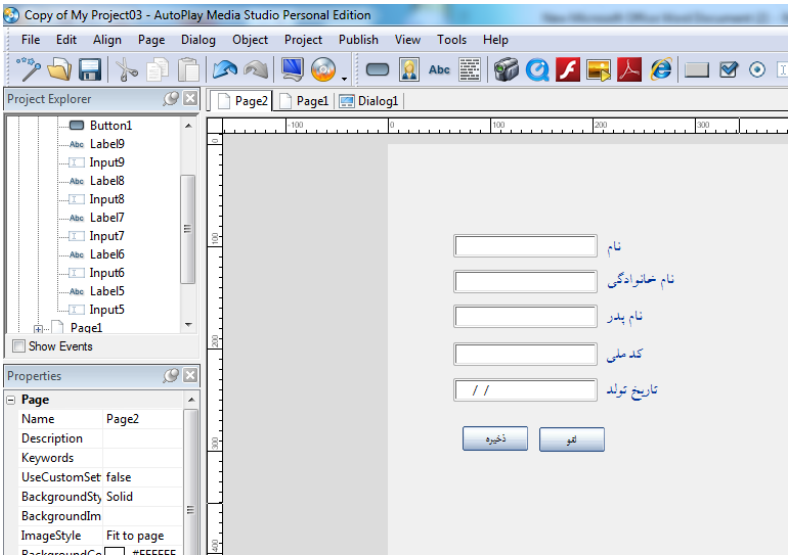
بعد از بررسی ساختار **SQLite** با یک مثال کاربردی قصد داریم دستورات مربوط به پایگاه داده و نحوه نمایش اطلاعات آن را در برنامه Autoplay media studio با ساخت یک برنامه مفید انجام دهیم .

در آغاز فایل جدید ایجاد میکنیم و از منوی **Plugins** < **Project** مورد **SQLite** را فعال میکنیم .

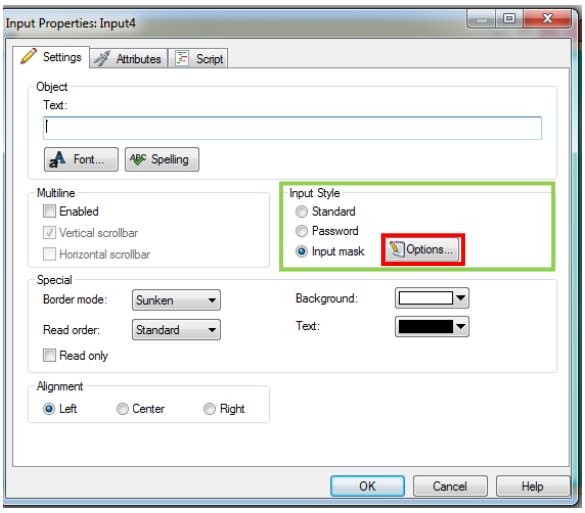


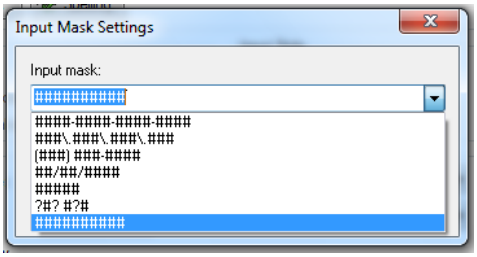


در صفحه برای ورود اطلاعات نیاز به تعدادی **Input** داریم و چند عدد **Lable** نیز در کنار آن قرار میدهیم .



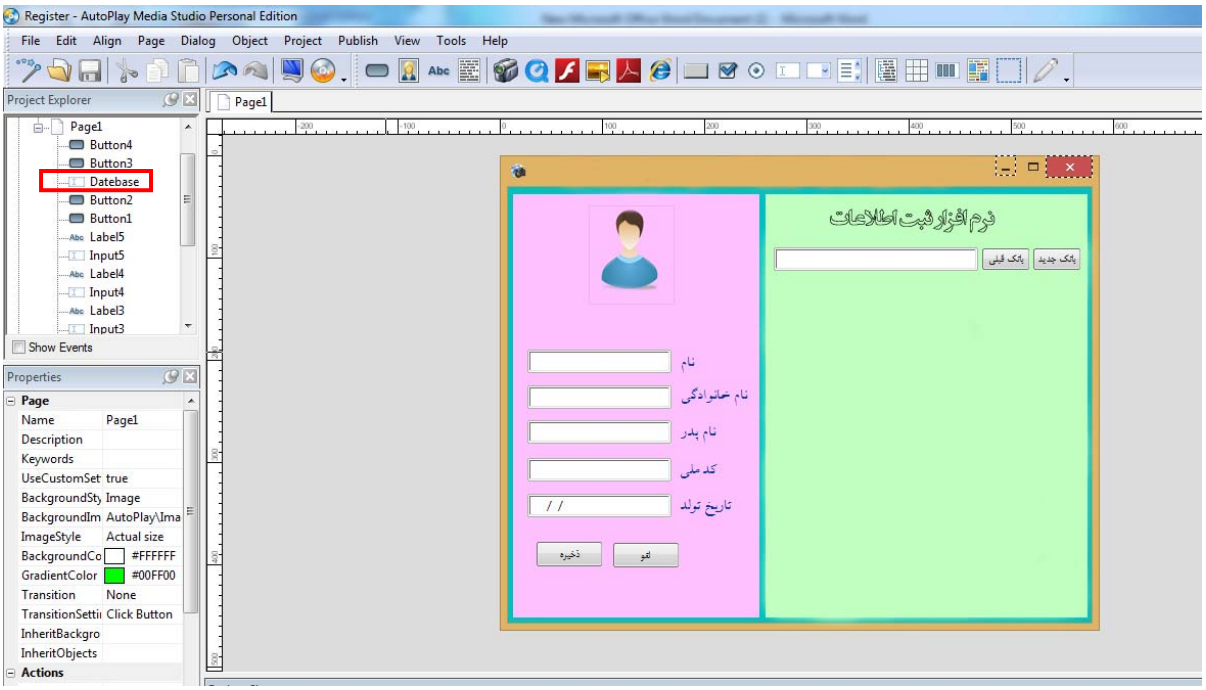
همانطور که در تصویر قابل مشاهده است برای بعضی از اطلاعات نیازمند عدد هستم و باید **Input** را طوری تنظیم کنیم که فقط عدد دریافت کند .
برای این منظور روی **Input** مورد نظر دوبار کلیک می کنیم و در بخش **Setting** گزینه ی **Input mask** را انتخاب میکنیم .
با کلیک روی **Browse** امکان محدود کردن آن را داریم





\# نشان دهنده کارکتر عددی است مثالا برای عدد ۱۰ رقمی باید ۱۰ # در **mask** قرار داده و روی **OK** کلیک کنیم .

برای جلوه ی بهتر تصاویر و اشکال گرافیکی را به برنامه اضافه می کنیم .



برای ایجاد بانک اطلاعاتی و بارگزاری آن در برنامه دو دکمه و یک **input** دیگر به برنامه اضافه کردیم (سمت راست).

از قسمت سمت راست **Project Explorer** نام **Input** ایجاد شده را به **Datebase** تغییر می دهیم .

معرفی توابع

جهت اجرای برنامه نیازمند دستوراتی هستیم که در سراسر برنامه چندین بار مورد استفاده قرار میگرید . برای پرهیز از تکرار چنین دستوراتی که بعضا طولانی هستند باید از مفهوم تابع(**Function**)در این برنامه استفاده کنیم . دستور را یکبار تعریف میکنیم و در پنجره ها و دکمه فقط نام آن را می نویسم .

از منوی **Project** گزینه **Global Function** را انتخاب میکنیم و در کادر ظاهر شده عملیات مربوط به ساخت جدول **SQLite** را به صورت یک تابع تعریف میکنیم .

```
Register={};
function Register.Creat(db)
	db = SQLite.Open(db_Path);
	SQLite.Query(db, "create table Users(userid integer primary key, name text, family text, father text, Code integer,tarikh text)", nil);
	SQLite.Close(db);
end
```

تابعی با نام **Register .Creat** تعریف کردیم :

db_Path : مکان ذخیره بانک اطلاعاتی است که کاربر به برنامه میدهد .

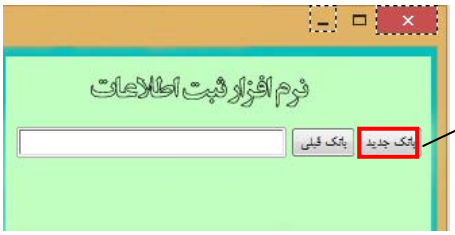
SQLite.Query : دستور ساخت جدول در دیتابیس است که در صفحات قبل توضیح داده شد .

خانه های جدول با رنگ قرمز مشخص شده اند . این نام ها هنگام ذخیره داده در جدول مورد استفاده قرار میگیرند .

SQLite.Close(db) : به طور معمول پس از بازگشایی **SQLite.Open** و انجام عملیات ، در انتها باید ارتباط بسته شود .

در این قسمت **Global Function** را ذخیره می کنیم .

ساخت بانک اطلاعاتی جدید



نوشتن ادامه کدهای دیتابیس را در دکمه های سمت راست ادامه میدهیم .
از این دکمه برای ایجاد یک دیتابیس خالی استفاده میکنیم .

ابتدا در بخش **Add Action** کدها از یک **Dialog.FileBrowse** استفاده میکنیم :

```
db_Path = Dialog.FileBrowse(false, "Open Database", "", "Database Files (*.db)|*.db|", "", "db", false, true)[1];
```

در تابع **Register.Creat** که در **Global Function** تعریف کردیم متغیری به نام **db_Path** وجود داشت کـه ایـن متغیـر در اینجـا توسط **Dialoge** مقدار دهی خواهد شد .

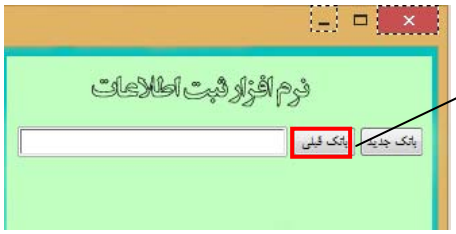
[1] : این مقدار را به صورت دستی اضافه کردیم .... در برخی کدهای باید از این عبارت در انتها استفاده کنیم تا آدرس را برگرداند .

در ادامه ی کد ها باید آدرسی که توسط کاربر تعیین میشود و فایل دیتابیس را در آن ذخیره میکند را در **Input** روبروی آن نمایش می دهیم . برای این مورد از یک عبارت شرطی استفاده می کنیم و آن را در ادامه کد قبلی درج می کنیم :

```
if db_Path ~= "CANCEL" then
        Input.SetText("Database", db_Path);
        db = SQLite.Open(db_Path);
        Register.Creat(db)
end
```

آدرس در یک شی**Input** به نام **Database** نمایش داده میشود .
تابع **Register.Creat** فراخوانی میشود .

اجرای بانک اطلاعاتی در برنامه



ایجاد بانک اطلاعاتی به پایان رسید . در ادامه باید نحوه فراخوانی فایل ایجاد شده را بررسی کنیم .همانطور که در تصویر مشخص شده . این دکمه را برای فراخوانی فایل استفاده میکنیم .

```
db_Path = Dialog.FileBrowse(true, "Locate File", _DesktopFolder, "All Files (*.*)|*.*|", "", "dat", false, false)[1];
if db_Path ~= "CANCEL" then
        Input.SetText("Database", db_Path);
        db = SQLite.Open(db_Path);
end
```

ایجاد بانک جدید و اجرای فایل ساخته شده در برنامه با پایان رسید .

ذخیره داده ها در فایل SQLite :

در اینجا دستورات برنامه را به نحوی تعیین میکنیم که محتوای **Input** ها را در خانه های جدول ذخیره نماید . داده های رشته ای در **Autoplay** با فرمت **String** اعلام میشوند اما در **SQLite** با فرمت دیگری باید ذخیره شود . **SQLite** داده هایی را که با فرمت **String** در **Autoplay** ساخته میشود نمی شناسد بنابراین قبل از ارسال داده باید فرمت انها را تغییر دهیم .

در بخش **Global Function** تابع دیگری تعریف می کنیم .

```
function Enclose(strText)
        return string.format("%q", strText);
end
```

سپس با دستور **Insert into** آن را به پایگاه داده می فرستیم .

با توجه به تابع **Register.Creat** باید حتما دیتابیس فراخوانی شده باشد تا بتوانیم به آن داده ارسال کنیم

بنابراین در این کد شرط میگذاریم **if db** : چنانچه فایل بانک اطلاعاتی فراخوانی شده باشد اطلاعات را دریافت کند

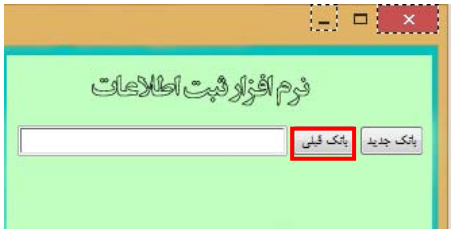
کد روبرو را در دکمه مربوط به ذخیره (صفحه قبل) درج میکنیم .

داده ها در Input ها را دریافت میکند و با ۴ نام متفاوت به SQLite ارسال میکند .

```
if db then
 name = Input.GetText("Input1");
 family = Input.GetText("Input2");
 father = Input.GetText("Input3");
 code = Input.GetText("Input4");
 tarikh = Input.GetText("Input5");
Query = "insert into Users values(NULL, "..Enclose(name).."', "..Enclose(family).."', "..Enclose(father).."',"..Enclose(code).."',"..Enclose(tarikh)..")";
        db = SQLite.Open(db_Path);
        SQLite.Query(db, Query, nil);
        SQLite.Close(db);
else
        Dialog.Message("اخطار", ".هیچ بانک اطلاعاتي جهت ذخیره وجود ندارد", MB_OK, MB_ICONEXCLAMATION, MB_DEFBUTTON1);
end
```

استفاده از Enclose در کدهای بالا برای تبدیل فرمت با دستور String.Format که در تابع Enclose تعریف کردیم قرار گرفته است .

نمایش جدول در Grid



برای نمایش دادن سطر و ستون جدول در برنامه از شی **Grid** استفاده میکنیم .

در ادامه کدهای مربوط به باز کردن جدول (تصویر روبرو) دستورات مربوط به **Grid** را می نویسیم .

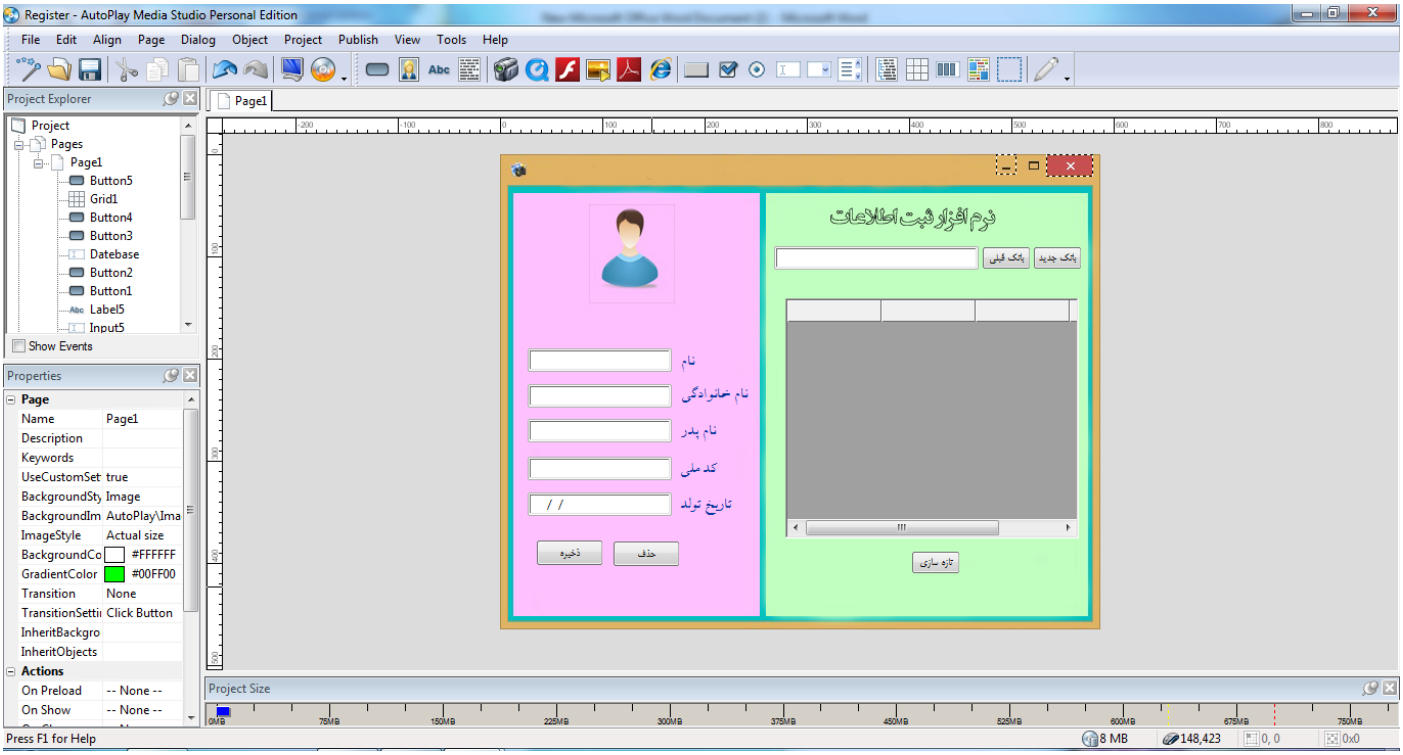
خانه های Grid را نام گذاری می کند

```
Grid.SetCellText("Grid1", 0, 0, "نام", true);
Grid.SetCellText("Grid1", 0, 1, "نام خانوادگي", true);
Grid.SetCellText("Grid1", 0, 2, "نام پدر", true);
Grid.SetCellText("Grid1", 0, 3, "کدملی ", true);
Grid.SetCellText("Grid1", 0, 4, "تاریخ تولد ", true);
db = SQLite.Open(db_Path);
tbResult = SQLite.QueryToTable(db, "select * from users");

if tbResult and tbResult.Rows >= 0 then
        Grid.SetRowCount("Grid1", tbResult.Rows+1);
        for i=1, tbResult.Rows do
                Grid.SetCellText("Grid1", i, 0, tbResult.Data[i]["name"], true);
                Grid.SetCellText("Grid1", i, 1, tbResult.Data[i]["family"], true);
                Grid.SetCellText("Grid1", i, 2, tbResult.Data[i]["father"], true);
                Grid.SetCellText("Grid1", i, 3, tbResult.Data[i]["code"], true);
                Grid.SetCellText("Grid1", i, 4, tbResult.Data[i]["tarikh"], true);
        end
else
        Dialog.Message("خطا", ". پایگاه داده ها پیدا نشد ", MB_OK, MB_ICONEXCLAMATION, MB_DEFBUTTON1);
        DialogEx.Close(-1);
end
Grid.AutoSize("Grid1", GVS_BOTH, true);
```

خانه های جدول در بانک اطلاعاتی را در Grid درج میکند

پیش نمایش برنامه



مراحل ساخت برنامه به اتمام رسید . فایل نمونه برنامه ساخته شده در پوشه همین آموزش موجود می باشد .