

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

برنامه‌سازی پیشرفته و کارگاه

دیتابیس مقدماتی ۱

استاد درس

دکتر مهدی قطعی

استاد دوم

بهنام یوسفی مهر

نگارش

سیدآرمان حسینی و یونس کاظمی

بهار ۱۴۰۳

فهرست

3	مقدمه
4	Relational Database
5	کدوم ؟AP Project
6	پیدا کردن قدم‌ها
9	اطلاعات متناقض
12	قدم‌های تموم شده
15	دست به کد بشید!
15	درست کردن پروژه توی DataGrip
18	درست کردن دیتابیس
21	درست کردن tableها
24	اضافه کردن داده‌ها
27	آتیشش بزنید
28	دوباره درستشون کنید
32	بالآخره select
35	پاک کردن رکوردها
36	تغییر ستون‌ها
39	مشاهده دیاگرام
40	چیزی که یاد گرفتیم
40	منابع بیشتر

"In God we trust. All others must bring data."

— *W. Edwards Deming*

مقدمه

تا به این جای کار، برنامه‌های شما راه خاصی برای ذخیره اطلاعاتشون بعد از اتمام اجرا نداشتند. اگر برنامه‌ای رو اجرا می‌کردید و می‌بستیدش، هر `variable` و هر چیزی که توی مموری ذخیره کرده بودین برای همیشه از دست می‌رفت و راهی هم برگرداندنش نبود!^۱

همون طور که احتمالاً حدس زدین، برنامه‌های جدی‌تر، این‌طور کار نمی‌کنن. هر بار که یک سایت خراب می‌شه و برنامه‌ش از کار می‌افته، کل دیتاش از بین نمیره. تقریباً تمام این برنامه‌ها، برای ذخیره داده‌هاشون از `database` استفاده می‌کنن. توی این داک، می‌خوایم اولین نگاه‌هایمان رو به دیتابیس‌ها بندازیم و ببینیم چطور می‌تونیم ازشون استفاده کنیم.

^۱ البته دروغ گفتم یه جوارایی... فکر کنم اگر خیلی دوست داشتین، می‌تونستید دیتاتون رو توی یه فایل بنویسید و موقع اجرای بعدی، از همون فایل بخونیدش. ولی خیلی کار راحتی نبود، نه؟

هاRelational Database

فرض کنید یه برنامه todo list درست کردین. توی این برنامه، شما یه کلاس Task دارین که کارهای مختلف کاربر رو نشون می‌ده و هر Task، ممکنه چند قدم (Step) داشته باشه. مثلاً ممکنه یک Task به اسم «پروژه AP» داشته باشین که شامل سه Step مثل «خوندن داکیومنت پروژه»، «نوشتن کد» و «تست کد» باشه.

شما یاد گرفتین که چطور کلاس‌های این todo list رو درست کنید. کد جاوای زیر این دو کلاس رو توصیف می‌کنه:

```
public class Task {
    public String name;
    public Date dueDate;
    public ArrayList<Step> steps;
}

public class Step {
    public String name;
}
```

فرض کنید برنامه شما برای مدتی کار کرده و کاربر، Task‌ها و Step‌های مختلفی برای خودش تعریف کرده. خیلی هم عالی. ولی هیچ‌کدام از ما نمی‌خوایم عصبانیت کاربرهاتون، بعد از این که todo list‌شون رو دوباره باز کردن و دیدن همه Task‌ها و Step‌هایی که تعریف کردن پریده رو ببینیم! چطور این دیتا رو توی به database ذخیره می‌کنید؟

ها چی هستن؟ Relational Database

۲ اون Relational Database‌ها، به شما اجازه می‌دن که داده‌هاتون رو، توی جدول‌های مختلف ذخیره کنید.^۲ مثلاً برای todo list‌تون، می‌توانید یه جدول به شکل زیر درست کنید:

^۲ اون Relational Database‌ای که توی اسمشون می‌بینید، ربط مستقیم داره به اون Relation‌هایی داره که توی مبانی علوم ریاضی یاد گرفتین. اون درسه اون‌قدران که فکر می‌کردین هم به درد نخور نیست! البته که کلا هم خیلی به درد بخورتر از چیزیه که فکرش رو می‌کنید.

tasks

name	due_date	steps
AP Project	2025/12/02	Reading doc Implementing project Testing program
DB Homework	2025/12/13	Learning about databases Writing db scripts
AP Project	2025/12/27	Reading doc Implementing project Testing program

این جدول، به خوبی می‌توانه تمام داده‌های برنامهٔ شما را نشون بده. هر کدام از فیلدهای Task توی برنامهٔ جاواتون، اینجا یک ستون دارن. توی relational database‌ها، به همچین جدولی یک relation یا table می‌گیم، به هر سطر اون یک record می‌گیم و به هر ستونش هم یک field می‌گیم. به همین سادگی! این جدول، به تنها یک دیتابیس کامله و هیچ مشکلی نداره. مگه نه؟ مگه نه؟

مومون یه خورده مشکل داره

ببخشید که باید اینو بگیم، ولی table بامزه کوچولوی ما چندان هم بی مشکل نیست. ولی خبر خوب این که این مشکلات، نسبتاً راحت برطرف می‌شن. بیاین به ترتیب این مشکلات رو ببینیم و با هم حل شون کنیم.

؟AP Project کدوم

فرض کنید که کاربر بهتون بگه که «تسک AP Project رو بهم نشون بده». همون طور که توی دیتابیس می‌بینید، کاربر دوتا تسک به این اسم تعریف کرده. کدومش رو نشونش می‌دید؟

ریشهٔ مشکل اینه که ما راهی برای تمایز دو Task نداریم. این که روی یکتا بودن اسم Task حساب باز کنیم کار خوبی نیست، و اگر کاربر دو Task کاملاً یکسان تعریف کنه ما نمی‌تونیم بفهمیم که

کدومشون کدومه. چه کار کنیم که این مشکل برطرف بشه؟ اینجا اونجاییه که باید متوقف شید و به راه حل این موضوع فکر کنید...

برای حل این مشکل، بیاین یه ستون جدید به اسم id درست کنیم و یک عدد یکتا به هر Task نسبت بدیم:

tasks

id	name	due_date	steps
1	AP Project	2025/12/02	1 - Reading doc 3 - Implementing project 4 - Testing program
2	DB Homework	2025/12/13	2 - Learning about databases 6 - Writing db scripts 5 - Reading doc
3	AP Project	2025/12/27	7 - Implementing project 8 - Testing program

می‌بینید که علاوه بر task‌ها، من یک id برای هر step هم گذاشتم تا خود step‌ها هم از هم دیگه قابل تمایز باشن. حالا به جای این که بگیم «تسکی به اسم AP Project رو بهم بده»، می‌تونیم بگیم «تسکی با id ۳ رو بهم بده» و معلومه که داریم راجع به تسک سوم لیست بالا صحبت می‌کنیم.

این ستون انقدر پر کاربرده که اسم خودش رو داره و بهش primary key می‌گن. اسمش یادتون بمونه که بعداً باهاش کار داریم.

خب، one problem down. بیاین به سراغ مشکل بعدی مون برمیم.

پیدا کردن قدم‌ها

پردازش ستون‌های این دیتابیس برای شما کار راحتیه. اگر کاربر ازتون تمام تسک‌هایی که اسمشون آئه رو بخواهد، می‌توانید با جستجو توی ستون name اون‌ها رو بهشون نشون بديد. اگر

ازتون تسک‌هایی که due date داشون قبل یه تاریخ خاصه رو بخواه هم به راحتی ستون due date رو نگاه می‌کنید.

ولی اگر کاربر ازتون بخواه که قدمی که `id` شده بده چطور؟ اطلاعات قدم‌ها، از اسمشون و آشون و هر اطلاعات اضافه‌ای که راجع بهشون ذخیره کردید، همه و همه توی فقط یک ستون جدول‌تون ذخیره شده. خلاصه‌ش که شما راه چندان خوبی برای پردازش درخواست‌های کاربر برای step‌ها ندارید.

و مشکل حتی بدتر هم می‌شه. فرض کنید که شما می‌خواستید به کاربر امکان فهرست‌بندی task‌هاش رو هم بدهی. کاربر ممکنه فهرستی به اسم «دانشگاه» برای تسک‌های دانشگاهی شعریف کنه، فهرستی به اسم «شرکت» برای تسک‌های شرکتی‌ش، و الی آخر. یه چیزی مثل این:

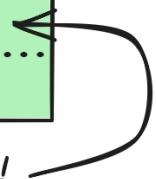
```
public class TaskList {
    public int id;
    public String name;
    public ArrayList<Task> tasks;
}
```

حالا دیتابیس‌تون چه شکلی می‌شه؟ فرض کنید مثل table‌های قبلی‌تون، task‌ها رو به عنوان ستونی از جدول task_lists تعریف کنید:

task_lists

<code>id</code>	<code>name</code>	<code>tasks</code>
.....

steps are somehow here!



حالا چطور step‌هاتون رو تعریف می‌کنید؟ ستونی از ستون tasks نه تنها این کار مشکلات خودش رو داره^۳، بلکه ستون‌هاتون هم، خیلی سریع، خیلی پیچیده می‌شن!

ریشه مشکل این‌جاست که توی یکی از ستون‌های دیتابیس‌تون، کل اطلاعات چندتا از record‌هاتون رو چپوندین. چطور می‌تونید این مشکل رو حل کنید؟ (متوقف شید و فکر کنید).

^۳ اولین مشکلش اینه که relational database به کل اجازه نمی‌دین ستونی داخل ستون دیگه تعریف کنید!

راه حلش اینه که برای step‌ها، به کل یه table جداگانه درست کنیم. برای این که بدونیم کدوم متعلق به چه task‌ایه، یه ستون به اسم task_id به این table اضافه می‌کنیم که id تسك مرتبط با این قدم رو نگه می‌داره^۴:

tasks

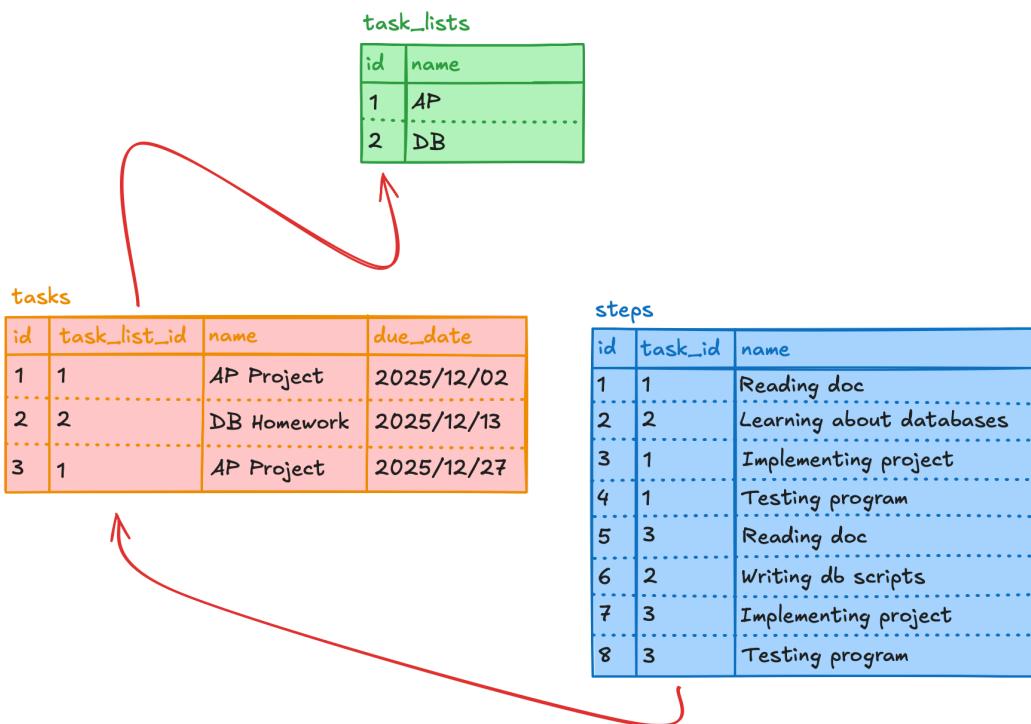
id	name	due_date
1	AP Project	2025/12/02
2	DB Homework	2025/12/13
3	AP Project	2025/12/27

steps

id	task_id	name
1	1	Reading doc
2	2	Learning about databases
3	1	Implementing project
4	1	Testing program
5	3	Reading doc
6	2	Writing db scripts
7	3	Implementing project
8	3	Testing program

خیلی خوب، خیلی بهتر شد. الان به جای این که تمام فیلدهای چند step مختلف توی یه جدول زور چپون شده باشه، table مستقل خودشون رو دارن. اضافه کردن فهرستهای دوستداشتی کاربرها به برنامه‌تون هم، از همیشه ساده‌تره. کافیه به جدول tasks ستونی به اسم task_list_id اضافه کنید:

⁴ این ستون‌ها هم خیلی معروف‌ن و اسم خودشون رو دارن. بهشون foreign key می‌گن.



شاید این طراحی به چشم شما، انسان خواننده، پیچیده بیاد. ولی باور کنید کامپیوترتون بابت ثانیه به ثانیه وقتی که موقع پردازش این داده‌ها صرفه‌جویی می‌کنه دعاتون می‌کنه!

وقتی که هیچ‌کدام از جداول دیتابیس شما، فیلدهای دردرس سازی مثل tasks, steps یا هر فیلدی که هم‌مان شامل چند record نداشته باشن، به اون دیتابیس «نرمال سطح اول» یا «first normal form (NF1)» می‌گن. به بهبود کیفیت دیزاین دیتابیس‌ها هم normalization می‌گن.

اطلاعات متناقض

بیاین یه فیچر به todo list اضافه کنیم. فرض کنید یه شرکت بزرگ برای مدیریت task‌های کل شرکت از این todo list استفاده می‌کنه و می‌خواهد هر task را به یکی از کارمندهای شرکت اختصاص بده. برای این کار، ما اطلاعات کارمند مسئول هر task را توی خود کلاس Task می‌نویسیم:

```
public class Task {
    public int id;
    public int taskListId;
    public String employeeName;
    public String employeeNationalId;
    public String name;
    public Date dueDate;
}
```

فیلدهای `employeeName` و `employeeNationalId` به ترتیب اسم و کد ملی کارمند مسئول این `task` را نشون می‌دان^۵. حالا اگر بخوایم به هر تスク یک کارمند اختصاص بدیم، از کد زیر استفاده می‌کنیم:

```
apProject1.employeeName = "Gholi";
apProject1.employeeNationalId = "1234567890";
dbHomework.employeeName = "Torob";
dbHomework.employeeNationalId = "9087653421";
apProject2.employeeName = "Mamad";
apProject2.employeeNationalId = "1234567890";
```

اگر بخوایم جدول `tasks` معادل این کلاس بموانه، باید ستون‌های `employee_name` و `employee_national_id` هم بهش اضافه کنیم.

tasks

<code>id</code>	<code>task_list_id</code>	<code>employee_name</code>	<code>employee_national_id</code>	<code>name</code>	<code>due_date</code>
1	1	Gholi	1234567890	AP Project	2025/12/02
2	2	Torob	9087653421	DB Homework	2025/12/13
3	1	Mamad	1234567890	AP Project	2025/12/27

خب، دیتابیس‌مون هنوزم کار می‌کنه. ولی اگر دقت کنید، کد ملی قلی و ممد توی دیتابیس بالا یکسانه. احتمالا وقتی یکی داشته تسك سوم رو وارد می‌کرده، به جای اسم قلی، ممد رو نوشته. این اشتباه، علاوه بر بی‌دقیقی کاربر `todo list` شما، ضعف دیزاین‌تون رو هم نشون می‌ده. چه کار می‌تونستیم بکنیم که جلوی بی‌دقیقی کاربر رو بگیریم؟

بیاین اول مشکل رو توی کد جاوامون حل کنیم و این کد رو یه خورده تمیزتر بنویسیم. کلاس `Employee` رو جدا می‌کنیم، و توی کلاس `Task` فقط یک فیلد از جنس `Employee` رو نگه می‌داریم:

```
public class Employee {
    public String name;
    public String nationalId;
}

public class Task {
    public int id;
    public int taskListId;
    public Employee responsibleEmployee;
    public String name;
```

اگر فکر می‌کنید بهتر بود یک کلاس `Employee` تعریف کنیم، توی اون فیلدهای `name` و `nationalId` رو تعریف کنیم و توی کلاس `Task`، فیلدی به شکل `Employee responsibleEmployee` داشته باشیم، کاملا درست فکر می‌کنید! بهش می‌رسیم.

```
    public Date dueDate;
}
```

بعد از این کار، لازمه برای هر کارمند یک آبجکت Employee درست کنیم:

```
Employee gholi = new Employee("Gholi", "1234567890");
Employee torob = new Employee("Torob", "9087653421");
```

و بعدش، از همین آبجکت‌ها موقع اختصاص دادن تسک‌ها به هر کارمند استفاده می‌کنیم:

```
apProject1.responsibleEmployee = gholi;
dbHomework.responsibleEmployee = torob;
apProject2.responsibleEmployee = gholi;
```

چون توی کد بالا، از آبجکت gholi برای هر دو تسک apProject1 و apProject2 استفاده کردیم، دیگه ممکن نیست که اسم قلی به اشتباه نوشته بشه.

بیاین جداول دیتابیس‌مون هم بر این اساس تغییر بدیم. یک جدول به اسم employees، معادل کلاس Employee درست می‌کنیم و به جدول tasks هم یه ستون به اسم employeeId اضافه تا کارمند مسئول اون تسک رو نشون بده:



employees		
id	name	national_id
1	Gholi	1234567890
2	Torob	9087653421

tasks				
id	task_list_id	employee_id	name	due_date
1	1	1	AP Project	2025/12/02
2	2	2	DB Homework	2025/12/13
3	1	1	AP Project	2025/12/27

دیتابیس ما، به لحاظ دیزاین یک سطح بهتر شد. الان که داده‌های نامریوط به task رو به جدول خودشون منتقل کردیم، دیتابیس ما توی «سطح دوم نرمال‌سازی» یا «NF2» نیست.

قدم‌های تmom شده

شما از همیشه todo list شرکت بزرگ و کوچیک ازش استفاده می‌کنن. این شرکت‌ها می‌خوانند بدون کارمندهاشون از هر تسك، چند قدم رو انجام دادن. برای این کار، شما اول ستون is_completed رو به جدول steps اضافه می‌کنید تا وضعیت هر قدم مشخص بشه. بعد از آن، ستون‌های number_of_completed_steps و total_number_of_steps هم به جدول tasks اضافه می‌کنید که به ترتیب، تعداد کل قدم‌ها و قدم‌های انجام شده هر تسك رو نشون بدن:



tasks						
id	task_list_id	employee_id	total_number_of_steps	number_of_completed_steps	name	due_date
1	1	1	3	2	AP Project	2025/12/02
2	2	2	2	1	DB Homework	2025/12/13
3	1	1	3	0	AP Project	2025/12/27

steps			
id	task_id	name	is_completed
1	1	Reading doc	true
2	2	Learning about databases	true
3	1	Implementing project	true
4	1	Testing program	false
5	3	Reading doc	false
6	2	Writing db scripts	false
7	3	Implementing project	false
8	3	Testing program	false

چک کنید که اطلاعات ستون‌های tasks و steps is_completed منطبق باشند. همون‌طور که می‌بینید، تسك اول ۲ قدم انجام شده داره (قدم‌های «Reading doc» و «Implementing project»)، تسك دوم ۱ قدم انجام شده داره (قدم «Learning about databases») و تسك سوم هیچ قدم انجام شده‌ای نداره.

دیتابیس ما درست کار می‌کنه! ولی خب، به چه قیمتی؟ هر برنامه‌نویس باید حواسش باشه که اگر جایی از کد، وضعیت یک step رو به «انجام شده» تغییر داد، حتماً ستون number_of_completed_steps هم تغییر بده، و اگر task ای رو به اضافه کرد، ستون total_number_of_steps هم عوض کنه. فرض کنید یه برنامه‌نویسی این رو یادش بره، اون وقت توی دیتابیس شما اطلاعات متناقض ثبت می‌شه:

tasks

id	task_list_id	employee_id	total_number_of_steps	number_of_completed_steps	name	due_date
1	1	1	3	1	AP Project	2025/12/02
2	2	2	2	0	DB Homework	2025/12/13
3	1	1	3	0	AP Project	2025/12/27

steps



id	task_id	name	is_completed
1	1	Reading doc	true
2	2	Learning about databases	true
3	1	Implementing project	true
4	1	Testing program	false
5	3	Reading doc	false
6	2	Writing db scripts	false
7	3	Implementing project	false
8	3	Testing program	false

همون‌طور که می‌بینید، توی جداول بالا مقادیر ستون number_of_completed_steps درست نیستن.⁶ شما چطور می‌تونستید با دیزاینی بهتر، به کل جلوی این اشتباه برنامه‌نویس‌ها رو بگیرید؟ (متوقف شید و فکر کنید)

راه حل اینه که ساده‌تر فکر کنید! کلا از دست دو ستون جدید جدول tasks خلاص بشید:

⁶ حتی خود نویسنده هم یکی-دو بار توی شمردن این‌ها اشتباه کرده راستش!

The diagram illustrates a many-to-one relationship between the `tasks` table and the `steps` table. The `steps` table has a foreign key `task_id` that references the primary key `id` in the `tasks` table.

tasks

id	task_list_id	employee_id	name	due_date
1	1	1	AP Project	2025/12/02
2	2	2	DB Homework	2025/12/13
3	1	1	AP Project	2025/12/27

steps

id	task_id	name	is_completed
1	1	Reading doc	true
2	2	Learning about databases	true
3	1	Implementing project	true
4	1	Testing program	false
5	3	Reading doc	false
6	2	Writing db scripts	false
7	3	Implementing project	false
8	3	Testing program	false

حالا اگر کاربر ازتون تعداد کل قدم‌های یک تسك رو خواست چه کار می‌کنید؟ کافیه به جدول `steps` نگاه کنید تا ببینید چند قدم برای تسك مورد نظر کاربر ثبت شده! اگر قدم‌های تموم شده اون تسك رو خواست چطور؟ دوباره جدول `steps` رو نگاه می‌کنید تا ببینید چند قدم از اون تسك تموم شدن. در واقع، مشکل این بود که ستون‌های `number_of_completed_steps` و `total_number_of_steps`، بر اساس اطلاعات جدول `steps` قابل محاسبه بودن. بعضی وقتا راه حل یه مسئله این نیست که کد جدیدی بنویسید، بلکه اینه که کدهای قبلی رو پاک کنید.

اگر توی یه دیتابیس، مقدار هیچ ستونی بر حسب ستون‌های دیگه اون دیتابیس قابل محاسبه نباشه (مثل ستون‌های `number_of_completed_steps` و `total_number_of_steps`)، به اون دیتابیس «نرمال سطح ۳» یا «third normal form (3NF)» می‌گن.

دست به کد بشید!

دیتابیس todo list ما، یه دیزاین خیلی تمیز و خوب روی کاغذ داره. ولی خب، این دیزاین هنوز روی کاغذه! وقتشه که DataGrip را باز کنیم، جدول‌های record را درست کنیم، بهش اضافه کنیم. برای این کار، از زبانی به اسم SQL استفاده می‌کنیم.

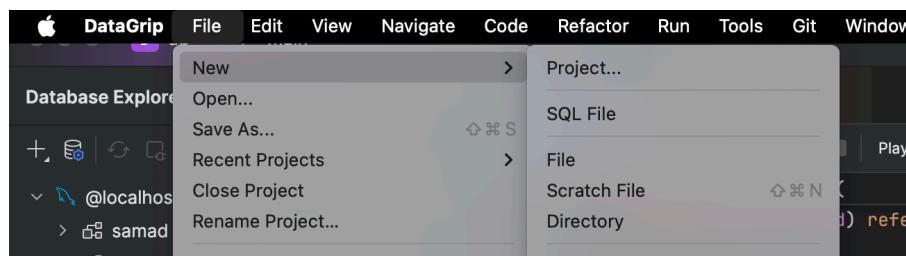
SQL

برای کار با دیتابیس، از زبانی به اسم SQL استفاده می‌شده. این زبان، یه مقدار از زبان‌های برنامه‌نویسی‌ای که تا الان دیدین متفاوته، و طراحی شده تا هر کسی، بدون داشتن سابقه خاصی توی برنامه‌نویسی بتونه باهاش کار کنه. به خاطر همین هم کدهایی که با اون نوشته می‌شده، خیلی شبیه زبان انگلیسیه. هر کدی که از اینجا به بعد می‌نویسید، به زبان SQLه.

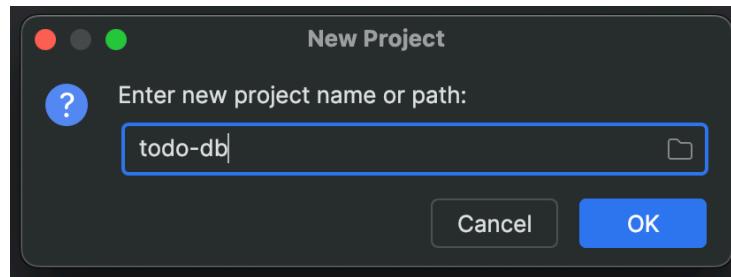
صحبت کردن بسه! شما اینجا باید که کد رو ببینید، پس بباید اول از همه، به پروژه کوچیک توی درست کنیم. DataGrip

درست کردن پروژه توی DataGrip

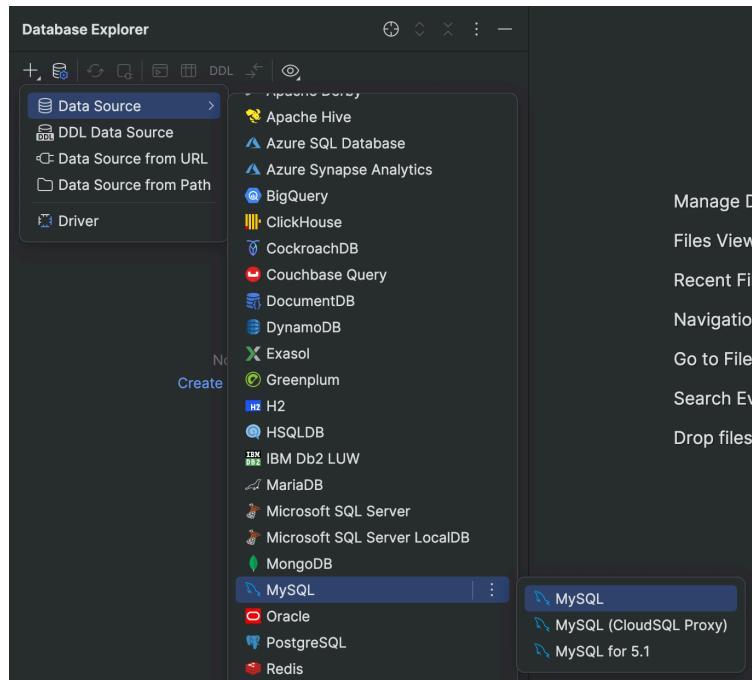
رو باز کنید و از منوی File، بخش New، دکمه Project را بزنید:



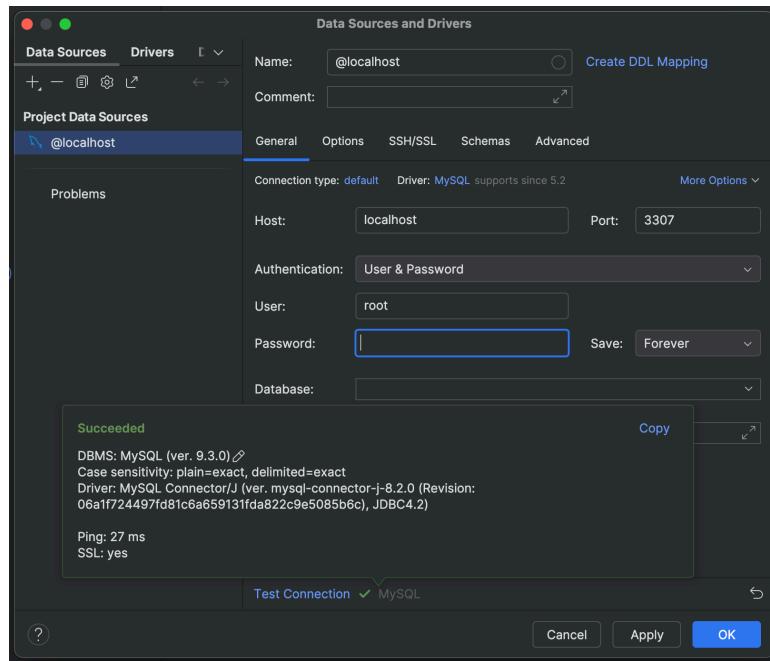
اسم پروژه‌تون رو انتخاب کنید. اگر لازم بود، مسیر دایرکتوری پروژه‌تون هم مشخص کنید. در غیر این صورت DataGrip Projects home توي دایرکتوری DataGrip Projects رو توی پروژه‌تون را ایجاد می‌کنه:



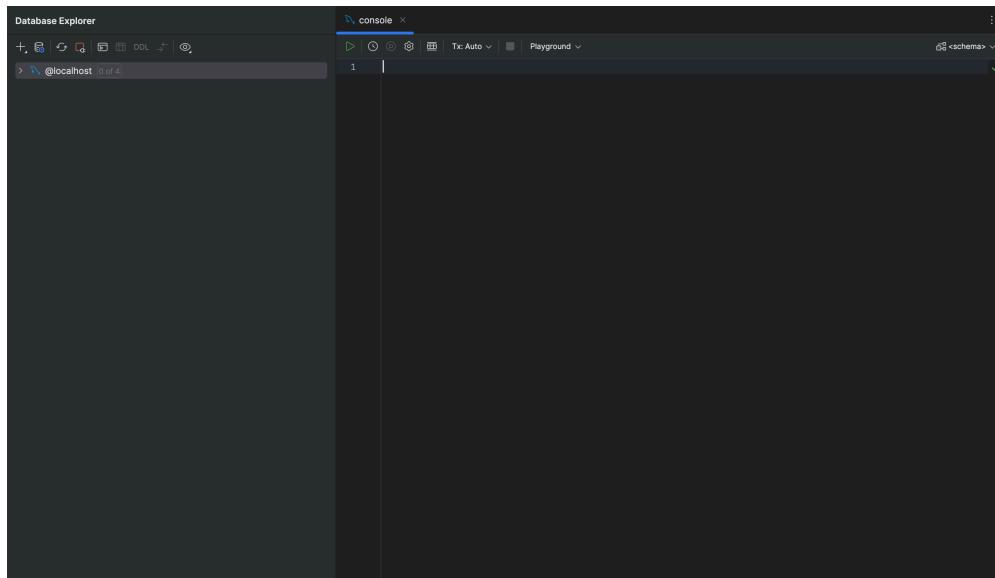
حال با اون مثبت کوچولوی بالای صفحه، mysql رو به DataGrip وصل کنید:



اطلاعات mysql رو پر کنید و با استفاده از دکمه **Test Connection**، چک کنید که با موفقیت به mysql وصل بشه. اگر نمی‌دونید توی این صفحه چه کار کنید، به داک «نصب و راهاندازی و کار با DataGrip» مراجعه کنید. نهایتاً دکمه **OK** رو بزنید:



بعد از اضافه شدن mysql instance که از داشتین، می‌توانید آون رو سمت چپ صفحه ببینید.
علاوه بر این، تب console هم برآتون باز می‌شه که از طریق آون می‌توانید دستورات SQL رو اجرا کنید:



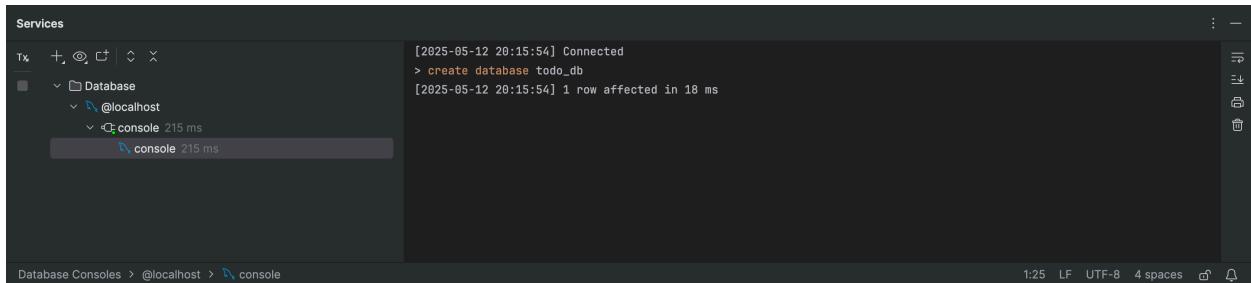
حالا، شما mysql instance که از داشتین رو با موفقیت به DataGrip اضافه کردین.

درست کردن دیتابیس

هر از mysql instance، می‌توانه دیتابیس‌های مختلفی توی خودش داشته باشه. اول از همه، باید دیتابیس برنامه خودتون رو ایجاد کنید. دستور زیر رو توی console وارد کنید:

```
create database todo_db;
```

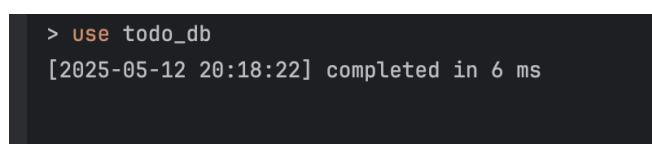
با استفاده از این دستور، می‌توانید یک database به اسم todo_db ایجاد کنید. سمی‌کالن (;) انتهای دستورات اختیاریه، ولی خوبه که همیشه بذارینش. با زدن دکمه اجرا، این دستور رو اجرا کنید. می‌بینید که تب زیر به پایین DataGrip اضافه می‌شه:



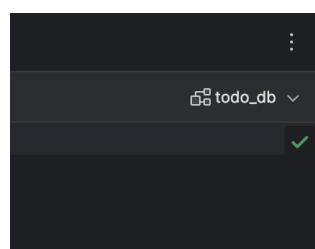
اگر هیچ چیزی قرمز نبود (!) یعنی دیتابیس‌تون با موفقیت ساخته شده. دستور قبلی رو از کنسول پاک کنید و دستور زیر رو بنویسید تا کوئری‌های بعدی‌تون توی همین دیتابیس اجرا بشه:

```
use todo_db;
```

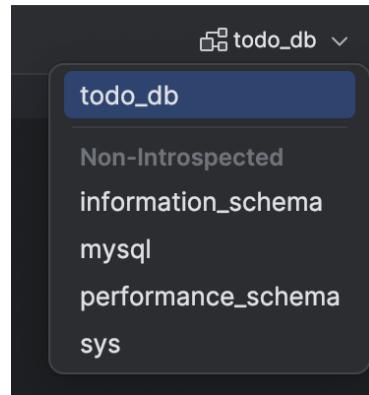
دباره اجراش کنید، خط زیر به تب Services اضافه می‌شه:



دباره اگر هیچ چیزی قرمز نبود، یعنی همه چی درست کار کرده. شاید همزمان حواستون به این نکته هم جلب شده باشه که بالا سمت چپ صفحه، یه همچین آیکونی پدیدار شده که اسم دیتابیس‌تون روشن نوشته شده:

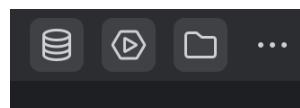


این یعنی که دستورات کنسول از این به بعد روی دیتابیس todo_db اجرا می‌شود. روی این دکمه کلیک کنید:

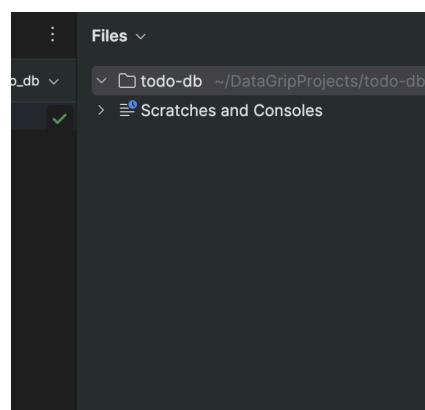


چندتا دیتابیس دیگه هم اینجا می‌بینید. این‌ها، دیتابیس‌هایی‌ان که خود mysql برای نگهداری از اطلاعات این instance ایجاد می‌کنند. ترجیحاً بهشون دست نزنید!

باید یه دیتابیس دیگه هم برای گرم شدن دستمون درست کنیم. اگر حواستون باشه، دستوراتی که توی console می‌نویسید هیچ‌جا ذخیره نمی‌شن. در واقع ما از console فقط برای تست و بررسی اطلاعات دیتابیس استفاده می‌کنیم. برای این که دستورات SQL‌تون رو یه جا ذخیره کنید لازمه که یه فایل sql درست کنید. اول اون دکمهٔ پوشه‌مانند رو از بالای DataGrip بزنید:

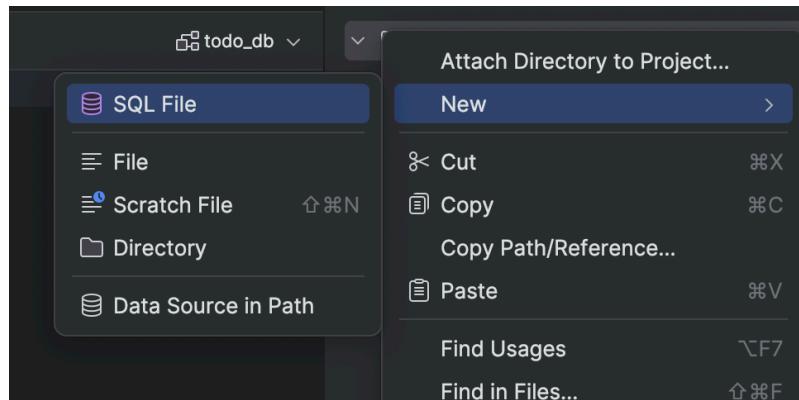


تب Files برآتون باز می‌شه⁷:

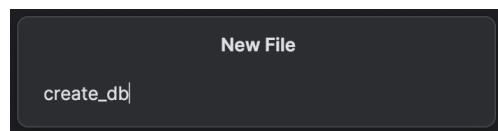


⁷ خوبی محصولات JetBrains اینه که الاشون به هم شبیه‌ئه. DataGrip خیلی شما رو یاد لینیا می‌ندازه، نه؟

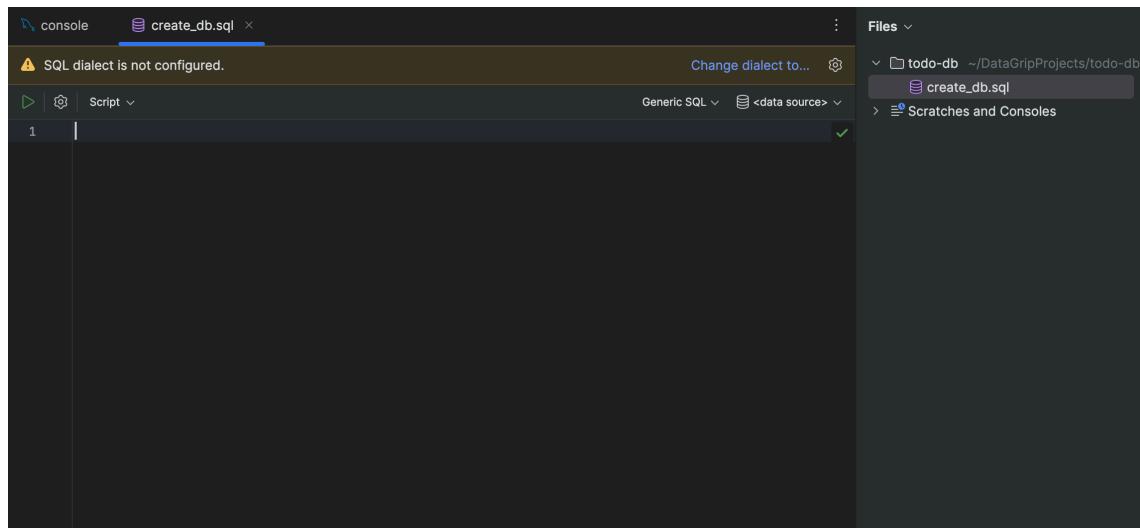
روی دایرکتوری todo-db راست کلیک کنید، تب New را انتخاب کنید و SQL File رو بزنید:



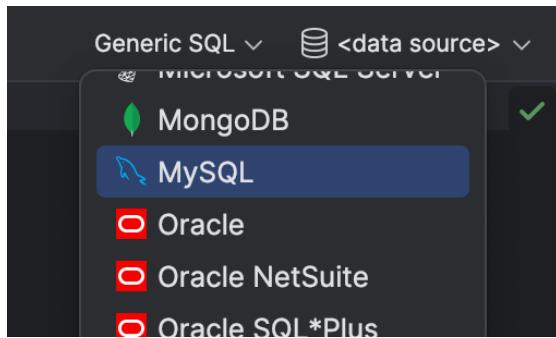
حالا اسم فایل‌تون رو انتخاب کنید:



بعد از درست کردن فایل‌تون، می‌بینید که یک تب جدید کنار console برای اون باز می‌شه:



از اون بالا، اون جایی که نوشه MySQL را انتخاب کنید. این به می‌گه که دارین برای دیتابیس MySQL کد می‌زنین و باعث می‌شه که DataGrip راحت‌تر توی کد زدن کمک‌تون کنه:



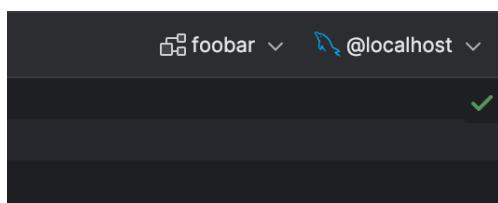
دستورات زیر را برای ایجاد دیتابیس و استفاده از آن بنویسید:

```
create database foobar;
use foobar;
```

حالا اسکریپتون را اجرا کنید:

```
> create database foobar
[2025-05-12 20:44:44] 1 row affected in 14 ms
> use foobar
[2025-05-12 20:44:44] completed in 5 ms
```

می‌بینید که دستورات `create_db.sql` از این به بعد توی دیتابیس `foobar` اجرا می‌شه:



حالا برای حذف این دیتابیس از دستور زیر استفاده کنید:

```
drop database foobar;
```

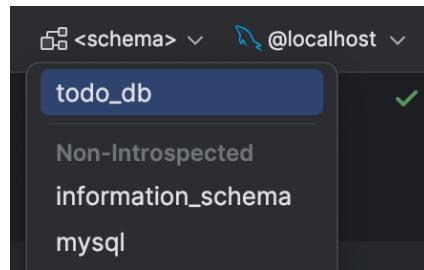
درست کردن table‌ها

خیلی خوب، حالا که دیتابیس‌مون آماده‌ست، وقتی که `table`‌ها‌مون رو درست کنیم. از جدول `tasks` شروع می‌کنیم. اگر یادتون باشه، هر `task` یک `id` داشت، یک `name` و یک `due_date`. فایل `tasks.sql` رو درست کنید و دستور زیر را توی آن بنویسید:

```
create table tasks (
    id int,
    name nvarchar(255),
```

```
due_date date
);
```

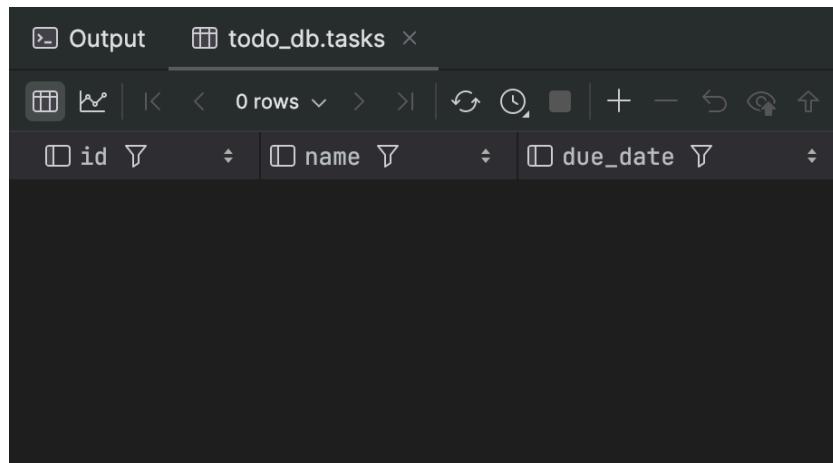
یه مقدار جلوتر بهتون توضیح می‌دم که این دستورات یعنی چی، ولی الان حواستون باشه که قبل از اجرای کد بالا اون، دیتابیس todo_db رو برای اجرا انتخاب کنید:



کوئری‌تون رو اجرا کنید. به تب console بربین تا خیلی سریع با هم اولین جدول‌مون رو ببینیم. کد زیر رو توی کنسول بنویسید و اجرا کنید، بعدها دستور select رو دقیق‌تر بهتون توضیح می‌دم:

```
select * from tasks;
```

اولین table، in all it's glory، روی صفحه‌ست! تبریک!



یه لحظه صبر کنید. قبل از ساخت جدول steps، بیاین اون دستور create table که بالاتر زدیم رو دقیق‌تر بررسی کنیم. یه همچین دستوری بود:

```
create table tasks (
    id int,
    name nvarchar(255),
    due_date date
);
```

کد، به خودی خود، نسبتاً گویاست. با این دستور، یه جدول به اسم tasks درست کردیم که سه ستون داره. اسم اولین ستونش idه و جنسش int. دومین ستونش، یعنی name، یه مقدار از جنس (255) nvarchar(255) و این یعنی این ستون یه رشته، با حداکثر 255 کاراکتر توی خودش نگه می‌داره. اگر به جای ۲۵۵، از سایز دیگه‌ای، مثلًا nvarchar(50) استفاده می‌کردیم، حداکثر ۵۰ کاراکتر توش جا می‌گرفت. ستون سوم، یعنی due_date هم از جنس date.

تا دستمون گرمه، جدول steps هم درست کنیم. فایل steps.sql رو ایجاد کنید و دستور زیر رو توی اون بنویسید:

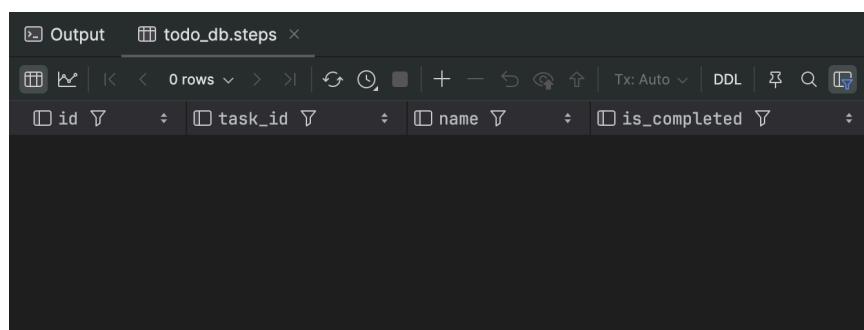
```
create table steps (
    id int,
    task_id int,
    name nvarchar(255),
    is_completed boolean
);
```

این دستور، تا حد خوبی شبیه دستور قبلیه. توی اون، یکی از data type جدید mysql به اسم boolean رو می‌بینید.^۸ احتمالاً می‌تونید حدس بزنید این ستون چه مقادیری رو نگه می‌داره دیگه، مگه نه؟

بعد از اجرای دستور بالا، به console برگردین و دستور زیر رو برای دیدن table جدیدتون اجرا کنید:

```
select * from steps;
```

تبریک دوباره!



^۸ MySQL کلی data type دیگه هم داره. حتماً به [این لینک](#) یه نگاه بندازید تا با اون‌ها آشنا بشید. تایپ‌هایی مثل decimal و bigint خیلی مهمن و در آینده ازشون استفاده می‌کنید.

اضافه کردن داده‌ها

اون ها خیلی خالی به نظر می‌رسن، نه؟ بیاین تا اولین record هامون رو به دیتابیس اضافه کنیم! برای این کار، از دستور `insert into` استفاده می‌شه:

```
insert into tasks(id, name, due_date)
values (1, 'AP Project', '2025-10-12');
```

این دستور رو توی console بنویسید و اجراس کنید. بعدش دوباره `select` بزنید:

```
select * from tasks;
```

می‌تونید اولین رکوردتون رو توی جدول tasks ببینید:

	<code>id</code>	<code>name</code>	<code>due_date</code>
1	1	AP Project	2025-10-12

حالا برای این task، دو تا step هم تعریف کنید:

```
insert into steps(id, task_id, name, is_completed)
values (1, 1, 'Reading doc', false);

insert into steps(id, task_id, name, is_completed)
values (2, 1, 'Implementing the project', false);
```

بعد از اجرای دستور بالا، دوباره `select` تا step‌های جدیدتون رو هم ببینید:

```
select * from steps;
```

خروجی‌ش همچین شکلیه:

	<code>id</code>	<code>task_id</code>	<code>name</code>	<code>is_completed</code>
1	1	1	Reading doc	0
2	2	1	Implementing the project	0

یه چیزی که خوبه حواس‌تون بهش باشه اینه که ستون `is_completed` به جای `true` و `false`، مقادیر °^۹ و ۱ رو نگه‌داری می‌کنه.

لازمه دستور `insert into` رو دقیق‌تر بررسی کنیم. دستورش یک همچین ریختی داشت:

```
insert into tasks(id, name, due_date)
values (1, 'AP Project', '2025-10-12');
```

بعد از خود `insert into`، اسم جدول‌مون رو نوشتیم. بعد از اون، ستون‌هایی که می‌خوایم مقداردهی کنیم رو به ترتیب مشخص کردیم و توی خط بعد، مقادیر اون ستون‌ها رو، به همون ترتیبی که توی خط اول اومنده بودن، نوشتیم. این یعنی می‌توانستیم ستون‌ها رو به هر ترتیب دیگه‌ای هم بنویسیم:

```
insert into tasks(id, due_date, name)
values (2, '2025-10-12', 'DB Homework');
```

اگر این دستور رو اجرا کنید، می‌بینید که ستون‌ها به ترتیبی که مشخص کردیم مقداردهی شدن:

	□ id ↴	□ name ↴	□ due_date ↴
1	1	AP Project	2025-10-12
2	2	DB Homework	2025-10-12

اگر مقادیر ستون‌ها رو، به ترتیبی که توی دستور `create table tasks` اومنده بود مشخص کنید، لازم نیست که ترتیب ستون‌ها رو بنویسید. اگر یادتون بیاد، موقع ساخت جدول `tasks`، ستون‌ها به ترتیب زیر تعریف شدن:

```
create table tasks(
    id int,
    name nvarchar(255),
    due_date date
);
```

پس می‌توانیم `insert into` رو به شکل زیر بنویسیم:

```
insert into tasks
values (3, 'DS Homework', '2025-10-12');
```

اگر این دستور رو اجرا کنید می‌بینید که task جدید‌تون هم با موفقیت توی دیتابیس ذخیره شده:

^۹ در واقع MySQL اصلاً `boolean` به اسم `tinyint(1)` ذخیره می‌شون که فقط اعداد ۰ و ۱ رو نگه می‌داره.

	id	name	due_date
1	1	AP Project	2025-10-12
2	2	DB Homework	2025-10-12
3	3	DS Homework	2025-10-12

یه نکته کوچولو، می‌بینید که رشته‌ها رو به شکل 'DS Homework' مشخص کردیم. توی SQL فقط از double quotes برای مشخص کردن رشته‌ها استفاده می‌کنیم و نمی‌تونیم مثل جاوا از single quote ('') استفاده کنیم. اجرای دستور زیر خطای داشته باشد:

```
insert into tasks
values (3, "DS Homework", '2025-10-12');
```

همچنین due_date رو هم به شکل یک رشته مشخص کردیم. خود mysql قبل ذخیره اون سال و ماه و روز این رشته رو جدا می‌کنه و اون رو به date تبدیل می‌کنه.

خیلی هم خوب، حالا بیاین یه مقدار از دستور insert into سوء استفاده کنیم! اولین کاری که می‌کنیم اینه که اسم یک task رو مشخص نمی‌کنیم:

```
insert into tasks(id, due_date)
values (3, '2025-10-12');
```

اجراش کنید و بعد select بزنید. جدولتون یه همچین شکلی داره:

	id	name	due_date
1	1	AP Project	2025-10-12
2	2	DB Homework	2025-10-12
3	3	DS Homework	2025-10-12
4	3	<null>	2025-10-12

می‌بینید که رکورد جدیدتون هم با موفقیت ذخیره شده، و توی ستون name عبارت null افتاده. این یعنی تسک جدیدتون بدون اسمه! نه تنها بدون اسمه، که چون یادمون رفت id اش هم عوض کنیم، id اون هم با تسک DS Homework یکیه. شما حتی می‌توانستید تمامی ستون‌ها رو null بذارید:

```
insert into tasks
values ();
```

می‌بینید؟ رکورد جدیدمون (اگر بتونیم اسمش رو رکورد بذاریم!) نه اسم دارد، نه id و نه due_date همه‌ش null‌ه!^{۱۰}

□ id ▾	□ name ▾	□ due_date ▾
1	AP Project	2025-10-12
2	DB Homework	2025-10-12
3	DS Homework	2025-10-12
4	<null>	2025-10-12
5	<null>	<null>

بذارین یه خرابکاری دیگه هم نشونتون بدم قبل این که پیش بريم. شما می‌تونید step‌ای اضافه کنید که task_id هم نداره:

```
insert into steps(id, task_id, name, is_completed)
values (100, 1234, 'bluh', 0);
```

step جدید ما برای تسک ۱۲۳۴ تعریف شده، که وجود نداره!

□ id ▾	□ task_id ▾	□ name ▾	□ is_completed ▾
1	1	1 Reading doc	0
2	2	1 Implementing the project	0
3	100	1234 bluh	0

خوشبختانه، MySQL راه و روش خودش رو برای جلوگیری از اضافه شدن این رکوردهای نامربوط دارد. قبل از این که این مشکلات رو حل کنیم، بذارید از شر این جدول‌های بی‌کیفیت^{۱۰} راحت بشیم.

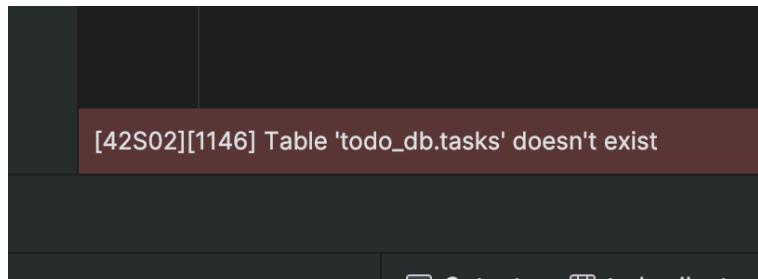
آتیشش بزنید

برای حذف جدول‌هایمان از دستور drop table استفاده می‌کنیم. دستورات زیر رو توی console وارد کنید:

```
drop table tasks;
drop table steps;
```

بعد از اجرای اون، می‌بینید که اگر دوباره روی این جداول select بزنید به خطای خورید:

^{۱۰} ببخشید که به اولین جدول‌هایتون گفتم بی‌کیفیت. اونقدر انم بی‌کیفیت نبودن راستش. :



تموم!

دوباره درستشون کنید

به فایل tasks.sql برگردید و کدش رو با کد زیر جایگزین کنید:

```
create table tasks(
    id int primary key auto_increment,
    name nvarchar(255) not null,
    due_date date not null
);
```

یه سری چیز جدید اضافه شدن، بیاین دونه دونه بررسی‌شون کنیم.

اول از همه، جلوی بعضی از ستون‌هاتون علاوه بر type، عبارت not null هم اومنده. این عبارت، به mysql می‌گه که نذاره مقدار null به این ستون‌ها اضافه بشه.

دوم، عبارت primary key جلوی id اضافه شده. این به mysql می‌گه که ستون id، عبارت primary key جدول ماست. این یعنی که:

1. نباید null باشه. هر record حتما باید id داشته باشه.
2. id دو رکورد نباید یکسان باشه. به عبارتی مقادیر ستون id باید یکتا باشن.

سوم، عبارت auto_increment هم جلوی ستون id اضافه شده. این به mysql می‌گه که به صورت اتوماتیک، مقدار ستون id را توی هر insert into مشخص کنه تا ما لازم نباشه هر بار مقداردهی شنیم.

دستور create table رو اجرا کنید و بعدش توی console روند روی این جدول بزنید:

	id	name	due_date

ظاهر جدولتون تغییر چندانی نکرده. اگر دقت کنید یک کلید کوچولو کنار ستون id اضافه شده که یعنی این ستون، primary key این جدوله.

حالا دستور insert زیر رو توی console اجرا کنید:

```
insert into tasks(name, due_date)
values ('AP Project', '2025-12-12');
```

اگر دوباره select بزنید، میبینید که رکوردون با موفقیت به دیتابیس اضافه شده و حتی ستون id هم، اتوماتیک مقداردهی شده:

	id	name	due_date
	1	AP Project	2025-12-12

دقت کنید که اگر توی insert، ترتیب ستونها رو مشخص نکنید، mysql فکر میکنه که اولین مقدار توی values متعلق به ستون id است. پس دستور زیر کار نمیکنه:

```
insert into tasks
values ('DB Homework', '2025-12-12');
```

ولی شما همچنان میتونید id رو دستی مشخص کنید، پس دستور زیر کار میکنه:

```
insert into tasks
values (2, 'DB Homework', '2025-12-12');
```

و بعد از اجرای اون، select تون همچین خروجی ای میده:

id	name	due_date
1	AP Project	2025-12-12
2	DB Homework	2025-12-12

اگر یه رکورد دیگه هم اضافه کنید، می‌بینید که id اون هم به درستی مقدار دهی می‌شه:

```
insert into tasks(name, due_date)
values ('DS Project', '2025-12-12');
```

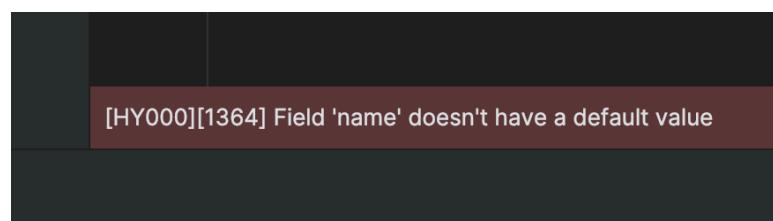
اگر دوباره select بزنید:

id	name	due_date
1	AP Project	2025-12-12
2	DB Homework	2025-12-12
3	DS Project	2025-12-12

بذرین دوباره تلاش کنیم که رکوردهای نامربوط به دیتابیس اضافه کنیم. توی دستور زیر، تلاش می‌کنیم ستون name رو خالی بذریم:

```
insert into tasks(due_date)
values ('2025-12-12');
```

اگر اون رو اجرا کنیم، با خطای زیر مواجه می‌شیم:



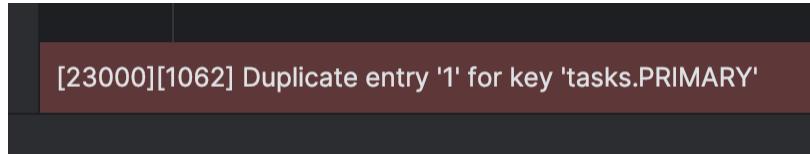
این خطأ به ما می‌گه که فیلد name مقداردهی نشده، و مقدار دیفالت^{۱۱} هم نداره.

اگر تلاش کنیم که مقدار تکراری توی ستون id بذریم:

^{۱۱} پنج خط پایین‌تر یاد می‌گیرید که چطور برای فیلدهاتون مقادیر دیفالت تعریف کنید. ولی ارزشش به اینه که خودتون سرچش کنید و یاد بگیریدش.

```
insert into tasks
values (1, 'FooBar', '2025-12-12');
```

باز هم خطای گیریم:



خب. خوبه! جدول tasks الان رکوردهای نامربوط ذخیره نمی‌کنه. حالا بیاین جدول steps هم درست کنیم. به steps.sql برید و کوئری زیر رو اونجا بنویسید:

```
create table steps(
    id int primary key auto_increment,
    task_id int not null,
    name nvarchar(255) not null,
    is_completed boolean not null default false,
    foreign key (task_id) references tasks(id)
);
```

رو که قبل دیدیم. اما پایین این دستور یه چیز جدید هست. به خط زیر توجه کنید:

```
foreign key (task_id) references tasks(id)
```

این خط، به mysql می‌گه که ستون task_id، یه foreign key که به ستون id از جدول tasks وصله. به خاطر همین موضوع، mysql مطمئن می‌شه که task_id هر step، حتماً توی جدول tasks وجود داشته باشد.

اگر حواستون باش، می‌بینید که ما ستون task_id هم not null تعريف کردیم. برخلاف اینها می‌تونن null باشن. اگر task_id مقدار null داشت، یعنی step ما به هیچ taskی وصل نیست و برای خودش مستقله. البته که ما این رو نمی‌خوایم، پس این ستون not null.

یه چیز جدید دیگه هم توی این جدول هست. برای ستون is_completed مقدار دیفالت false تعريف کردیم تا لازم نباشه توی هر insert اون رو مقداردهی کنیم:

```
is_completed boolean not null default false
```

حالا بیاین یه سری رکورد جدید به steps اضافه کنید. همه insert‌های زیر درستن:

```
insert into steps(task_id, name, is_completed)
values (1, 'Reading the doc', false);
```

```
insert into steps(task_id, name, is_completed)
values (1, 'Implementing the project', true);

insert into steps(task_id, name)
values (2, 'Learning about db');
```

اگر اونها رو اجرا کنید و بعدش روی این جدول select بزنید، خروجی زیر رو می‌بینید:

id	task_id	name	is_completed
1	7	1 Reading the doc	0
2	8	1 Implementing the project	1
3	9	2 Learning about db	0

اما اگر تلاش کنید مقدار نامربوطی رو به عنوان task_id مشخص کنید:

```
insert into steps(task_id, name)
values (100, 'Learning about db');
```

خطای زیر رو می‌گیرین:

22
[23000][1452] Cannot add or update a child row: a foreign key constraint fails ('todo_db`.`steps', CONSTRAINT `steps_ibfk_1` FOREIGN KEY (`task_id`) REFERENCES `tasks` (`id`))

این خطا (که توی سیستم من خیلی بد نمایش داده شده!) یعنی تسویه با id مشخص شده نیست و در نتیجه امکان اضافه کردن این رکورد وجود نداره.

بالاخره

تا این جای کار دیگه با این دستور به اندازه کافی آشنایی شدید:

```
select * from tasks;
```

این دستور، بهتون تمام تمام record های یک جدول رو نشون می‌ده. اون ستاره کوچولو اون وسط، یعنی این که می‌خوايد تمام ستون‌های این جدول رو ببینید. خیلی وقت‌ها، شما نمی‌خوايد تمام ستون‌های یک جدول رو ببینید! در این صورت، کافیه اسم ستون‌هایی که می‌خوايد رو به جای ستاره بنویسید. مثلًا دستور زیر:

```
select name from tasks;
```

فقط اسمی task هاتون رو نشون می‌ده:

	name
1	AP Project
2	DB Homework
3	DS Project

یا دستور زیر:

```
select task_id, name from steps;
```

فقط ستون‌های name و task_id از جدول steps را بهتون نشون می‌ده:

	task_id	name
1		Reading the doc
2		Implementing the project
3		Learning about db

خب، خیلی هم خوب. به نظرتون دستور زیر چه کار می‌کنه؟

```
select * from steps
where task_id = 1;
```

این دستور، فقط step‌هایی را نشون‌تون می‌ده که task_id یکه¹²:

	id	task_id	name	is_completed
1		7	Reading the doc	0
2		8	Implementing the project	1

مثل قبل، می‌تونید از mysql بخواید که فقط یه سری فیلد خاص از این step‌ها را نشون‌تون بده:

```
select name, is_completed from steps
where task_id = 1;
```

و خروجی این دستور، همچین شکلیه:

	name	is_completed
1	Reading the doc	0
2	Implementing the project	1

¹² برخلاف java و بیشتر زبان‌های برنامه‌نویسی که برای شرط تساوی از == استفاده می‌کنن، sql از = اضافه می‌کنه. ولی باقی شرط‌ها، مثل < و > و امثال این‌ها شبیه همون java هستن.

علاوه بر این، می‌توانید با عبارات `and` و `or`، شرط‌های مختلفی توی بخش `where` بنویسید:

```
select name from steps
where task_id = 1 and is_completed = true;
```

دستور بالا، فقط step‌های تسك شماره ۱ که تموم شدن رو نشون‌تون می‌ده:

	name
1	Implementing the project

اضافه شدن `where` به دایرۀ لغات SQL‌تون قدرت خیلی زیادی بهتون می‌ده. مثلا می‌توانید تسكی که اسمش AP Project‌ئه رو با استفاده از `where` پیدا کنید:

```
select * from tasks
where name = 'AP Project';
```

برای مقایسه رشته‌ها توی `where`، می‌توانید از یه شرط جالب‌تر به اسم `like` هم استفاده کنید. مثلا دستور بالا رو می‌شه به شکل زیر هم نوشت:

```
select * from tasks
where name like 'AP Project';
```

توی این حالت، استفاده از `like` فرق خاصی با `=` نداره. اما خوبی `like` توی اینه که می‌تونه وجود pattern‌های مختلف رو هم توی رشته‌ها بررسی کنه. مثلا اگر بخوايد تمام task‌هایی که توی اسم آون‌ها عبارت Project او مده رو `select` کنید می‌توانید از دستور زیر استفاده کنید:

```
select * from tasks
where name like '%Project%';
```

خروجی این دستور به شکل زیره:

	id	name	due_date
1	1	AP Project	2025-12-12
2	4	DS Project	2025-12-12

پترن '`%Project%`' توی خودش دوتا % داره. این دو %، وقتی در کنار `like` استفاده می‌شن، به SQL می‌گن که «مهم نیست قبل یا بعد Project چیه. اگر توی رشته‌ای Project دیدی، نشونش بده».

پاک کردن رکوردها

دستور زیر، تمام رکوردهای جدول steps را پاک می‌کنه:

```
delete from steps;
```

صبر کنید! اجراش نکنید که دستور خطرناکیه. انقدر خطرناک که اگر بخوايد اجراش کنید DataGrip سرتون داد می‌زنه:



علتش اینه که توی دنیای واقعی، هیچ برنامه‌نویسی نمی‌خواهد اونی باشه که کل اطلاعات دیتابیس شرکتش رو پاک می‌کنه!



به جای این کار، با اضافه کردن یه where به دستورتون می‌تونید از sql بخوايد که فقط بعضی stepها رو پاک کنه. مثلا اگر الان روی select با جدول زیر مواجه می‌شید:

	id	task_id	name	is_completed
1	7	1	Reading the doc	0
2	8	1	Implementing the project	1
3	9	2	Learning about db	0

می‌تونید از sql بخوايد که فقط stepهای تسك دوم رو پاک کنه:

```
delete from steps
where task_id = 2;
```

اگر این دستور رو اجرا کنید و بعد select بزنید می‌بینید که دیگه هیچ step ای برای این task وجود نداره:

	id	task_id	name	is_completed
1	7	1	Reading the doc	0
2	8	1	Implementing the project	1

هر دستوری توی where برای select هاتون می‌زدین اینجا هم می‌تونید بزنید. مثلاً and، or، دستورات مقایسه‌ای، like و غیره همگی و همگی اینجا هم کار می‌کنن.

یه چیزی رو لازمه اینجا بدونید. می‌بینید که ما توی جدول steps دو رکورد برای task شماره ۱ داریم. چی می‌شه اگر بخوایم این task رو پاک کنیم؟

```
delete from tasks
where id = 1;
```

اگر دستور بالا رو اجرا کنید، با خطای زیر مواجه می‌شید (که بازم توی لپتاپ من خوب نمی‌افته!):



این خطا بهتون می‌گه که این task چندتا step داره، و اول باید اونها رو پاک کنید تا بتونید این task رو پاک کنید. اگر دستور زیر رو اجرا کنید:

```
delete from steps
where task_id = 1;
```

بعدش می‌تونید با خیال راحت، task شماره ۱ رو پاک کنید.

تغییر ستون‌ها

یادتون می‌یاد وقتی می‌خواستیم جدول‌های step و table رو عوض کنیم، به کل اونها رو drop کردیم؟ با drop کردن این دو جدول، تمام داده‌های اونها هم از دیتابیس پاک شد. اگر راستش رو بخوايد مشتری‌هاتون خیلی خوشحال نمی‌شن اگر با هر تغییر توی table‌های دیتابیس‌تون، دیاشون رو کامل پاک کنید!

برای یادگرفتن این تغییرات، بیاین جدول employees که توی بخش اول راجع بهش صحبت کردیم رو به دیتابیس مون اضافه کنیم. بعد از اون، ستون employee_id هم به جدول tasks اضافه می‌کنیم.

اول فایل employees.sql رو درست کنید و کوئری ایجاد جدول employees رو بزنید:

```
create table employees(
    id int primary key auto_increment,
    name nvarchar(255) not null,
    national_id nvarchar(10) not null
);
```

خب، این کوئری رو اجرا کنید تا جدولتون ساخته بشه. حالا وقتشه که ستون employee_id رو به جدول tasks اضافه کنیم. برای این کار، از خانواده دستورات alter table استفاده می‌کنیم. دستور زیر رو به tasks.sql اضافه کنید:

```
alter table tasks
add column employee_id int;
```

دستور بالا، برای ایجاد تغییر توی جدول tasks نوشته شده. این دستور، ستون employee_id از جنس int رو به دیتابیس مون اضافه می‌کنه. دقیق کنید که این ستون، null نه.

اگر الان روی tasks کوئری select بزنید با خروجی زیر مواجه می‌شید:

	id	name	due_date	employee_id
1	1	AP Project	2025-12-12	<null>
2	2	DB Homework	2025-12-12	<null>
3	4	DS Project	2025-12-12	<null>

همون طور که می‌بینید، ستون employee_id به جدول مون اضافه شده و توی همه رکوردهای قبلی مقدار null گرفته.¹³

از اون جایی که این ستون، foreign key لازمه که با یه alter table دیگه این رو به mysql بگیم. دستور زیر رو به فایل tasks.sql اضافه کنید:

```
alter table tasks
add constraint foreign key (employee_id) references employees(id);
```

¹³ فرض کنید که ستون جدیدتون، not null بود. تلاش کنید یه ستون not null به یکی از جدول هاتون اضافه کنید. به چه مشکلی می‌خورید؟ آیا می‌تونید با بررسی توی اینترنت این مشکل رو برطرف کنید؟

با این کوئری، محدودیت‌های مربوط به ستون‌های foreign key رو هم به ستون employee_id اضافه می‌کنیم.

بیاین این ستون جدید رو تست کنیم. اول یه employee به دیتابیس مون اضافه می‌کنیم:

```
insert into employees(name, national_id)
values ('Raees', '1234567890');
```

اگر الان روی جدول select employee بزنید همچین چیزی می‌بینید:

id	name	national_id
1	Raees	1234567890

حالا یه تسك جدید برای این employee تعريف می‌کنیم:

```
insert into tasks(name, due_date, employee_id)
values ('Riasat', '2099-12-12', 1);
```

اگر الان روی جدول select tasks بزنید می‌توانید رکورد جدید مون رو ببینید:

id	name	due_date	employee_id
1	AP Project	2025-12-12	<null>
2	DB Homework	2025-12-12	<null>
3	DS Project	2025-12-12	<null>
4	Riasat	2099-12-12	1

حالا، صرفاً جهت یادگیری، بیاین یکی از ستون‌های جدول‌های مون رو پاک کنیم. قربانی مون، ستون بی‌آزار is_completed. برای پاک کردن یه ستون از دستور زیر استفاده می‌کنیم:

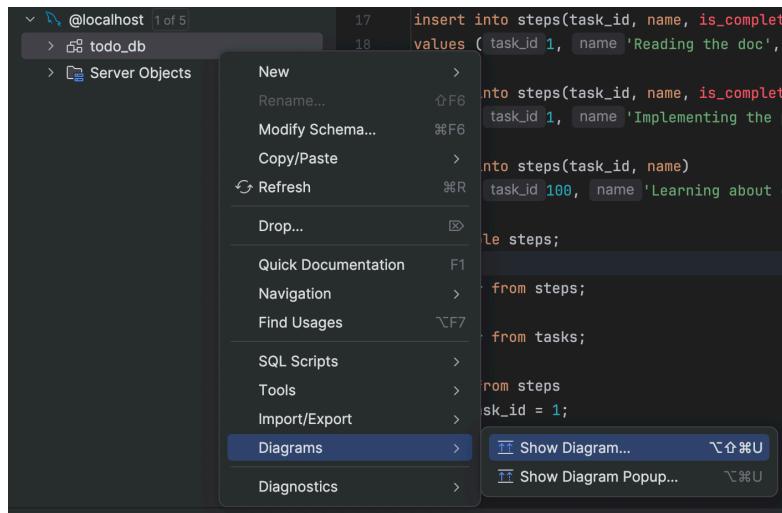
```
alter table steps
drop column is_completed;
```

بعد از این دستور، دیگه ستون is_completed توان select نمی‌شه:

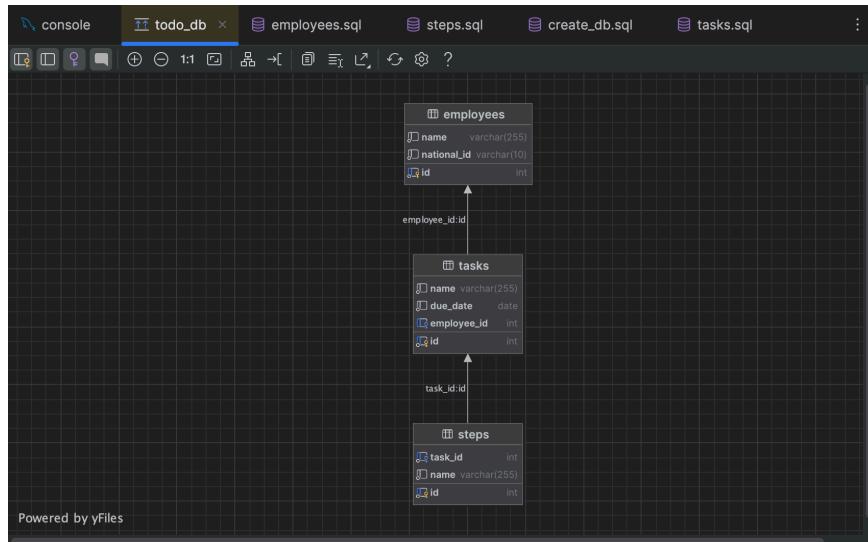
id	task_id	name
1	7	Reading the doc
2	8	Implementing the project

مشاهده دیاگرام

حالا که دیتابیس‌تون کامل شده، روی اسمش توی سمت چپ صفحه‌تون راست کلیک کنید، رو بزنید و روی Show Diagram کلیک کنید:



می‌بینید که یه دیاگرام ساده و خلاصه، شامل تمام اطلاعات جداویل‌تون دیده می‌شه!



این دیاگرام الان خیلی ساده‌ست. ولی برای پروژه‌های بزرگتر، خیلی خیلی به درد بخوره.

چیزی که یاد گرفتیم

ما توی این داکیومنت، اولین قدم‌هایمان رو توی دنیای دیتابیس‌ها زدیم. قدم‌هایی که هر چند کوچیک‌تر، ولی مهم و ضروری‌ان. شما توی این داک با موارد زیر آشنا شدید:

- چطور می‌شه دیتا رو، به شکل یک سری جدول مرتب‌سازی کرد.
- چطور می‌شه کیفیت این جدول‌ها رو بالاتر برد و اون‌ها رو normalize کرد؟
- SQL چیه؟ چطور می‌شه با اون کار کرد؟

منابع بیشتر

دنیای دیتابیس‌ها، خیلی وسیعه و ما حتی اگر بخوایم هم نمی‌تونیم کل این رو پوشش بدیم. انقدر دیتابیس‌ها دنیای بزرگ و گستردگی دارن که database engineering یه عنوان شغلی خیلی خفن توی دنیای برنامه‌نویسیه.

برای این که با این دنیا بیشتر آشنا شید، می‌توانید از [سایت w3schools](#) استفاده کنید. همچنین [курس جمع و جور ماش همدانی](#) هم می‌توانه خیلی بهتون کمک کنه:

