قواعد البيانات

تاريخ التحديث: يونيو 9, 2024

لغة SQL الدليل الشامل الأول

يأخذك هذا الدليل في رحلة تعريفية إلى عالم لغة SQL، المستخدمة لإدارة وتنظيم البيانات في قواعد البيانات. هذا الشرح مناسب للمبتدئين ويشرح بلغة سهلة ومبسطة كيف يمكن استخدام SQL لإنشاء واستعلام وتحديث البيانات داخل قواعد البيانات.

اسم الدليل: لغة SQL

رقم الدليل: 300

المجال: قواعد البيانات SQL

المتطلبات: لا يوجد

مسار تعلم متعلق بهذا الدليل: -

قائمة فيديوهات الدليل

تنزيل الدليل كملف الكتروني PDF

تاريخ تحديث الدليل: أغسطس 11, 2024

ما هي SQL؟

SQL أو **Structured Query Language** هي لغة معيارية تُستخدم لإدارة قواعد البيانات العلائقية. تُمكِّنك من إجراء عمليات متعددة مثل الاستعلام عن البيانات، تحديثها، إدخالها، وحذفها من الجداول.

شاهد مقدمة حول هذا الدليل كفيديو تعليمي: رابط الفيديو

ماقبل أساسيات لغة SQL

هذه الأمور عليك أن تعرفها قبل البدء بتعلم لغة SQL

ما هي قواعد البيانات؟

قاعدة البيانات هي مجموعة منظمة من البيانات يمكن الوصول إليها وإدارتها وتحديثها بسهولة. تُستخدم قواعد البيانات لتخزين المعلومات بطريقة تجعلها سهلة البحث والفرز والاسترجاع.

مثال بسيط

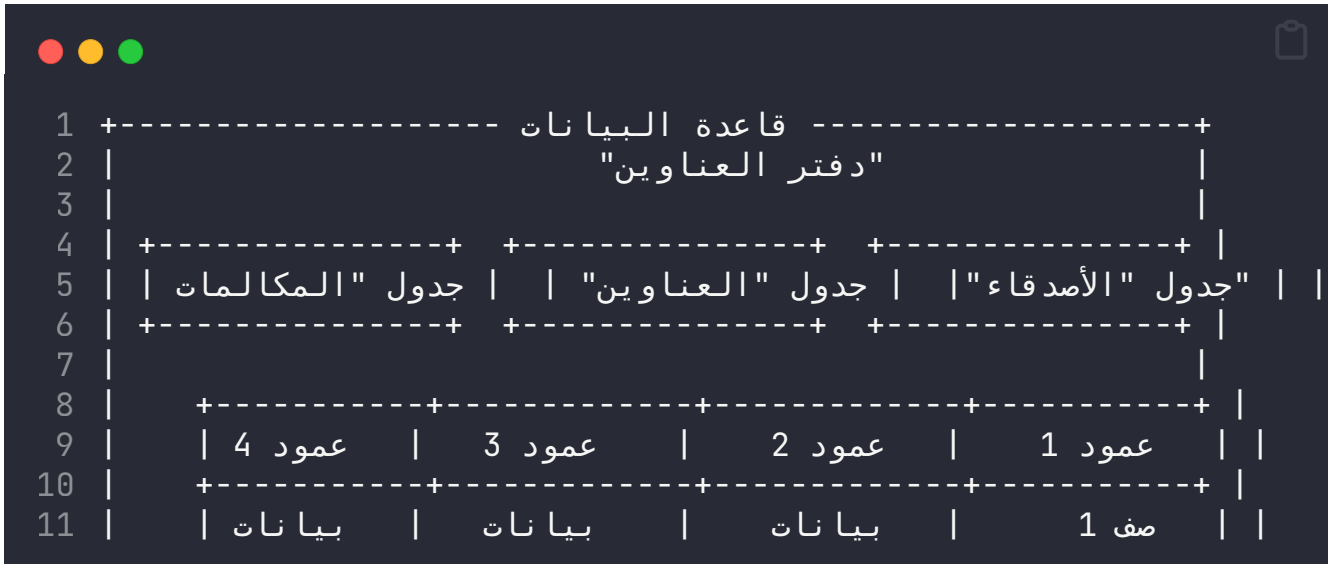
الاسم الأول	الاسم الأخير	رقم الهاتف	البريد الإلكتروني
علي	الأحمد	123456789	ali@example.com
سارة	أحمد	987654321	sara@example.com
محمد	الطيب	456123789	mohamed@example.com

جدول بيانات الأصدقاء

- **اسم الجدول**: "الأصدقاء".
- **الصفوف**: كل صف يمثل سجلاً (Record) أو مدخل بيانات. في المثال، الصف الأول يحتوي على معلومات صديق يدعى "علي الأحمد".
- **الأعمدة**: كل عمود يمثل نوعًا معينًا من البيانات. في المثال، الأعمدة هي "الاسم الأول"، "الاسم الأخير"، "رقم الهاتف"، و"البريد الإلكتروني".

باختصار، قاعدة البيانات هي مثل نظام تصنيف للمعلومات حيث يتم تخزين المعلومات في جداول (مثل ملفات على الرف)، وكل جدول يتكون من أعمدة (مثل التصنيفات)، وصفوف (مثل الأوراق داخل الملفات) لتخزين البيانات المنظمة.

توضيح بصري بسيط:



```
| | صف 2 | بيانات | بيانات | بيانات | بيانات |
+-----------+-----------+-----------+-----------+
+-----------------------------------------------------+
```

Plaintext

ما هي الأعمدة؟

الأعمدة هي مكونات الجدول التي تمثل خصائص معينة أو نوعًا معينًا من البيانات. كل عمود في الجدول له اسم ويخزن نوعًا معينًا من المعلومات.

مثال بسيط

في جدول "الأصدقاء" أعلاه:

- **العمود "الاسم الأول"**: يحتوي على الأسماء الأولى للأصدقاء.
- **العمود "الاسم الأخير"**: يحتوي على الأسماء الأخيرة للأصدقاء.
- **العمود "رقم الهاتف"**: يحتوي على أرقام هواتف الأصدقاء.
- **العمود "البريد الإلكتروني"**: يحتوي على عناوين البريد الإلكتروني للأصدقاء.

ما هي الصفوف؟

الصفوف هي مكونات الجدول التي تحتوي على البيانات الفعلية. كل صف يمثل سجلًا (Record) واحدًا ويخزن مجموعة من القيم في كل عمود من أعمدة الجدول.

مثال بسيط

في جدول "الأصدقاء" أعلاه، كل صف يحتوي على معلومات عن صديق واحد. الصف الأول يحتوي على بيانات "علي الأحمد"، بما في ذلك الاسم الأول، الاسم الأخير، رقم الهاتف، والبريد الإلكتروني.

الملخص

تخيل قاعدة بيانات كحافظة ملفات كبيرة:

- **قاعدة البيانات**: الحافظة.
- **الجدول**: كل مجلد في الحافظة يمثل جدولًا. مثلا، مجلد يحتوي على معلومات الأصدقاء.
- **الأعمدة**: الفئات في كل مجلد مثل "الاسم الأول"، "الاسم الأخير"، إلخ.
- **الصفوف**: كل ورقة داخل المجلد تحتوي على المعلومات عن شخص واحد.

الحاجة للغة استعلام

قواعد البيانات الحديثة غالبًا ما تكون معقدة وتتضمن العديد من الجداول والعلاقات بينها. هذا يعني أنه من الضروري أن يكون هناك واجهة قوية وفعالة لإدارة وتحليل هذه البيانات بكفاءة، وهنا تأتي دور لغة SQL.

لغة SQL تقدم مجموعة من الأدوات والأوامر التي تسمح للمستخدمين بالتعامل مع قواعد البيانات بسهولة وفعالية. من خلال SQL، يمكن للمستخدمين القيام بالعديد من الأنشطة مثل:

1. استعلام البيانات: باستخدام عبارات SELECT يمكن للمستخدمين استعلام البيانات من مجموعة متنوعة من الجداول باستخدام مجموعة متنوعة من الشروط والمعايير.
2. تحديث وحذف البيانات: يمكن للمستخدمين استخدام عبارات UPDATE و DELETE لتحديث البيانات الموجودة بالفعل في الجداول أو حذفها.
3. إدخال البيانات الجديدة: يمكن للمستخدمين استخدام عبارة INSERT لإدخال بيانات جديدة إلى الجداول.
4. إدارة هيكل قاعدة البيانات: يمكن للمستخدمين إنشاء وتعديل وحذف الجداول والفهارس والقيود باستخدام عبارات DDL (Data Definition Language).

بفضل هذه الأدوات والأوامر، يمكن للمستخدمين إدارة وتحليل البيانات في قواعد البيانات بكفاءة، بغض النظر عن حجمها أو تعقيدها. وبذلك، تكون لغة SQL أداة قوية وحيوية في مجال إدارة البيانات الحديثة.

كيف تتبع هذا الدليل؟

سيتم توضيح كل التعليمات من خلال تطبيق عملي.

> سنقوم بتنفيذ كود وتعليمات SQL على موقع:
> https://www.db-fiddle.com

1. اذهب لرابط الموقع: https://www.db-fiddle.com
2. سيكون هنالك مربعان: الأول خاص بإضافة جداول قاعدة البيانات تحت اسم schema SQL
3. المربع الثاني: يتم اضافة فيه الاستعلامات تحت اسم Query SQL
4. اكتب أوامر التمرينات في المربع الذي يحدده التمرين، ثم قم بالنقر على تنفيذ RUN موجودة في الأعلى.
5. سأقوم بمشاركة رابط الكود الذي أكتبه في كل مرة معكم، وسيتم اضافة الكود لنفس الموقع السابق.
6. يمكنك مقارنة الكود الذي تكتبه مع الكود الموجود في التمرين لتتأكد من صحة الكود الذي كتبته أو تتبع الأخطاء
7. لا تيأس إذا وجدت صعوبة بفهم فكرة أو تمرين ما، فالعقل يحتاج وقت للاعتياد على المعلومات الجديدة.
8. استمر! حاول! استمر!

ماذا تعني Schema SQL؟

هنا سيتم اضافة قاعدة البيانات نفسها أي بمعنى آخر البيانات التي لدينا مجمعة بجداول ومعبر عنها بصفوف وجداول.

هذا هو من أهم مزايا SQL، أنها تسمح لنا بتنظيم البيانات بجداول التي بدورها تحتوي على أعمدة وصفوف.

ماذا تعني SQL Query ؟

بعد إنشاء وتنظيم البيانات بقاعدة البيانات، سيكون هنالك حاجة لطلب هذه البيانات.

اذا كانت هذه بيانات الأصدقاء كما في المثال في القسم الأول، فالآن نريد أن نحصل على رقم هاتف سارة، ففي هذه الحالة يتم القيام بعملية استعلام أو استرجاع أو ما تسمى باللغة الانكليزية Query.

فعندما نريد استرجاع أو الاستعلام عن معلومة ما سواء كان هذا الاستعلام معقد أو بسيط، فهذه العملية تسمى Query

ما هذا الموقع db-fiddle.com

للأسف، عملية تثبيت أدوات قواعد البيانات على جهاز الحاسوب قد تكون عملية صعبة للمبتدئين.

ولذلك، وجدت عن موقع يساعدنا على التدرب على أوامر لغة SQL دون الحاجة فعلياً لتثبيت أي أداة على جهاز الحاسوب.

هذا سيسهل عملية التعلم ولاحقاً يمكن أن تتعلم كيفية تثبيت لغة SQL على جهازك.

لا، هذا ليس إعلان أو مقالة دعائية لموقع db-fiddle.com. وجدت أنه موقع مفيد ومجاني، وأحببت مشاركة الفائدة معكم.

أساسيات لغة SQL

هنا سيتم توضيح أهم الأوامر التي عليك تعلمها في لغة SQL.

سيتم توضيح الأوامر الأساسية بالتدريبات العملية

تدريب 301: إنشاء جدول وإدخال بيانات والاستعلام عنها.

شاهد تطبيق التمرين كفيديو تعليمي: لغة SQL الدليل الشامل الأول - التمرين 301

تفاصيل التمرين

1. المطلوب: قم بإنشاء جدول في قاعدة بيانات يحتوي معلومات الأصدقاء في الجدول أدناه.
2. رابط الأمر النهائي - الخطوة الأولى: تمرين 301 - الكود النهائي 1
3. رابط الأمر النهائي - الخطوة الثانية: تمرين 301 - الكود النهائي 2
4. رابط الأمر النهائي - الخطوة الثالثة: تمرين 301 - الكود النهائي 3

الاسم الأول	الاسم الأخير	رقم الهاتف	البريد الإلكتروني
علي	الأحمد	123456789	ali@example.com
سارة	أحمد	987654321	sara@example.com
محمد	الطيب	456123789	mohamed@example.com

نعتبر أن لدينا قاعدة بيانات منشئة وجاهزة، ونبدأ من نقطة إنشاء جداول.

الخطوة الأولى

الحل مع التوضيح

نكتب الكود الآتي في موقع db-fiddle.com في قسم Schema

```sql
CREATE TABLE Friends (
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    PhoneNumber VARCHAR(15),
    Email VARCHAR(100)
);

```

هذا الكود يقوم بإنشاء جدول في قاعدة البيانات يُسمى "Friends"، وهذا الجدول يحتوي على أربعة أعمدة:

1. FirstName (الاسم الأول): هذه العمود يحتوي على الأسماء الأولى لأصدقائك.
2. LastName (الاسم الأخير): هذه العمود يحتوي على الأسماء الأخيرة لأصدقائك.
3. PhoneNumber (رقم الهاتف): هذه العمود يحتوي على أرقام هواتف أصدقائك.
4. Email (البريد الإلكتروني): هذه العمود يحتوي على عناوين البريد الإلكتروني لأصدقائك.

كل عمود له نوع محدد للبيانات التي يمكن تخزينها فيه. في هذا الجدول، يتم استخدام أنواع بيانات "VARCHAR"، وهي تعني سلسلة محارف بعدد معين. على سبيل المثال، "FirstName" و "LastName" يمكنهما أن يحتويا على نص يصل إلى 50 محرفاً، و "PhoneNumber" يمكنه أن يحتوي على نص يصل إلى 15 محرفاً، و "Email" يمكنه أن يحتوي على نص يصل إلى 100 محرفاً.

ملاحظات في دفترك

```SQL
يتم إنشاء الجدول من خلال الأمر
Create Table
```

varchar يعبر أن قيمة العمود قد تكون بين 0 لقيمة قصوى يتم تحديدها. مثلاً:
varchar(50) يعني أن قيمة العمود لا يمكن أن تكون أكثر من 50 محرف (حرف أو رقم أو رمز خاص)

سؤال: هل يجب أن أكتب كل كلمات الأمر بأحرف كبيرة (upper case)؟

لا، في SQL، ليس من الضروري كتابة الأوامر أو الكلمات الرئيسية بأحرف كبيرة. يعمل الكود بشكل متماثل سواء كتبت الكلمات بأحرف كبيرة أو صغيرة. عادةً ما يكتب الأشخاص بعض الكلمات بأحرف كبيرة لتسهيل قراءة الكود وتمييز الكلمات الرئيسية من الأسماء العادية في الجداول والأعمدة. ومع ذلك، يمكن استخدام أحرف صغيرة أيضًا دون مشاكل في تنفيذ الأوامر في SQL.

الخطوة الثانية

الحل مع التوضيح

نكتب الكود الآتي في موقع db-fiddle.com في قسم Schema

```SQL
CREATE TABLE Friends (
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    PhoneNumber VARCHAR(15),
    Email VARCHAR(100)
);

INSERT INTO Friends (FirstName, LastName, PhoneNumber, Email)
```

```sql
VALUES
    ('Ali', 'Al-Ahmad', '123456789', 'ali@example.com'),
    ('Sara', 'Ahmad', '987654321', 'sara@example.com'),
    ('Mohamed', 'Al-Tayeb', '456123789', 'mohamed@example.com');


```

هذا الكود هو أمر INSERT في SQL، ويتم استخدامه لإضافة سجلات جديدة (صفوف) إلى جدول. دعني أقدم لك تفسيرًا بسيطًا للكود:

1. **INSERT INTO Friends**: هذا الجزء يحدد اسم الجدول الذي نريد إدراج السجلات فيه، وهو "Friends". سنقوم بإضافة السجلات إلى جدول "Friends".
2. **(FirstName, LastName, PhoneNumber, Email)**: هذا الجزء يحدد الأعمدة التي نريد إدراج البيانات فيها. تُسمى الأعمدة FirstName، LastName، PhoneNumber وEmail.
3. **VALUES**: هذه الكلمة الرئيسية تُستخدم لتحديد القيم التي نريد إدراجها في الأعمدة المحددة.
4. **('علي', 'الأحمد', '123456789', 'ali@example.com')**: هذه هي مجموعة القيم الأولى التي سنقوم بإدراجها في الجدول. كل مجموعة من القيم تتوافق مع صف واحد في الجدول. القيم تُقدم في نفس ترتيب الأعمدة المحددة سابقًا. لذا، 'علي' تتوافق مع FirstName، 'الأحمد' تتوافق مع LastName، '123456789' تتوافق مع PhoneNumber، و'ali@example.com' تتوافق مع Email.
5. **('سارة', 'أحمد', '987654321', 'sara@example.com')**: هذه هي مجموعة القيم الثانية التي سنقوم بإدراجها. مرة أخرى، كل قيمة تتوافق مع العمود المقابل لها بنفس الترتيب.
6. **('محمد', 'الطيب', '456123789', 'mohamed@example.com')**: هذه هي مجموعة القيم الثالثة التي سنقوم بإدراجها، وتتبع نفس النمط.

لذا، يقوم هذا الكود SQL بإدراج ثلاثة سجلات جديدة في جدول "Friends"، حيث يحتوي كل سجل على اسم،

ملاحظات في دفترك

```sql
اضافة سجلات للجدول
INSERT INTO اسم الجدول
```

يجب تحديد اسماء الاعمدة التي تريد اضافة قيم لها، ووضعها ضمن أقواس عادية، وتفصل بينها بفاصلة.

سؤال: هل استخدم اشارات اقتباس أحادية ' أو ازدواجية "؟

استخدم اشارات الاقتباس الأحادية للقيم النصية لتفادي أي التباس. ستتعلم أكثر عن هذا في المستقبل.

لمزيد من المعلومات:
شاهد فيديو: لن تخطأ باستخدام ‘ , ” في SQL بعد الآن.

الخطوة الثالثة

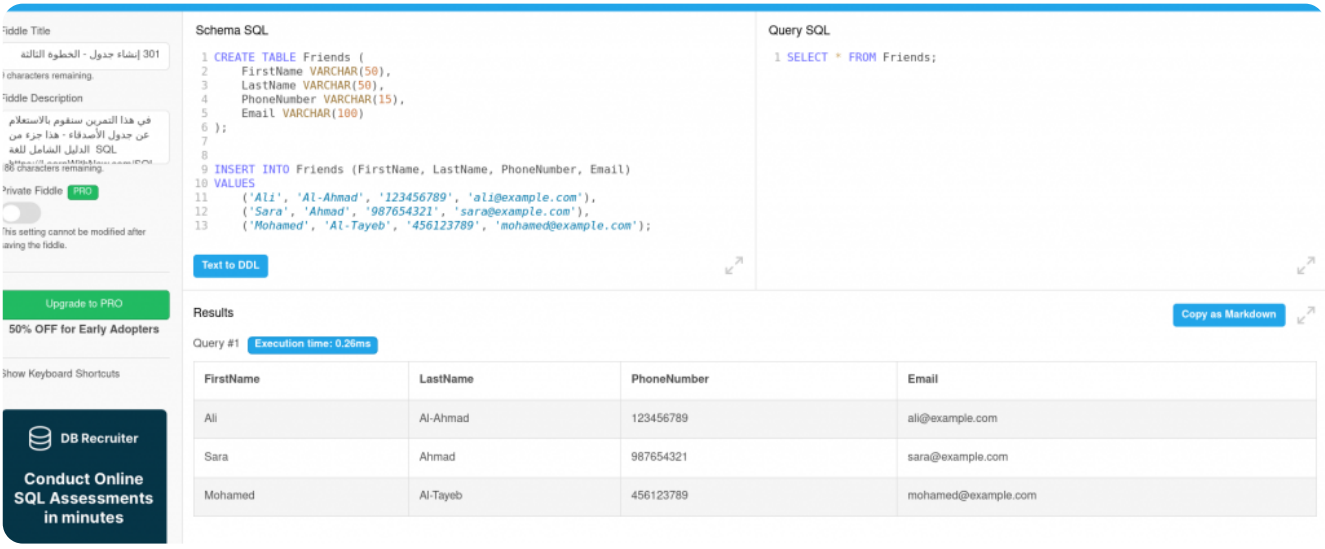
الحل مع التوضيح

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
SELECT * FROM Friends;
```

يقوم هذا الأمر بالاستعلام لاسترجاع جميع البيانات المخزنة في جدول معين. دعنا نوضحه بشكل مبسط:

- **SELECT**: تعني “اختر”. تُستخدم لتحديد الأعمدة التي تريد استرجاعها من قاعدة البيانات.
- *: تعني “كل الأعمدة”. عندما نستخدم *، فنحن نطلب استرجاع جميع الأعمدة الموجودة في الجدول.
- **FROM**: تعني “من”. تُستخدم لتحديد الجدول الذي نريد استرجاع البيانات منه.
- **Friends**: اسم الجدول الذي نريد استرجاع البيانات منه. في هذا المثال، الجدول اسمه “Friends”.



نتيجة الاستعلام عن الأصدقاء

ملاحظات في دفترك

```sql
الاستعلام عن السجلات
SELECT
```

سؤال: ماذا تعني *؟

في SQL، الرمز * يُعرف باسم "علامة النجمة" أو "النجمة"، وهو يستخدم لاسترجاع جميع الأعمدة من الجدول المحدد.

تدريب 302: إنشاء جدول الأفلام والاستعلام عنه

تفاصيل التمرين

1. المطلوب: الاستعلام عن الأفلام من جدول الأفلام.
2. يمكنك محاولة إنشاء الجدول الآتي وإدخال البيانات فيه.
3. أو يمكنك الاكتفاء بقراءة أمر إنشاء الجدول وإدخال بيانات فيه.
4. والمواصلة لتمرين الاستعلام والتصفية.
5. رابط الأمر النهائي - الخطوة الأولى: تمرين 302 - الكود النهائي 1
6. رابط الأمر النهائي - الخطوة الثانية: تمرين 302 - الكود النهائي 2
7. رابط الأمر النهائي - الخطوة الثالثة: تمرين 302 - الكود النهائي 3

المعرّف	عنوان الفيلم	النوع	سنة الإصدار	التقييم
1	Inception	خيال علمي	2010	8.8
2	The Godfather	جريمة	1972	9.2
3	The Dark Knight	أكشن	2008	9.0
4	Pulp Fiction	جريمة	1994	8.9
5	The Shawshank Redemption	دراما	1994	9.3
6	The Matrix	خيال علمي	1999	8.7
7	Forrest Gump	دراما	1994	8.8
8	The Avengers	أكشن	2012	8.0
9	Gladiator	أكشن	2000	8.5
10	Titanic	رومانسية	1997	7.8

جدول الأفلام

نعتبر أن لدينا قاعدة بيانات منشئة وجاهزة، ونبدأ من نقطة إنشاء جداول.

شاهد تطبيق التمرين كفيديو تعليمي: لغة SQL الدليل الشامل الأول - التمرين 302

الخطوة الأولى

الحل مع التوضيح

نكتب الكود الآتي في موقع db-fiddle.com في قسم Schema

```sql
-- إنشاء جدول باسم Movies
CREATE TABLE Movies (
    ID INT PRIMARY KEY,                -- المعرف الفريد للفيلم
    Title VARCHAR(255),                -- عنوان الفيلم
    Genre VARCHAR(50),                 -- نوع الفيلم
    ReleaseYear INT,                   -- سنة إصدار الفيلم
    Rating DECIMAL(3, 1)               -- (نظام النقاط حتى عشري واحد)
);
```

1. **`CREATE TABLE Movies` :)**
   - هذا الجزء يعني أننا نقوم بإنشاء جدول جديد بإسم "Movies".
2. **`ID INT PRIMARY KEY`:**
   - يعرّف عمودًا بإسم "ID" من نوع INT (أي عدد صحيح)، ويتم تعيينه كمفتاح أساسي PRIMARY KEY للجدول. الفهرس الأساسي يضمن أن قيمة كل سجل في هذا العمود فريدة ولا تتكرر، وهو يُستخدم عادة لتعريف كيفية تمثيل كل سجل بشكل فريد.
3. **`Title VARCHAR(255)`:**
   - يعرّف عمودًا بإسم "Title" من نوع VARCHAR، وهو نوع بيانات يمكنه تخزين سلسلة نصية بحد أقصى يصل إلى 255 حرفًا. يُستخدم هنا لتخزين عناوين الأفلام.
4. **`Genre VARCHAR(50)`:**
   - يعرّف عمودًا بإسم "Genre" من نوع VARCHAR، يمكنه تخزين سلسلة نصية بحد أقصى يصل إلى 50 حرفًا. يُستخدم هذا العمود لتخزين نوع أو تصنيف الفيلم مثل "Action" أو "Comedy".

5. **`ReleaseYear INT`:**
   - يعرّف عمودًا بإسم "ReleaseYear" من نوع INT (أي عدد صحيح). يُستخدم لتخزين سنة إصدار الفيلم.
6. **`Rating DECIMAL(3, 1)`:**
   - يعرّف عمودًا بإسم "Rating" من نوع DECIMAL، حيث يمثل التقييم الخاص بالفيلم. الـ DECIMAL(3, 1) يعني أن التقييم يمكن أن يحتوي على حد أقصى 3 أرقام كاملة مع رقم عشري واحد، مما يسمح بتخزين تقييمات مثل 8.5 أو 9.0.

بهذا الشكل، يتم إنشاء جدول "Movies" الذي يحتوي على خمسة أعمدة مختلفة تمثل معلومات مختلفة عن الأفلام مثل العنوان، النوع، سنة الإصدار، والتقييم.

ملاحظات في دفترك

```SQL
CREATE TABLE Movies (
    ID INT PRIMARY KEY,
```

في قواعد قواعد البيانات، مفتاح Primary Key (المفتاح الأساسي أو المفتاح الرئيسي) يُستخدم لتحديد بشكل فريد كل سجل في جدول قاعدة البيانات. هذا المفتاح له أهمية كبيرة لأنه يوفر وسيلة فعالة للتعرف على سجلات معينة ومنع اختلاط سجلات مع سجلات أخرى. حيث يكون لكل سجل معرّف خاص مميز **ولا يُمكن أبداً لأي سجلين أن يكون لهما نفس المعرّف.**

```SQL
CREATE TABLE Movies (
    ------
    ReleaseYear INT,

```

INT هو نوع بيانات يُستخدم لتخزين الأرقام الصحيحة (بدون أي أرقام عشرية)، وهو جزء من أنواع البيانات التي تُستخدم لتعريف حقول الجدول. تعريف الحقل `ReleaseYear INT` يعني أن هذا الحقل سيحتوي على قيم صحيحة تمثل سنة الإصدار.

```SQL
CREATE TABLE Movies (
    ------
    Rating DECIMAL(3, 1)

```

`DECIMAL(3, 1)` في SQL يعني أنه حقل يمكنه تخزين أرقام عشرية بدقة تصل إلى 3 أرقام كليًا، مع 1 رقم بعد الفاصلة العشرية. على سبيل المثال، يمكن تخزين القيم مثل `123.4` أو `-7.8`، ولكن لا يمكن تخزين القيم مثل `12.345` لأنها تتجاوز الدقة المحددة للأرقام العشرية.

? سؤال: ماذا يعني "- -" في المثال السابق؟

هذه العبارة `- -` تُستخدم لبدء تعليق في سطر واحد في SQL. ويكون أي شيء بعد `- -` حتى نهاية السطر تعليقًا ولا يتم تنفيذه كجزء من الاستعلام.
التعليق في البرمجة يعني كتابة ملاحظة أو توضيح حول الأمر، وهذا فقط لغايات التوضيح والتنظيم ولا ينفذ أبداً.

الخطوة الثانية

الحل مع التوضيح

نكتب الكود الآتي في موقع db-fiddle.com في قسم Schema

```sql
-- إنشاء جدول باسم Movies
CREATE TABLE Movies (
    ID INT PRIMARY KEY,                -- المعرف الفريد للفيلم
    Title VARCHAR(255),                -- عنوان الفيلم
    Genre VARCHAR(50),                 -- نوع الفيلم
    ReleaseYear INT,                   -- سنة إصدار الفيلم
    Rating DECIMAL(3, 1)               -- (نظام النقاط حتى عشري واحد)
);


-- إدراج بيانات عينة في الجدول Movies
INSERT INTO Movies (ID, Title, Genre, ReleaseYear, Rating) VALUES
(1, 'Inception', 'Sci-Fi', 2010, 8.8),
(2, 'The Godfather', 'Crime', 1972, 9.2),
(3, 'The Dark Knight', 'Action', 2008, 9.0),
(4, 'Pulp Fiction', 'Crime', 1994, 8.9),
(5, 'The Shawshank Redemption', 'Drama', 1994, 9.3),
(6, 'The Matrix', 'Sci-Fi', 1999, 8.7),
(7, 'Forrest Gump', 'Drama', 1994, 8.8),
(8, 'The Avengers', 'Action', 2012, 8.0),
(9, 'Gladiator', 'Action', 2000, 8.5),
```

```
22 (10, 'Titanic', 'Romance', 1997, 7.8);
23
```

SQL

هذا الكود يُقوم بإدراج بيانات عينة داخل جدول يسمى Movies في قاعدة البيانات.

الخطوة الثالثة

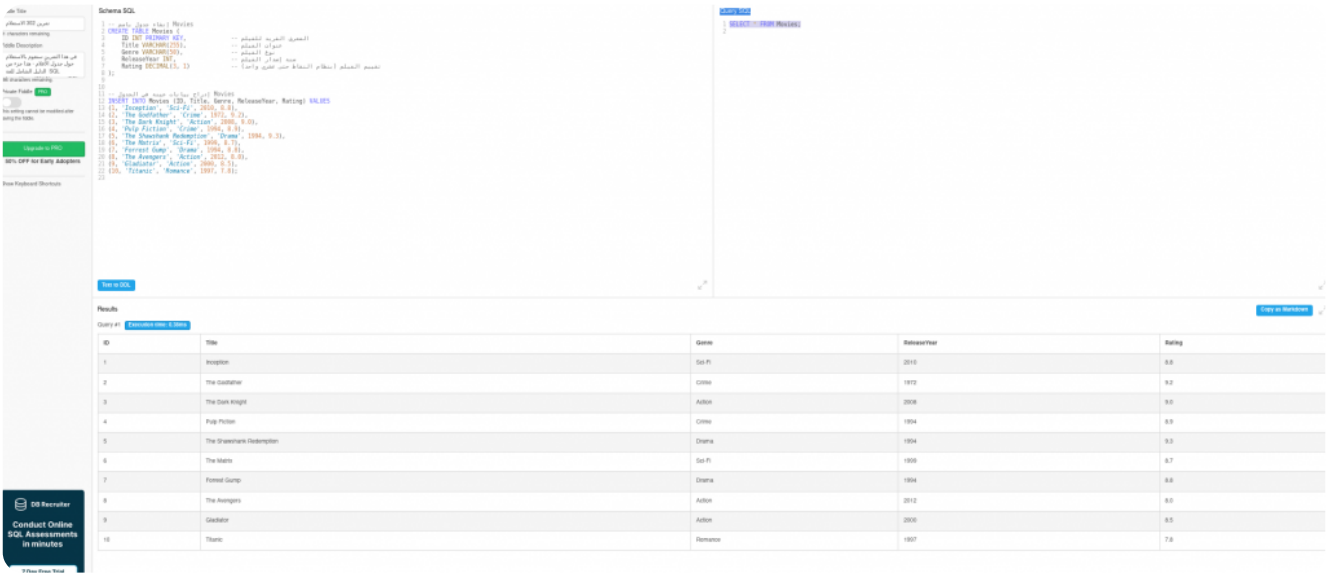
الحل مع التوضيح

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```
1 SELECT * FROM Movies;
2
```

SQL

هذا أمر استعلام بسيط، سيعرض لنا كل الأعمدة من جدول الأفلام



نتيجة الاستعلام عن الأفلام

تدريب 303: الاستعلام والتصفية.

تفاصيل التمرين

1. يجب أن تقوم بالتدريب السابق 302 قبل الاستمرار بهذا التدريب.
2. سنقوم باستخدام جدول الأفلام للاستعلام عن سجلات معينة.
3. الاستعلام الأول: قم بالاستعلام عن الأفلام التي من نوع Action فقط.
4. الاستعلام الثاني: قم بالاستعلام عن الأفلام التي أصدرت بعد عام 2000.
5. الاستعلام الثالث: الاستعلام عن الأفلام التي قيمت أقل من 8.5.
6. الاستعلام الرابع: الاستعلام عن الأفلام التي صدرت في وبعد عام 2008
7. الاستعلام الخامس: الاستعلام عن الأفلام التي لها تقييم أقل أو يساوي 8.0.
8. الاستعلام السادس: قم بالاستعلام عن الأفلام التي أصدرت بين عامي 1990 و 2000.
9. رابط الأمر النهائي تمرين 303 - الكود النهائي

المعرّف	عنوان الفيلم	النوع	سنة الإصدار	التقييم
1	Inception	خيال علمي	2010	8.8
2	The Godfather	جريمة	1972	9.2
3	The Dark Knight	أكشن	2008	9.0
4	Pulp Fiction	جريمة	1994	8.9
5	The Shawshank Redemption	دراما	1994	9.3
6	The Matrix	خيال علمي	1999	8.7
7	Forrest Gump	دراما	1994	8.8
8	The Avengers	أكشن	2012	8.0
9	Gladiator	أكشن	2000	8.5
10	Titanic	رومانسي	1997	7.8

جدول الأفلام

شاهد تطبيق التمرين كفيديو تعليمي: لغة SQL الدليل الشامل الأول - التمرين 303

WHERE =

الاستعلام عن الأفلام من نوع Action

رابط الكود النهائي: الاستعلام عن أفلام Action

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
-- استعلم عن الأفلام من نوع
-- action
SELECT * FROM Movies WHERE Genre = 'Action';
```

هذا الكود يُستخدم في SQL لاسترجاع جميع البيانات من جدول يسمى `Movies` حيث تكون قيمة العمود `Genre` تساوي `'Action'`. لنشرحه بشكل مبسط:

- **SELECT ***: يعني استرجاع كافة الأعمدة من الجدول.
- **FROM Movies**: يحدد الجدول الذي نريد استخدامه للاستعلام، وفي هذه الحالة هو جدول `Movies`.
- **WHERE Genre = 'Action'**: يعمل كـ"شرط" للبحث، حيث يقوم الاستعلام بتحديد الصفوف التي تحتوي على قيمة `Genre` تساوي `'Action'`.

باختصار، هذا الاستعلام سيُرجع جميع الأفلام التي تنتمي إلى نوع الأكشن (Action) من جدول `Movies`.

ملاحظات في دفترك

```sql
WHERE columnName = 'Value';
```

قم باستعادة السجلات التي يكون فيها قيمة عمود معين يساوي قيمة معينة. الغرض من WHERE هو فلترة وتصفية السجلات بناء على شرط معين.

WHERE >

الاستعلام عن الأفلام التي اصدرت بعد عام 2000

رابط الكود النهائي: الاستعلام عن الأفلام التي أصدرت بعد عام 2000

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
SELECT * FROM Movies WHERE ReleaseYear > 2000;
```

يقوم الأمر السابق بالتالي:

1. **SELECT * FROM Movies**: يعني استرجاع كافة الأعمدة (* تعني جميع الأعمدة) من جدول يسمى "Movies".
2. **WHERE ReleaseYear > 2000**: يُضيف شرطاً للبحث، يعني استرجاع الصفوف (السجلات) التي تحقق الشرط المحدد، وهو أن قيمة العمود "ReleaseYear" تكون أكبر من 2000.

باختصار، هذا الكود يسترجع جميع الأفلام من جدول "Movies" التي تم إصدارها بعد عام 2000.

ملاحظات في دفترك

```sql
WHERE columnName > 'Value';
```

يقوم بتحديد الصفوف التي تحتوي على قيم في عمود محدد (`columnName`) تكون أكبر من قيمة معينة (`'Value'`).

WHERE <

الاستعلام عن الأفلام التي قيمت أقل من 8.5

رابط الكود النهائي: الاستعلام عن الأفلام التي لها تقييم أقل من 8.5

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
SELECT * FROM Movies WHERE Rating < 8.5;
```

يقوم بالتالي:

1. **SELECT * FROM Movies**: يعني استرجاع كافة الأعمدة (* تعني جميع الأعمدة) من جدول يسمى "Movies".
2. **WHERE Rating < 8.5**: يضيف شرطاً للبحث، حيث يتم استرجاع الصفوف (السجلات) التي تحقق الشرط المحدد. في هذه الحالة، يتم استرجاع الأفلام التي تمتلك تقييم (Rating) أقل من 8.5.

باختصار، هذا الأمر يعني أننا نريد استرجاع كافة الأفلام من الجدول "Movies" التي لديها تقييم أقل من 8.5.

ملاحظات في دفترك

```sql
WHERE columnName < 'Value';
```

يقوم بتحديد الصفوف في جدول قاعدة البيانات حيث تكون قيمة العمود المحدد (`columnName`) أقل من القيمة المحددة (`'Value'`).

WHERE >=

الاستعلام عن الأفلام التي صدرت في وبعد عام 2008

رابط الكود النهائي: الاستعلام عن الأفلام التي صدرت في وبعد عام 2008

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
SELECT * FROM Movies WHERE ReleaseYear ⩾ 2008;
```

هذا الكود يقوم بجلب جميع السجلات من جدول الأفلام (`Movies`) حيث أنّ سنة الإصدار (`ReleaseYear`) أكبر من أو تساوي 2008. بمعنى آخر، يعيد الاستعلام جميع الأفلام التي تم إصدارها في عام 2008 أو بعد ذلك.

بشكل مبسط، يعني هذا أنه سيتم استرجاع كل البيانات من الأفلام التي صدرت في السنوات 2008 وبعد ذلك، وذلك باستخدام الأمر `SELECT * ` لجميع الأعمدة (*) من الجدول `Movies`، والشرط `WHERE ReleaseYear >= 2008` لتحديد السجلات التي تنطبق عليها الشرط.

ملاحظات في دفترك

```sql
WHERE columnName ⩾ 'Value';
```

باستخدام هذا الشرط، يتم اختيار الصفوف حيث تكون قيمة العمود (columnName) أكبر من أو تساوي القيمة ('Value') المحددة. ويتم استخدام هذا النوع من الشروط في العديد من الاستعلامات لتحديد البيانات التي تتوافق مع معايير معينة بناءً على القيم في الجداول.

WHERE <=

الاستعلام عن الأفلام التي لها تقييم أقل أو يساوي 8.0

رابط الكود النهائي: الاستعلام عن الأفلام التي لها تقييم أقل أو يساوي 8.0

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
SELECT * FROM Movies WHERE Rating ⩽ 8.0
```

1. **SELECT * FROM Movies**: يعني استرجاع كافة الأعمدة (* تعني جميع الأعمدة) من جدول يسمى "Movies".
2. **WHERE Rating <= 8.0**: يُضيف شرطاً للبحث، حيث يتم استرجاع الصفوف (السجلات) التي تحقق الشرط المحدد. في هذه الحالة، يتم استرجاع الأفلام التي يكون تقييمها (Rating) أقل من أو تساوي 8.0.

باختصار، هذا الأمر يسترجع كل الأعمدة من جدول "Movies" حيث تكون قيمة العمود "Rating" أقل من أو تساوي 8.0.

ملاحظات في دفترك

```sql
WHERE columnName ⩽ 'Value';
```

يستخدم لتحديد الصفوف في جدول قاعدة البيانات حيث تكون قيمة العمود المحدد (columnName) أقل من أو تساوي القيمة المحددة ('Value').

WHERE BETWEEN AND

الاستعلام عن الأفلام التي أصدرت بين عام 1990 وعام 2000

رابط الكود النهائي: الاستعلام عن الأفلام التي أصدرت بين عام 1990 وعام 2000

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
SELECT * FROM Movies WHERE ReleaseYear BETWEEN 1990 AND 2000;
```

يقوم بالتالي:

1. **SELECT * FROM Movies**: يعني استرجاع كافة الأعمدة (* تعني جميع الأعمدة) من جدول يسمى "Movies".
2. **WHERE ReleaseYear BETWEEN 1990 AND 2000**: يُضيف شرطاً للبحث، حيث يتم استرجاع الصفوف (السجلات) التي تحقق الشرط المحدد. في هذه الحالة، يتم استرجاع الأفلام التي تم إصدارها في الفترة بين عام 1990 وعام 2000، بما في ذلك السنتين الحدودية 1990 و 2000 أيضاً.

باختصار، هذا الأمر يساعد في استرجاع البيانات من جدول "Movies" حيث يكون عام الإصدار (ReleaseYear) في نطاق محدد معين، وفي هذه الحالة هو بين عامي 1990 و 2000.

ملاحظات في دفترك

```sql
WHERE columnName BETWEEN 'Value1' AND 'Value2';
```

يستخدم لتحديد الصفوف في جدول قاعدة البيانات حيث تكون قيمة العمود المحدد (columnName) ما بين القيمتين المحددتين (القيمة الأولى والثانية).

تدريب 304: ترتيب النتائج وتحديد عددها.

تفاصيل التمرين

1. يجب أن تقوم بالتدريب السابق 302 قبل الاستمرار بهذا التدريب.
2. سنقوم باستخدام جدول الأفلام للاستعلام عن سجلات معينة.
3. الاستعلام الأول: قم بالاستعلام عن كل الأفلام واعرضها مرتبة حسب تاريخ الإصدار من الأقدم للأحدث.
4. الاستعلام الثاني: قم بالاستعلام عن كل الأفلام واعرضها مرتبة حسب تاريخ الإصدار من الأحدث للأقدم.
5. الاستعلام الثالث: قم بالاستعلام عن كل الأفلام واعرضها مرتبة حسب تاريخ الإصدار من الأقدم للأحدث، ولكن اعرض فقط أول 3 نتائج من الاستعلام.
6. الاستعلام الرابع: قم بالاستعلام عن كل الأفلام واعرضها مرتبة حسب تاريخ الإصدار من الأحدث للأقدم، ولكن اعرض فقط أول 3 نتائج من الاستعلام.
7. رابط الأمر النهائي تمرين 304 - الكود النهائي

المعرّف	عنوان الفيلم	النوع	سنة الإصدار	التقييم
1	Inception	خيال علمي	2010	8.8
2	The Godfather	جريمة	1972	9.2
3	The Dark Knight	أكشن	2008	9.0
4	Pulp Fiction	جريمة	1994	8.9
5	The Shawshank Redemption	دراما	1994	9.3
6	The Matrix	خيال علمي	1999	8.7
7	Forrest Gump	دراما	1994	8.8
8	The Avengers	أكشن	2012	8.0
9	Gladiator	أكشن	2000	8.5
10	Titanic	رومانسي	1997	7.8

جدول الأفلام

شاهد تطبيق التمرين كفيديو تعليمي: لغة SQL الدليل الشامل الأول - التمرين 304

ORDER BY

الاستعلام عن كل الأفلام واعرضها مرتبة حسب تاريخ الإصدار من الأقدم للأحدث.

رابط الكود النهائي: الاستعلام عن الأفلام مرتبة حسب تاريخ الإصدار

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
SELECT *
FROM Movies
ORDER BY ReleaseYear;
```

هذا الأمر البسيط في SQL يعني "اختيار كل الأعمدة من جدول الأفلام وترتيب النتائج حسب سنة الإصدار بترتيب تصاعدي (من الأقدم إلى الأحدث)".

تفصيلاً:

- `SELECT *`: يختار جميع الأعمدة من الجدول المحدد، في هذه الحالة هو جدول `Movies`.
- `FROM Movies`: يحدد أن البيانات ستسترجع من جدول الأفلام.
- `ORDER BY ReleaseYear`: يعني أنه سيتم ترتيب الصفوف (السجلات) حسب قيمة العمود `ReleaseYear` بترتيب تصاعدي افتراضياً (من الأقدم إلى الأحدث).

بالتالي، هذا الاستعلام سيعيد كل البيانات من جدول الأفلام، مرتبة وفقاً لسنة الإصدار من الأقدم إلى الأحدث.

ملاحظات في دفترك

```sql
ORDER BY Column_name ASC
```

يتم ترتيب السجلات حسب قيمة عمود معين وبشكل افتراضي يتم الترتيب من الأقل للأكبر أو يعني من الأقدم للأحدث. يعني ترتيب تصاعدي.

ORDER BY ... DESC

الاستعلام عن الأفلام من الأحدث للأقدم

رابط الكود النهائي: الاستعلام عن الأفلام من الأحدث للأقدم

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
SELECT *
FROM Movies
ORDER BY ReleaseYear DESC;
```

هذا الأمر في SQL يعني "اختيار جميع الأعمدة من جدول الأفلام وترتيب النتائج حسب سنة الإصدار بترتيب تنازلي (من الأحدث إلى الأقدم)".

تفصيلاً:

- `SELECT *`: يختار جميع الأعمدة من الجدول المحدد، في هذه الحالة هو جدول `Movies`.
- `FROM Movies`: يحدد أن البيانات ستسترجع من جدول الأفلام.
- `ORDER BY ReleaseYear DESC`: يعني أنه سيتم ترتيب الصفوف (السجلات) حسب قيمة العمود `ReleaseYear` بترتيب تنازلي، أي من السنة الأحدث إلى السنة الأقدم.

بالتالي، هذا الاستعلام سيعيد كل البيانات من جدول الأفلام، مرتبة وفقاً لسنة الإصدار من الأحدث إلى الأقدم.

ملاحظات في دفترك

```sql
ORDER BY Column_name DESC
```

هذه العبارة في SQL تعني "ترتيب النتائج حسب قيمة العمود المحدد (`Column_name`) بترتيب تنازلي (من الأكبر إلى الأصغر)".

ORDER BY ... LIMIT

الاستعلام عن أقدم 3 أفلام.

رابط الكود النهائي: الاستعلام عن أقدم 3 أفلام

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
SELECT *
FROM Movies
ORDER BY ReleaseYear
LIMIT 3;
```

هذا الأمر في SQL يقوم بأداء الخطوات التالية:

1. `SELECT *`: يختار جميع الأعمدة من جدول الأفلام (`Movies`).
2. `FROM Movies`: يحدد جدول البيانات من الذي سيتم استرجاع البيانات.
3. `ORDER BY ReleaseYear`: يرتب الصفوف (السجلات) وفقاً لقيمة العمود `ReleaseYear` بترتيب تصاعدي (من الأقدم إلى الأحدث).
4. `LIMIT 3`: يحدد أنه سيتم استرجاع فقط أول 3 صفوف من النتائج المرتبة.

بالتالي، هذا الاستعلام سيقوم بإرجاع أول 3 أفلام من جدول الأفلام، حسب تاريخ الإصدار بترتيب من الأقدم إلى الأحدث.

ملاحظات في دفترك

```sql
ORDER BY Column_name LIMIT no_of_results
```

هذا الأمر في SQL يقوم بترتيب النتائج حسب قيمة العمود المحدد (`Column_name`) ومن ثم يحدد عدد النتائج التي سيتم استرجاعها (`no_of_results`).

ORDER BY ... DESC ... LIMIT

الاستعلام عن أحدث 3 أفلام.

رابط الكود النهائي: الاستعلام عن أحدث 3 أفلام

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
SELECT *
FROM Movies
ORDER BY ReleaseYear DESC
LIMIT 3;
```

هذا الأمر في SQL يعني:

1. `SELECT *`: يختار جميع الأعمدة من الجدول `Movies`.
2. `FROM Movies`: يحدد جدول البيانات الذي سيتم استرجاع البيانات منه، وفي هذه الحالة هو جدول `Movies`.
3. `ORDER BY ReleaseYear DESC`: يعني أنه سيتم ترتيب الصفوف (السجلات) حسب قيمة العمود `ReleaseYear` بترتيب تنازلي، أي من السنة الأحدث إلى السنة الأقدم.
4. `LIMIT 3`: يحدد أنه سيتم استرجاع فقط أول 3 صفوف من النتائج المرتبة بترتيب السنة الأحدث إلى السنة الأقدم.

بالتالي، هذا الاستعلام سيقوم بإرجاع أول 3 أفلام من جدول `Movies`، حسب تاريخ الإصدار بترتيب من الأحدث إلى الأقدم.

ملاحظات في دفترك

```sql
ORDER BY Column_name DESC LIMIT no_of_results
```

يُرتب الصفوف في نتيجة الاستعلام بناءً على قيمة العمود المحدد (`Column_name`) بترتيب تنازلي. يعني ذلك أن القيم الأعلى (أو الأكبر) في العمود ستكون أولاً. كما سيعيد عدد سجلات معين حسب قيمة `no_of_results`

تدريب 305: الدوال التجميعية

تفاصيل التمرين

1. قم بإنشاء جدول الطلبات أدناه.
2. قم بعد السجلات الموجودة في الجدول باستخدام COUNT
3. قم بحساب عدد كل المنتجات التي تم بيعها باستخدام SUM(Quantity)
4. قم بحساب متوسط سعر المبيعات باستخدام AVG(Price)
5. قم بحساب أصغر/أقل سعر للمنتجات باستخدام MIN(Price)
6. قم بحساب أعلى/أكبر سعر للمنتجات باستخدام MAX(Price)
7. رابط الأمر النهائي تمرين 305 - الكود النهائي

معرّف الطلب	معرف المنتج	الكمية	السعر	تاريخ الطلب
1	101	5	10.00	2024-07-01
2	102	3	20.00	2024-07-01
3	101	7	10.00	2024-07-02
4	103	2	30.00	2024-07-02
5	104	1	40.00	2024-07-03
6	102	4	20.00	2024-07-03
7	101	8	10.00	2024-07-04
8	103	6	30.00	2024-07-04

جدول الطلبات

شاهد تطبيق التمرين كفيديو تعليمي: لغة SQL الدليل الشامل الأول - التمرين 305

جدول الطلبات

قم بإنشاء جدول الطلبات

رابط الكود النهائي:إنشاء جدول الطلبات

نكتب الكود الآتي في موقع db-fiddle.com في قسم SCHEMA

```sql
-- Create the Sales table
CREATE TABLE Sales (
    SaleID INT PRIMARY KEY,
    ProductID INT,
    Quantity INT,
    Price DECIMAL(10, 2),
    SaleDate DATE
);

-- Insert some sample data into the Sales table
INSERT INTO Sales (SaleID, ProductID, Quantity, Price, SaleDate)
(1, 101, 5, 10.00, '2024-07-01'),
(2, 102, 3, 20.00, '2024-07-01'),
(3, 101, 7, 10.00, '2024-07-02'),
(4, 103, 2, 30.00, '2024-07-02'),
(5, 104, 1, 40.00, '2024-07-03'),
(6, 102, 4, 20.00, '2024-07-03'),
(7, 101, 8, 10.00, '2024-07-04'),
(8, 103, 6, 30.00, '2024-07-04');

```

نقوم بإنشاء جدول الطلبات في قاعدة البيانات

COUNT

قم بعد سجلات جدول الطلبات

رابط الكود النهائي: عد عدد السجلات الموجودة في الجدول

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
SELECT COUNT(*) AS TotalSales FROM Sales;
```

تفصيلاً:

SELECT: هذه الكلمة الرئيسية تُستخدم لتحديد البيانات التي تريد استخراجها من قاعدة البيانات.

COUNT(*): هذه الدالة تجمع عدد السجلات في الجدول المحدد. في هذه الحالة، تقوم بحساب عدد كل السجلات في جدول "Sales"

AS TotalSales.: هذه العبارة تُستخدم لإعطاء اسم مستعار للعمود الناتج. هنا، سيتم تسمية الناتج باسم "TotalSales"."

FROM Sales: هذه العبارة تحدد الجدول الذي سيتم استخراج البيانات منه، في هذه الحالة هو جدول "Sales"

هذه الجملة تقوم بحساب إجمالي عدد السجلات في جدول "المبيعات" (Sales) وتعطي الناتج اسم "إجمالي المبيعات" (TotalSales).

ملاحظات في دفترك

```sql
COUNT(*)
```

دالة COUNT(*) تحسب إجمالي عدد السجلات في الجدول المحدد

SUM

قم بعد سجلات جدول الطلبات

رابط الكود النهائي: حساب مجموع عدد المنتجات التي تم بيعها

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
SELECT SUM(Quantity) AS TotalQuantitySold FROM Sales
```

الأمر التالي هو استعلام بلغة SQL ويستخدم لجمع وإظهار مجموع الكميات المباعة من جدول المبيعات. دعنا نشرح كل جزء من الاستعلام:

1. **SELECT SUM(Quantity) AS TotalQuantitySold**:

- **SELECT**: يستخدم لتحديد الأعمدة التي نريد إظهارها في نتيجة الاستعلام.
- **SUM(Quantity)**: دالة تجمع (تجمع القيم) تستخدم لجمع كل القيم في عمود "Quantity".
- **AS TotalQuantitySold**: يحدد اسم مستعار (Alias) للعمود الناتج من عملية الجمع، في هذه الحالة الاسم سيكون "TotalQuantitySold".

1. **FROM Sales**:

- **FROM**: يستخدم لتحديد الجدول الذي نريد استخراج البيانات منه.
- **Sales**: اسم جدول المبيعات.

بالتالي، الاستعلام يقوم بجمع جميع القيم الموجودة في عمود "Quantity" في جدول "Sales" ويعرض النتيجة في عمود يسمى "TotalQuantitySold".

ملاحظات في دفترك

```sql
SUM(ColumnName)
```

دالة `SUM(ColumnName)` تُستخدم في SQL لحساب مجموع القيم الموجودة في عمود محدد.

AVG

حساب متوسط السعر في جدول المبيعات

رابط الكود النهائي: حساب متوسط سعر المنتجات

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
SELECT AVG(Price) AS AveragePrice FROM Sales
```

الأمر التالي هو استعلام بلغة SQL ويستخدم لحساب وإظهار متوسط الأسعار من جدول المبيعات. دعنا نشرح كل جزء من الاستعلام:

1. **SELECT AVG(Price) AS AveragePrice**:

- `SELECT`: يُستخدم لتحديد البيانات التي تريد استخراجها من قاعدة البيانات.
- `AVG(Price)`: هذه دالة `AVG` التي تقوم بحساب متوسط قيمة عمود معين، في هذه الحالة، العمود هو `Price` (السعر).
- `AS AveragePrice`: يستخدم لتسمية النتيجة المحسوبة بوسم معين، في هذا المثال، تم تسمية متوسط السعر بـ `AveragePrice`.

1. **FROM Sales**:

- `FROM`: يُستخدم لتحديد الجدول الذي سيتم استخراج البيانات منه.
- `Sales`: اسم الجدول الذي يحتوي على البيانات، في هذه الحالة، جدول `Sales` (المبيعات).

هذه هي الخطوات الأساسية لحساب متوسط السعر باستخدام دالة `AVG` في SQL.

ملاحظات في دفترك

```sql
AVG(ColumnName)
```

دالة **`AVG(ColumnName)`** تُستخدم في SQL لحساب متوسط القيم الموجودة في عمود محدد.

MIN

حساب أقل سعر في جدول المبيعات

رابط الكود النهائي: حساب أقل سعر المنتجات

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
SELECT MIN(Price) AS MinimumPrice FROM Sales
```

الأمر التالي هو استعلام بلغة SQL ويستخدم لحساب وإظهار أقل الأسعار من جدول المبيعات. دعنا نشرح كل جزء من الاستعلام:

1. **SELECT MIN(Price) AS MinimumPrice**: هذا الجزء يطلب من قاعدة البيانات أن تحدد أقل قيمة في عمود Price باستخدام الدالة MIN. يتم تسمية هذه القيمة باسم "MinimumPrice" في النتيجة.
2. **FROM Sales**: يحدد هذا الجزء من الاستعلام أن يتم جلب البيانات من جدول المبيعات.

ملاحظات في دفترك

```sql
MIN(ColumnName)
```

في SQL تُستخدم لاستخراج أقل قيمة في عمود معين. عندما تُستخدم هذه الدالة، تقوم بفحص جميع القيم في العمود المحدد وتعيد القيمة الصغرى. يُمكن استخدامها مع أنواع البيانات العددية والنصية.

MAX

حساب أعلى سعر في جدول المبيعات

رابط الكود النهائي: حساب أعلى سعر المنتجات

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
SELECT MAX(Price) AS MaximumPrice FROM Sales;
```

يقوم بتحديد أعلى قيمة للعمود Price في جدول Sales، ويعيد هذه القيمة كنتيجة واحدة مع تسمية النتيجة باسم MaximumPrice. في هذا السياق، MAX(Price) تستخدم لاسترجاع أكبر قيمة موجودة في عمود Price من بين جميع الصفوف في جدول Sales.

ملاحظات في دفترك

```sql
MAX(ColumnName)
```

يُستخدم لاسترجاع أكبر قيمة متواجدة في عمود محدد (ColumnName) من بين الصفوف في الجدول المحدد.

تدريب 306: GROUP BY

تخيل أنك تدير مكتبة ولديك قائمة بالكتب التي تم استعارتها. يمكنك تصنيف القائمة حسب نوع الكتاب، بحيث تكون كل الروايات معًا، وكل كتب العلم معًا. بعد ذلك، يمكنك حساب عدد الكتب المستعارة من كل نوع.

إذا كنت تدير مطعمًا وتحتفظ بقائمة طلبات الطعام، يمكنك تصنيف هذه الطلبات حسب نوع الطبق، مثل تصنيف جميع البيتزا معًا وجميع السلطات معًا. بعد ذلك، يمكنك معرفة عدد الطلبات لكل نوع من الأطباق.

إذا كنت تعمل في مدرسة ولديك قائمة بالدرجات التي حصل عليها الطلاب في مختلف المواد، يمكنك تصنيف هذه الدرجات حسب المادة، مثل وضع جميع درجات الرياضيات معًا وجميع درجات العلوم معًا. بعد ذلك، يمكنك حساب متوسط الدرجات لكل مادة

تفاصيل التمرين

1. هذا التمرين معتمد على جدول الطلبات في تمرين 305. حاول فهم تمرين 305 قبل الانتقال إلى هذا التمرين.
2. قم بحساب مجموع الكميات التي تم بيعها لكن لكل منتج على حدا.
3. قم بحساب متوسط الكميات التي تم بيعها حسب كل منتج على حدا.
4. قم بكتابة استعلام SQL يحسب إجمالي المبيعات لكل تاريخ (SaleDate) من جدول المبيعات (Sales). استخدم دالة SUM لحساب مجموع نتيجة ضرب الكمية (Quantity) في السعر (Price). يجب أن يتم تجميع النتائج حسب تاريخ البيع.
5. إذا كنت تريد معرفة أكبر كمية من المنتجات تم بيعها في يوم معين، يمكنك البحث عن أكبر عدد من الكميات في كل يوم لمعرفة اليوم الذي تم فيه بيع أكبر كمية.
6. إذا كنت تريد معرفة عدد المبيعات لكل منتج في المتجر، يمكنك عدّ عدد مرات بيع كل منتج لمعرفة عدد المبيعات لكل منتج.
7. رابط الأمر النهائي تمرين 306 - الكود النهائي

معرّف الطلب	معرف المنتج	الكمية	السعر	تاريخ الطلب
1	101	5	10.00	2024-07-01
2	102	3	20.00	2024-07-01
3	101	7	10.00	2024-07-02
4	103	2	30.00	2024-07-02
5	104	1	40.00	2024-07-03
6	102	4	20.00	2024-07-03
7	101	8	10.00	2024-07-04
8	103	6	30.00	2024-07-04

جدول الطلبات

شاهد تطبيق التمرين كفيديو تعليمي: لغة SQL الدليل الشامل الأول - التمرين 306

GROUP BY...SUM

مجموع الكميات حسب المنتج

رابط الكود النهائي: مجموع الكميات حسب المنتج

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
SELECT ProductID, SUM(Quantity) AS TotalQuantitySold
FROM Sales
GROUP BY ProductID;
```

تخيل أنك تدير متجرًا ولديك قائمة بجميع المنتجات التي تم بيعها وكمياتها. تريد معرفة إجمالي الكمية المباعة لكل منتج. لذا، تقوم بجمع الكميات المباعة لكل نوع من المنتجات لتعرف كم تم بيع كل منتج

بشكل إجمالي.

الكود يقوم بالآتي:

- **ProductID**: يعرض معرف كل منتج.
- **SUM(Quantity)**: يجمع الكميات المباعة لكل منتج.
- **FROM Sales**: يأخذ البيانات من جدول المبيعات.
- **GROUP BY ProductID**: يجمع البيانات حسب معرف المنتج.

الناتج سيكون جدولًا يعرض كل منتج والكمية الإجمالية التي تم بيعها منه

ملاحظات في دفترك

```sql
SUM(ColumnName)  GROUP BY AnotherColumnName
```

هذا الاستعلام يهدف إلى حساب مجموع قيمة عمود معين (ColumnName) بناءً على قيمة في عمود آخر (AnotherColumnName). يتم تجميع الصفوف التي تحتوي على نفس القيم في عمود AnotherColumnName معًا، ثم يتم حساب إجمالي قيمة العمود ColumnName لكل مجموعة.

AVG ... GROUP BY

متوسط الكمية حسب المنتج

رابط الكود النهائي: متوسط كمية الطلبات حسب المنتج

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
SELECT ProductID, AVG(QUANTITY) AS AverageQuantity
FROM Sales
GROUP BY ProductID;
```

هذا الاستعلام يقوم بتجميع بيانات المبيعات في جدول Sales بناءً على الحقل ProductID، ثم يحسب متوسط الكميات المباعة (Quantity) لكل منتج. الأمر يفعل ذلك باستخدام ال GROUP BY لتجميع

البيانات بحسب ProductID.

- `SELECT ProductID, AVG(Quantity) AS AverageQuantity`: يختار الاستعلام حقل ProductID ويحسب متوسط الكميات (Quantity) المباعة لكل منتج.

ملاحظات في دفترك

```sql
AVG(ColumnName)  GROUP BY AnotherColumnName
```

هذا الاستعلام يهدف إلى حساب متوسط قيمة عمود معين (ColumnName) بناءً على قيمة في عمود آخر (AnotherColumnName). يتم تجميع الصفوف التي تحتوي على نفس القيم في عمود AnotherColumnName معًا، ثم يتم حساب متوسط قيمة العمود ColumnName لكل مجموعة من الصفوف هذه.

SUM(Quantity * Price) GROUP BY

اجمالي مدفوعات الطلبات حسب اليوم

رابط الكود النهائي: اجمالي المبيعات حسب التاريخ

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
SELECT SaleDate, SUM(Quantity * Price)  TotalSales
FROM Sales
GROUP BY SaleDate;
```

هذا الأمر يهدف إلى حساب إجمالي المبيعات لكل تاريخ.

- `SELECT SaleDate, SUM(Quantity * Price) AS TotalSales`: يختار الأمر تاريخ البيع (SaleDate) ويحسب إجمالي المبيعات باستخدام الصيغة (Quantity * Price)، ثم يظهر النتيجة تحت اسم TotalSales.
- `FROM Sales`: يحدد الجدول الذي يأتي منه البيانات وهو جدول المبيعات.
- `GROUP BY SaleDate`: يجمع البيانات حسب تاريخ البيع (SaleDate)، مما يعني أنه سيحسب إجمالي المبيعات لكل تاريخ.

بالتالي، النتيجة ستكون قائمة تعرض تاريخ كل عملية بيع مع إجمالي المبيعات لذلك التاريخ.

MAX ... GROUP BY

أكبر كمية تم بيعها حسب اليوم

رابط الكود النهائي: العدد الأكبر لكمية المنتجات التي تم بيعها حسب اليوم

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql
SELECT SaleDate, MAX(Quantity) AS MaxQuantitySold
FROM Sales
GROUP BY SaleDate;
```

هذا الأمر يهدف إلى العثور على أقصى كمية تم بيعها في كل يوم.

- `SELECT SaleDate, MAX(Quantity) AS MaxQuantitySold`: يختار الأمر تاريخ البيع (SaleDate) ويحسب أقصى كمية تم بيعها باستخدام دالة MAX على الكمية (Quantity). النتيجة ستكون تحت اسم MaxQuantitySold.
- `FROM Sales`: يحدد الجدول الذي يأتي منه البيانات وهو جدول المبيعات.
- `GROUP BY SaleDate`: يجمع البيانات حسب تاريخ البيع (SaleDate)، مما يعني أنه سيتم حساب أقصى كمية مباعة لكل يوم مختلف.

بمعنى آخر، النتيجة ستكون قائمة تعرض تاريخ كل يوم مع أقصى كمية منتج تم بيعها في ذلك اليوم.

ملاحظات في دفترك

```sql
MAX(ColumnName)  GROUP BY AnotherColumnName
```

هذا الأمر في SQL يهدف إلى إيجاد أكبر قيمة في عمود معين (ColumnName) لكل مجموعة من الصفوف التي تتشابه في قيمة عمود آخر (AnotherColumnName).

COUNT(*) ... GROUP BY

عدد الطلبات لكل منتج

رابط الكود النهائي: عدد الطلبات لكل منتج

نكتب الكود الآتي في موقع db-fiddle.com في قسم Query

```sql

SELECT ProductID, COUNT(*) AS NumberOfSales
FROM Sales
GROUP BY ProductID;
```

هذا الأمر في SQL يهدف إلى إيجاد عدد المبيعات لكل منتج.

- `SELECT ProductID, COUNT(*) AS NumberOfSales`: يختار الأمر معرف المنتج (ProductID) ويحسب عدد المبيعات باستخدام دالة COUNT. يُعرض النتيجة تحت اسم NumberOfSales.
- `FROM Sales`: يُحدد الجدول الذي تأتي منه البيانات، وفي هذه الحالة هو جدول المبيعات (Sales).
- `GROUP BY ProductID`: يُجمع البيانات حسب معرف المنتج (ProductID)، مما يعني أنه سيتم حساب عدد المبيعات لكل منتج بناءً على معرفه.

بمعنى آخر، النتيجة ستكون قائمة تعرض معرف كل منتج مع عدد المبيعات التي تمت له.

ملاحظات في دفترك

```sql
COUNT(*) GROUP BY AnotherColumnName
```

هذا الأمر في SQL يستخدم لحساب عدد الصفوف (السجلات) في كل مجموعة من الصفوف التي تتشابه في قيمة عمود آخر (AnotherColumnName).

المزيد من أساسيات SQL

هنا سنتحدث عن المزيد من أساسيات SQL التي عليك معرفتها قبل الانتقال للمستوى التالي:

سيتم توضيح الأوامر الأساسية بالتدريبات العملية. نحن الآن في المستوى الابتدائي الثاني في SQL. سيبدأ ترقيم تمارين هذا القسم بـ 311

تدريب 311: Primary Key - المفتاح الرئيسي/المفتاح الأساسي

اذا أردت انشاء معرف لسجل لتمييزه فعليك تحديده أنه المعرف بكتابة بجانب اسم هذا العمود عند إنشاء الجدول PRIMARY KEY، وهذا يعني أن هذا هو المعرّف للسجل. عند تعيين ذلك، فأنت تقول لا يمكن لسجلين أن يكون لهما نفس المعرف. قيمة المعرّف ستكون فريدة ومميزة لكل قيد أو سجل.

```
StudentID PRIMARY KEY
```

من الشائع، أن يكون المعرّف عبارة عن رقم صحيح يعني INT

```
StudentID INT PRIMARY KEY
```

أيضاً لا يمكن للمعرّف أن يكون بدون قيمة يعني يجب أن يكون NOT NULL.

```
StudentID INT PRIMARY KEY NOT NULL
```

بدلاً من كتابة المعرّف مع كل إدخال، يمكن أن يتم اضافته بشكل أتوماتيكي بإضافة عبارة AUTO_INCREMENT والتي تعني الزيادة الأتوماتيكية لقيمة المعرّف. ستكون مع أول إدخال 1 ، والثاني 2 وهكذا.

```
StudentID INT PRIMARY KEY NOT NULL AUTO_INCREMENT
```

تفاصيل التمرين

1. قم بإنشاء جدول الطلاب
2. اجعل حقل معرّف الطالب هو المفتاح الأساسي.
3. اجعل قيمة المعرّف رقم صحيح.
4. لا يجب أن تكون قيمة المعرّف قيمة فارغة أو NULL.
5. لا داعي لإدخال المعرّف مع كل ادخال، اجعله يزيد بشكل أتوماتيكي بمقدار 1 مع كل ادخال.
6. رابط الأمر النهائي تمرين 311 - الكود النهائي

معرّف الطالب	الاسم الأول	الاسم الأخير	تاريخ الانضمام
1	علي	سليماني	2025-09-01
2	لانا	وسيم	2025-09-02
3	سامية	الخطاب	2025-09-03

جدول الطلاب

شاهد تطبيق التمرين كفيديو تعليمي: لغة SQL الدليل الشامل الأول - التمرين 311

جدول الطلاب

قم بإنشاء جدول الطلاب

رابط الكود النهائي:إنشاء جدول الطلاب

نكتب الكود الآتي في موقع db-fiddle.com في قسم SCHEMA

```sql
CREATE TABLE Students (
    StudentID  INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    EnrollmentDate DATE NOT NULL

);

INSERT INTO Students (FirstName, LastName, EnrollmentDate)
VALUES
('Ali', 'Suleiman', '2025-09-01'),
('Lana', 'Waseem', '2025-09-02'),
('Samia', 'Al-Khatib', '2025-09-03');

```

نقوم بإنشاء جدول الطلاب في قاعدة البيانات

ملاحظات في دفترك

```sql
ID  INT PRIMARY KEY NOT NULL AUTO_INCREMENT
```

- **INT**: هذا يحدد نوع البيانات للحقل، في هذه الحالة هو أرقام صحيحة (أعداد كاملة بدون أعشار).
- **PRIMARY KEY**: هذا يعني أن هذا الحقل هو المُعرف الرئيسي للسجل. يُستخدم لتمييز كل سجل بشكل فريد داخل الجدول.
- **NOT NULL**: يعني أن الحقل لا يمكن أن يكون فارغًا، أي لا يمكن أن يكون له قيمة مفقودة.
- **AUTO_INCREMENT**: يُستخدم لتعيين قيمة تتزايد تلقائيًا للحقل عند إضافة سجل جديد. هذا يسهل إنشاء مُعرفات فريدة لكل سجل بدون الحاجة لتحديدها يدويًا.

تدريب 312: Constrains | قيود SQL

قيود SQL هي قواعد تُفرض على البيانات في جداول قاعدة البيانات لضمان سلامة البيانات واتساقها. هذه القيود تساعد على منع الإدخالات غير الصالحة وضمان أن البيانات في قاعدة البيانات تفي بالشروط المحددة. في هذا التمرين، سنستعرض بعض القيود الشائعة في SQL وكيفية استخدامها عند إنشاء الجداول.

أمثلة على قيود SQL:

- **NOT NULL**: يضمن أن العمود لا يمكن أن يحتوي على قيمة فارغة.
- **NULL**: القيود `NULL` تسمح للأعمدة أن تحتوي على قيم فارغة. في الحقيقة، `NULL` هو الوضع الافتراضي لأي عمود إذا لم يتم تحديد أي قيد آخر مثل `NOT NULL`.
- **UNIQUE**: يضمن أن جميع القيم في العمود فريدة.
- **PRIMARY KEY**: مزيج من NOT NULL و UNIQUE. يحدد المفتاح الرئيسي لكل صف في الجدول.
- **FOREIGN KEY**: يربط عمودًا في جدول بعمود رئيسي في جدول آخر (*سنتعلم أكثر حول ذلك في المستوى القادم*).
- **CHECK**: يضمن أن القيم في العمود تفي بشرط معين.
- **DEFAULT**: يحدد قيمة افتراضية للعمود إذا لم يتم توفير قيمة.
- **AUTO_INCREMENT**: تُستخدم لجعل قيمة العمود تزيد تلقائيًا مع كل إدخال جديد. هذا القيد غالبًا ما يُستخدم مع الأعمدة التي تمثل المفاتيح الرئيسية (`PRIMARY KEY`).

يتم اضافة القيود أو constrains في نهاية تعريف العمود في أمر إنشاء الجدول.
مثال: `Username VARCHAR(50) NOT NULL UNIQUE`

تفاصيل التمرين

1. إنشاء جدول باسم `Countries` يحتوي على الأعمدة التالية:
   - **`CountryID`** من النوع **`INT`** ويكون *المفتاح الأساسي للجدول ويزداد تلقائيًا*.
   - **`CountryName`** من النوع **`VARCHAR`** بطول 50 حرفًا *ولا يقبل القيم الفارغة* ويكون *فريدًا*.

2. إنشاء جدول باسم `Missions` يحتوي على الأعمدة التالية:
   - **`MissionID`** من النوع **`INT`** ويكون *المفتاح الأساسي للجدول ويزداد تلقائيًا.*
   - **`MissionName`** من النوع **`VARCHAR`** بطول 100 حرف *ولا يقبل القيم الفارغة ويكون فريدًا.*
   - **`LaunchDate`** من النوع **`DATE`** ويكون له *القيمة الافتراضية `2023-01-01`.*
   - **`DurationDays`** من النوع **`INT`** *ويجب أن يكون أكبر من 0.*
3. إنشاء جدول باسم `Astronauts` يحتوي على الأعمدة التالية:
   - **`AstronautID`** من النوع **`INT`** ويكون *المفتاح الأساسي للجدول ويزداد تلقائيًا.*
   - **`FirstName`** من النوع **`VARCHAR`** بطول 50 حرفًا *ولا يقبل القيم الفارغة.*
   - **`LastName`** من النوع **`VARCHAR`** بطول 50 حرفًا *ولا يقبل القيم الفارغة.*
   - **`DateOfBirth`** من النوع **`DATE`** *ولا يقبل القيم الفارغة.*
   - **`Email`** من النوع **`VARCHAR`** بطول 100 حرف *ويكون فريدًا.*
   - **`CountryID`** من النوع **`INT`** ويمثل *مفتاحًا أجنبيًا يشير إلى `CountryID` في جدول `Countries`.*
   - **`MissionID`** من النوع **`INT`** *ويمكن أن يقبل القيم الفارغة ويمثل مفتاحًا أجنبيًا يشير إلى `MissionID` في جدول `Missions`*
4. رابط الأمر النهائي تمرين 312 - الكود النهائي

CountryName	CountryID
United States	1
Russia	2
China	3
Japan	4
France	5

جدول الدول أو البلدان

DurationDays	LaunchDate	MissionName	MissionID
8	1969-07-16	Apollo 11	1
172	2015-12-15	Soyuz TMA-19M	2
182	2021-10-15	Shenzhou 13	3
25	2023-01-01	Artemis I	4
42	2019-09-24	H-II Transfer Vehicle	5

جدول المهمات الفضائية

MissionID	CountryID	Email	DateOfBirth	LastName	FirstName	AstronautID
1	1	neil.armstrong@nasa.gov	1930-08-05	Armstrong	Neil	1
NULL	2	yuri.gagarin@roscosmos.ru	1934-03-09	Gagarin	Yuri	2
NULL	3	yang.liwei@cnsa.gov.cn	1965-06-21	Liwei	Yang	3
5	4	koichi.wakata@jaxa.jp	1963-08-01	Wakata	Koichi	4
2	5	thomas.pesquet@esa.int	1978-02-27	Pesquet	Thomas	5

جدول رواد الفضاء

شاهد تطبيق التمرين كفيديو تعليمي: لغة SQL الدليل الشامل الأول - التمرين 312

جدول البلدان

قم بإنشاء جدول الدول

رابط الكود النهائي:إنشاء جدول الدول

نكتب الكود الآتي في موقع db-fiddle.com في قسم SCHEMA

```
CREATE TABLE Countries (
    CountryID INT PRIMARY KEY AUTO_INCREMENT,
```

```sql
    CountryName VARCHAR(50) NOT NULL UNIQUE
);


INSERT INTO Countries (CountryID, CountryName) VALUES
(1, 'United States'),
(2, 'Russia'),
(3, 'China'),
(4, 'Japan'),
(5, 'France');
```

نقوم بإنشاء جدول الدول في قاعدة البيانات

ملاحظات في دفترك

```sql
CountryID  INT PRIMARY KEY AUTO_INCREMENT
```

- **INT**: هذا يحدد نوع البيانات للحقل، في هذه الحالة هو أرقام صحيحة (أعداد كاملة بدون أعشار).
- **PRIMARY KEY**: هذا يعني أن هذا الحقل هو المُعرف الرئيسي للسجل. يُستخدم لتمييز كل سجل بشكل فريد داخل الجدول.
- **AUTO_INCREMENT**: يُستخدم لتعيين قيمة تتزايد تلقائيًا للحقل عند إضافة سجل جديد. هذا يسهل إنشاء مُعرفات فريدة لكل سجل بدون الحاجة لتحديدها يدويًا.

```sql
CountryName VARCHAR(50) NOT NULL UNIQUE
```

- VARCHAR: **VARCHAR(50)** هو نوع بيانات يُستخدم لتخزين النصوص ذات الطول المتغير.
(50) يحدد الحد الأقصى لعدد الأحرف التي يمكن أن يحتويها العمود، وفي هذه الحالة، يمكن أن يحتوي العمود CountryName على ما يصل إلى 50 حرفًا.
- **NOT NULL**: يشير إلى أن العمود لا يمكن أن يحتوي على قيم فارغة (NULL). بمعنى آخر، يجب أن يكون هناك قيمة موجودة في هذا العمود لكل سجل في الجدول.
- **UNIQUE**: يضمن أن القيم في هذا العمود تكون فريدة عبر جميع السجلات في الجدول. لا يمكن أن يكون هناك تكرار للقيم في هذا العمود. إذا حاولت إدراج سجل جديد بنفس القيمة الموجودة في عمود CountryName لآخر سجل، ستظهر رسالة خطأ.

جدول المهمات

قم بإنشاء جدول المهمات

رابط الكود النهائي:إنشاء جدول المهمات

نكتب الكود الآتي في موقع db-fiddle.com في قسم SCHEMA

```sql
CREATE TABLE Missions (
    MissionID INT PRIMARY KEY AUTO_INCREMENT,
    MissionName VARCHAR(100) NOT NULL UNIQUE,
    LaunchDate DATE DEFAULT '2023-01-01',
    DurationDays INT CHECK (DurationDays > 0)
);

INSERT INTO Missions (MissionID, MissionName, LaunchDate, Durati
(1, 'Apollo 11', '1969-07-16', 8),
(2, 'Soyuz TMA-19M', '2015-12-15', 172),
(3, 'Shenzhou 13', '2021-10-15', 182),
(4, 'Artemis I', '2023-01-01', 25),
(5, 'H-II Transfer Vehicle', '2019-09-24', 42);

```

SQL

نقوم بإنشاء جدول المهمات في قاعدة البيانات

ملاحظات في دفترك

```sql
LaunchDate DATE DEFAULT '2023-01-01',
```

SQL

- **DATE**: هو نوع بيانات يُستخدم لتخزين التواريخ. يتضمن هذا النوع السنة والشهر واليوم. في بعض أنظمة إدارة قواعد البيانات، قد يكون هناك تنسيق محدد لتخزين التواريخ، ولكن بشكل عام، يُخزن التاريخ في تنسيق YYYY-MM-DD.
- **DEFAULT** يحدد القيمة الافتراضية التي تُستخدم إذا لم يتم تقديم قيمة للعمود عند إدراج سجل جديد.

- `'2023-01-01'` هي القيمة الافتراضية للتاريخ. إذا لم يتم تحديد قيمة لـ `LaunchDate` عند إدراج سجل جديد، فستُستخدم هذه القيمة الافتراضية.

```sql
DurationDays INT CHECK (DurationDays > 0)
```

- **INT**:هو نوع بيانات يُستخدم لتخزين القيم الرقمية الصحيحة. في هذه الحالة، يعني أن العمود سيحتوي على أعداد صحيحة.
- **CHECK (DurationDays > 0)**: هو قيد يُستخدم لتطبيق قواعد التحقق على القيم التي يمكن إدخالها في العمود.
- **(DurationDays > 0)** هو تعبير يحدد القاعدة التي يجب أن تحققها القيم المدخلة في هذا العمود. في هذه الحالة، القاعدة هي أن قيمة `DurationDays` يجب أن تكون أكبر من 0.

جدول رواد الفضاء

قم بإنشاء جدول رواد الفضاء

رابط الكود النهائي:إنشاء جدول رواد الفضاء

نكتب الكود الآتي في موقع db-fiddle.com في قسم SCHEMA

```sql
CREATE TABLE Countries (
    CountryID INT PRIMARY KEY AUTO_INCREMENT,
    CountryName VARCHAR(50) NOT NULL UNIQUE
);

INSERT INTO Countries (CountryID, CountryName) VALUES
(1, 'United States'),
(2, 'Russia'),
(3, 'China'),
(4, 'Japan'),
(5, 'France');

CREATE TABLE Missions (
    MissionID INT PRIMARY KEY AUTO_INCREMENT,
    MissionName VARCHAR(100) NOT NULL UNIQUE,
    LaunchDate DATE DEFAULT '2023-01-01',
    DurationDays INT CHECK (DurationDays > 0)
```

```sql
);

INSERT INTO Missions (MissionID, MissionName, LaunchDate, Durati
(1, 'Apollo 11', '1969-07-16', 8),
(2, 'Soyuz TMA-19M', '2015-12-15', 172),
(3, 'Shenzhou 13', '2021-10-15', 182),
(4, 'Artemis I', '2023-01-01', 25),
(5, 'H-II Transfer Vehicle', '2019-09-24', 42);


CREATE TABLE Astronauts (
    AstronautID INT PRIMARY KEY AUTO_INCREMENT,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    DateOfBirth DATE NOT NULL,
    Email VARCHAR(100) UNIQUE,
    CountryID INT,
    MissionID INT NULL,
    FOREIGN KEY (CountryID) REFERENCES Countries(CountryID),
    FOREIGN KEY (MissionID) REFERENCES Missions(MissionID)
);



INSERT INTO Astronauts (AstronautID, FirstName, LastName, DateOf
(1, 'Neil', 'Armstrong', '1930-08-05', 'neil.armstrong@nasa.gov'
(2, 'Yuri', 'Gagarin', '1934-03-09', 'yuri.gagarin@roscosmos.ru'
(3, 'Yang', 'Liwei', '1965-06-21', 'yang.liwei@cnsa.gov.cn', 3,
(4, 'Koichi', 'Wakata', '1963-08-01', 'koichi.wakata@jaxa.jp', 4
(5, 'Thomas', 'Pesquet', '1978-02-27', 'thomas.pesquet@esa.int',

```

SQL

نقوم بإنشاء جدول رواد الفضاء في قاعدة البيانات

ملاحظات في دفترك

```sql
FOREIGN KEY (CountryID) REFERENCES Countries(CountryID)
```

SQL

:FOREIGN KEY (CountryID)

- `CountryID` هو العمود في جدولك الحالي الذي يتم تحديده كمفتاح أجنبي (Foreign Key).
- المفتاح الأجنبي هو نوع من القيود يُستخدم لربط البيانات بين جداول مختلفة.

:REFERENCES Countries(CountryID)

- يعني أن العمود `CountryID` في جدولك يشير إلى العمود `CountryID` في جدول آخر يسمى `Countries`.
- هذا يعني أن القيم التي تُدخل في عمود `CountryID` في جدولك يجب أن تكون موجودة بالفعل في عمود `CountryID` في جدول `Countries`.

بالمختصر:

يتم استخدام المفتاح الأجنبي `CountryID` في جدولك لضمان أن أي قيمة يتم إدخالها فيه تتطابق مع قيمة موجودة في جدول `Countries`.

يضمن ذلك أن البيانات في العمود `CountryID` في جدولك تكون دائمًا متوافقة مع البيانات الموجودة في جدول `Countries`.

```sql
FOREIGN KEY (MissionID) REFERENCES Missions(MissionID)
```

- **`FOREIGN KEY (MissionID)`**:
- **`REFERENCES Missions(MissionID)`**:
  - يعني أن العمود `MissionID` في جدولك يشير إلى العمود `MissionID` في جدول آخر يسمى `Missions`.
  - هذا يعني أن القيم المدخلة في العمود `MissionID` في جدولك يجب أن تكون موجودة بالفعل في عمود `MissionID` في جدول `Missions`.

بالمختصر

المفتاح الأجنبي `MissionID` في جدولك يربط البيانات بمهمة معينة موجودة في جدول `Missions`.

يضمن هذا القيد أن كل قيمة في عمود `MissionID` في جدولك تتطابق مع قيمة موجودة بالفعل في عمود `MissionID` في جدول `Missions`

تدريب 313: أنواع البيانات | Data Types

في قواعد البيانات، تعتبر **أنواع البيانات** هي الأساس لفهم كيفية تخزين ومعالجة المعلومات. تُستخدم أنواع البيانات لتحديد نوع البيانات التي يمكن تخزينها في الأعمدة المختلفة داخل الجداول. كل نوع بيانات له خصائصه الفريدة، مثل الحجم والدقة، مما يحدد كيف يمكن التعامل مع البيانات المخزنة. من خلال استخدام الأنواع الصحيحة، يمكن تحسين أداء الاستعلامات وضمان دقة البيانات. في هذا التمرين، سنتعرف على الأنواع الأساسية للبيانات في SQL، مثل الأعداد الصحيحة، الأعداد العشرية، النصوص، والتواريخ، ونتعلم كيفية استخدامها بشكل فعال لإنشاء جداول وتخزين البيانات بطريقة منظمة.

أنواع البيانات في SQL:

سأقوم هنا بسرد أنواع البيانات في SQL، لا داعي أن تفهمها بشكل كامل. المهم أن تعرف أنها موجودة وتتقن الأنواع الموجودة في التمارين فقط.

1. **أنواع البيانات الرقمية (Numeric Data Types)**
   - **أعداد صحيحة (Integer Types)**
     - `INT`: للأعداد الصحيحة.
     - `TINYINT`: للأعداد الصغيرة جداً.
     - `SMALLINT`: للأعداد الصغيرة.
     - `MEDIUMINT`: للأعداد المتوسطة.
     - `BIGINT`: للأعداد الكبيرة جداً.
   - **أعداد عشرية (Decimal Types)**
     - `FLOAT`: للأعداد العشرية ذات الدقة العائمة.
     - `DOUBLE`: للأعداد العشرية ذات الدقة المزدوجة.
     - `DECIMAL`: للأعداد العشرية ذات الدقة الثابتة.
2. **أنواع البيانات النصية (Text Data Types)**
   - **نصوص ثابتة الطول (Fixed-Length Text)**
     - `CHAR(n)`: للنصوص ذات الطول الثابت، حيث يتم تحديد الطول عند إنشاء الجدول.
   - **نصوص متغيرة الطول (Variable-Length Text)**
     - `VARCHAR(n)`: للنصوص ذات الطول المتغير، حيث يمكنك تحديد الحد الأقصى لعدد الأحرف.
   - **نصوص كبيرة (Large Text)**
     - `TEXT`: للنصوص الطويلة، يمكن استخدامه لتخزين كميات كبيرة من النصوص.
     - `MEDIUMTEXT`: للنصوص الكبيرة جداً.
     - `LONGTEXT`: للنصوص الضخمة.
3. **أنواع البيانات الزمنية (Temporal Data Types)**
   - **تاريخ (Date Types)**
     - `DATE`: لتخزين التواريخ فقط (السنة، الشهر، اليوم).
     - `YEAR`: لتخزين السنة فقط.
   - **وقت (Time Types)**
     - `TIME`: لتخزين الوقت فقط (الساعات، الدقائق، الثواني).
   - **تاريخ ووقت (Date and Time Types)**
     - `DATETIME`: لتخزين التواريخ والأوقات معاً.
     - `TIMESTAMP`: لتخزين التاريخ والوقت مع توقيت UTC.
   - **مدة (Duration Types)**
     - `INTERVAL`: لتخزين فترات زمنية (مستخدم في بعض قواعد البيانات مثل PostgreSQL).
4. **أنواع البيانات الثنائية (Binary Data Types)**
   - **بيانات ثنائية كبيرة (Large Binary Data)**
     - `BLOB`: لتخزين البيانات الثنائية الكبيرة مثل الصور أو ملفات الفيديو.
   - **بيانات ثنائية متغيرة الطول (Variable-Length Binary Data)**
     - `VARBINARY(n)`: لتخزين البيانات الثنائية بحد أقصى محدد.
5. **أنواع البيانات المنطقية (Boolean Data Types)**
   - `BOOLEAN` أو `BOOL`: لتخزين القيم المنطقية True/False.
6. **أنواع البيانات الأخرى (Other Data Types)**
   - **القيم المحددة (Enumerated Types)**: لتخزين مجموعة من القيم المحددة مسبقاً، ويكون كل قيمة مميزة.
   - **المجموعات (Set Types)**: لتخزين مجموعة من القيم التي يمكن أن تحتوي على عدة اختيارات من مجموعة محددة.

- **JSON**: لتخزين البيانات في شكل JSON (مستخدم في قواعد بيانات مثل MySQL).
- **UUID**: لتخزين قيم معرف فريدة عالمياً.

في التمرين التالي سنقوم بالتركيز على أنواع البيانات الأكثر استخداماً والأكثر شيوعاً. الرجاء التركيز عليها واتقانها.

تفاصيل التمرين

1. قم بإنشاء جدول في قاعدة بيانات SQL يتضمن المعلومات التالية حول حركة الكواكب في النظام الشمسي:
   - **العمود الأول**: `planet_id` - مفتاح أساسي (Primary Key) من نوع `INT` يتم زيادته تلقائيًا مع كل إدراج جديد.
   - **العمود الثاني**: `planet_name` - اسم الكوكب من نوع `VARCHAR(50)`.
   - **العمود الثالث**: `orbital_period` - فترة دوران الكوكب حول الشمس بالأيام من نوع `DECIMAL(10, 2)`.
   - **العمود الرابع**: `average_distance` - متوسط المسافة بين الكوكب والشمس بملايين الكيلومترات من نوع `FLOAT`
2. قم بإنشاء جدول في قاعدة بيانات SQL لتخزين معلومات حول الكتب. يجب أن يتضمن الجدول الأعمدة التالية:
   - **العمود الأول**: `book_id` - مفتاح أساسي (Primary Key) من نوع `INT` يتم زيادته تلقائيًا مع كل إدراج جديد.
   - **العمود الثاني**: `title` - عنوان الكتاب من نوع `VARCHAR(100)`، حيث يمكن أن يتغير طوله ولكنه لا يتجاوز 100 حرف.
   - **العمود الثالث**: `author_name` - اسم المؤلف من نوع `CHAR(50)`، حيث يكون طول النص ثابتًا يصل إلى 50 حرف.
   - **العمود الرابع**: `summary` - ملخص الكتاب من نوع `TEXT`، يمكن أن يكون طويلاً.
3. قم بإنشاء جدول في قاعدة بيانات SQL لتخزين معلومات حول الأحداث والمناسبات. يجب أن يتضمن الجدول الأعمدة التالية:
   - **EventID** - معرف الحدث، يكون من نوع INT، ويعمل كمفتاح أساسي (Primary Key) ويتم زيادته تلقائيًا مع كل إدراج جديد.
   - **EventName** - اسم الحدث من نوع VARCHAR(100)، حيث يمكن أن يتغير طوله ولكنه لا يتجاوز 100 حرف.
   - **EventDate** - تاريخ الحدث من نوع DATE، يتم تخزينه بصيغة السنة والشهر واليوم.
   - **StartTime** - وقت بداية الحدث من نوع TIME، يتم تخزينه بصيغة الساعة والدقيقة والثانية.
   - **EndTime** - وقت نهاية الحدث من نوع TIME، يتم تخزينه بصيغة الساعة والدقيقة والثانية.
   - **EventTimestamp** - طابع زمني لتسجيل الوقت الفعلي لإضافة السجل، من نوع TIMESTAMP، وتكون قيمته الافتراضية هي الوقت الحالي عند الإدراج (DEFAULT CURRENT_TIMESTAMP).
   - **EventDateTime** - تاريخ ووقت الحدث معًا من نوع DATETIME، حيث يتم تخزين كل من التاريخ والوقت.
   - **EventYear** - السنة التي يحدث فيها الحدث من نوع YEAR، يتم تخزين السنة فقط.
   - **Location** - موقع الحدث من نوع VARCHAR(100)، حيث يمكن أن يتغير طوله ولكنه لا يتجاوز 100 حرف.

4. قم بإنشاء جدول في قاعدة بيانات SQL لتخزين معلومات حول البوكيمونات. يجب أن يتضمن الجدول الأعمدة التالية:
   - **PokemonID** – معرف البوكيمون، يكون من نوع INT، ويعمل كمفتاح أساسي (Primary Key) ويتم زيادته تلقائيًا مع كل إدراج جديد.
   - **PokemonName** – اسم البوكيمون من نوع VARCHAR(50)، حيث يمكن أن يتغير طوله ولكنه لا يتجاوز 50 حرف.
   - **PokemonType** – نوع البوكيمون من نوع ENUM، حيث يمكن أن يكون أحد القيم المحددة فقط: 'Fire', 'Water', 'Grass', 'Electric', 'Rock', 'Psychic', 'Ice', 'Dragon'.
   - **IsStrong** – قيمة من نوع BOOLEAN، تشير إلى ما إذا كان البوكيمون يعتبر قويًا، والقيمة الافتراضية هي FALSE.
   - **IsPowerful** – قيمة من نوع BOOLEAN، تشير إلى ما إذا كان البوكيمون يعتبر قويًا من حيث القوة، والقيمة الافتراضية هي FALSE.
   - **IsWaterType** – قيمة من نوع BOOLEAN، تشير إلى ما إذا كان البوكيمون من نوع الماء، والقيمة الافتراضية هي FALSE.
   - **BaseAttack** – قيمة من نوع INT، تمثل القيمة الأساسية لهجوم البوكيمون.
   - **BaseDefense** – قيمة من نوع INT، تمثل القيمة الأساسية لدفاع البوكيمون.
   - **BaseSpeed** – قيمة من نوع INT، تمثل القيمة الأساسية لسرعة البوكيمون.
5. رابط الأمر النهائي تمرين 313 – الكود النهائي

average_distance (millions km)	orbital_period (days)	planet_name	planet_id
57.91	88.00	Mercury	1
108.21	224.70	Venus	2
149.60	365.25	Earth	3
227.92	687.00	Mars	4
778.57	4332.82	Jupiter	5
1433.53	10759.22	Saturn	6
2872.46	30688.50	Uranus	7
4495.06	60182.00	Neptune	8

جدول حركة الكواكب

summary	author_name	title	book_id
A political treatise on how to acquire and maintain power, based on historical experiences and events.	Niccolò Machiavelli	Al-Amir	1

summary	author_name	title	book_id
A dystopian novel exploring social surveillance and totalitarian rule in a fictional world under constant monitoring by “Big Brother”.	George Orwell	1984	2
A social novel depicting the lives of various characters in an Egyptian alley, addressing social and cultural issues through symbolism.	Naguib Mahfouz	Awlad Haratina	3

جدول الكتب والمؤلفين

Location	EventYear	EventDateTime	EventTimestamp	EndTime	StartTime	EventDate	EventName	EventID
City Art Gallery	2024	2024-08-10 18:00:00	2024-08-06 14:32:10	22:00:00	18:00:00	2024-08-10	Art Exhibition	1
Downtown Music Hall	2024	2024-09-05 20:00:00	2024-08-06 14:32:10	23:30:00	20:00:00	2024-09-05	Jazz Concert	2
Grand Theater	2024	2024-10-12 19:30:00	2024-08-06 14:32:10	21:30:00	19:30:00	2024-10-12	Theater Play	3
Convention Center	2025	2025-01-15 09:00:00	2024-08-06 14:32:10	17:00:00	09:00:00	2025-01-15	Tech Conference	4

جدول المناسبات والأحداث

BaseSpeed	BaseDefense	BaseAttack	IsWaterType	IsPowerful	IsStrong	PokemonType	PokemonName	PokemonID
100	78	84	FALSE	TRUE	TRUE	Fire	Charizard	1

BaseSpeed	BaseDefense	BaseAttack	IsWaterType	IsPowerful	IsStrong	PokemonType	PokemonName	PokemonID
78	100	83	TRUE	TRUE	TRUE	Water	Blastoise	2
45	49	49	FALSE	FALSE	FALSE	Grass	Bulbasaur	3
90	40	55	FALSE	FALSE	TRUE	Electric	Pikachu	4
81	79	125	TRUE	TRUE	TRUE	Water	Gyarados	5

جدول البوكيمون

شاهد تطبيق التمرين كفيديو تعليمي: لغة SQL الدليل الشامل الأول - التمرين 313 الجزء الأول
لغة SQL الدليل الشامل الأول - التمرين 313 الجزء الثاني
لغة SQL الدليل الشامل الأول - التمرين 313 الجزء الثالث

جدول حركة الكواكب

قم بإنشاء جدول حركة الكواكب

رابط الكود النهائي:إنشاء جدول الكواكب

نكتب الكود الآتي في موقع db-fiddle.com في قسم SCHEMA

```sql
CREATE TABLE planetary_data (
    planet_id INT AUTO_INCREMENT PRIMARY KEY,      -- للسلي للكوكب
    planet_name VARCHAR(50),                       -- اسم الكوكب
    orbital_period DECIMAL(10, 2),                 -- شمس (بالأيام)
    average_distance FLOAT                         -- الكيلومترات)
);

```

```sql
-- إدراج بيانات
INSERT INTO planetary_data (planet_name, orbital_period, average
VALUES
('Mercury', 88.00, 57.91),
('Venus', 224.70, 108.21),
('Earth', 365.25, 149.60),
('Mars', 687.00, 227.92),
('Jupiter', 4332.82, 778.57),
('Saturn', 10759.22, 1433.53),
('Uranus', 30688.50, 2872.46),
('Neptune', 60182.00, 4495.06);
```

SQL

نقوم بإنشاء جدول حركة الكواكب في قاعدة البيانات:

- **`planet_id INT AUTO_INCREMENT PRIMARY KEY`**
  - عمود لتخزين اسم الكوكب. يمكن تخزين أسماء مثل "Venus"، "Mars"، "Earth".
- **`planet_name VARCHAR(50)`**
  - عمود لتخزين الرقم التسلسلي الفريد لكل كوكب. يتم زيادته تلقائيًا مع كل إدراج جديد
- **`orbital_period DECIMAL(10, 2)`**
  - عمود لتخزين فترة دوران الكوكب حول الشمس. تُستخدم دقة ثابتة لتخزين قيم مثل 365.25 للأرض، مع تحديد 2 منازل عشرية.
- **`average_distance FLOAT`**
  - عمود لتخزين متوسط المسافة بين الكوكب والشمس. تُستخدم دقة عائمة لتخزين قيم مثل 149.6 لملايين الكيلومترات للأرض.

ملاحظات في دفترك

```sql
planet_id INT AUTO_INCREMENT PRIMARY KEY
```

SQL

INT: لتخزين الأعداد الصحيحة مثل 1 أو 100 بدون أي قيمة عشرية.

```sql
orbital_period DECIMAL(10, 2)
```

SQL

DECIMAL: لتخزين الأعداد العشرية بدقة ثابتة، مما يسمح بتخزين قيم دقيقة مثل 12345.67.

```sql
average_distance FLOAT
```

FLOAT: لتخزين الأعداد العشرية بدقة عائمة، مما يوفر دقة أقل من DECIMAL ويستخدم في القيم التي لا تحتاج لدقة عالية جداً مثل 3.4465481

جدول الكتب

قم بإنشاء جدول الكتب والمؤلفين

رابط الكود النهائي:إنشاء جدول الكتب والمؤلفين

نكتب الكود الآتي في موقع db-fiddle.com في قسم SCHEMA

```sql
CREATE TABLE books (
    book_id INT AUTO_INCREMENT PRIMARY KEY,        -- للسلي للكتاب
    title VARCHAR(100),                            -- جاوز 100 حرف
    author_name CHAR(50),                          -- ل إلى 50 حرف
    summary TEXT                                   -- أن يكون طويلاً
);

-- إدراج بيانات
INSERT INTO books (title, author_name, summary)
VALUES
('Al-Amir', 'Niccolò Machiavelli', 'A political treatise on how
('1984', 'George Orwell', 'A dystopian novel exploring social su
('Awlad Haratina', 'Naguib Mahfouz', 'A social novel depicting t
```

نقوم بإنشاء جدول حركة الكتب والمؤلفين في قاعدة البيانات:

- `title VARCHAR(100)`

- **varchar** نوع بيانات يُستخدم لتخزين النصوص ذات الطول المتغير. يمكن أن يتغير طول النص حسب الحاجة
- **(100)**: يُحدد الحد الأقصى لطول النص المسموح به، والذي هو 100 حرف في هذه الحالة. يعني أن عنوان الكتاب يمكن أن يصل إلى 100 حرف كحد أقصى.

- **author_name CHAR(50)**
  - **CHAR**: نوع بيانات يُستخدم لتخزين النصوص ذات الطول الثابت. يتم تخصيص طول ثابت لكل قيمة.
- **summary TEXT**
  - **Text**: نوع بيانات يُستخدم لتخزين كميات كبيرة من النصوص. يتيح تخزين نصوص طويلة تتجاوز الطول الذي يمكن تخزينه في أنواع بيانات أخرى مثل VARCHAR.

ملاحظات في دفترك

```sql
title VARCHAR(100)
```

VARCHAR(n): لتخزين نصوص بطول متغير يصل إلى n حرف، مما يوفر مرونة في تخزين النصوص ذات الأطوال المختلفة.

```sql
 author_name CHAR(50)
```

CHAR(n): لتخزين نصوص بطول ثابت قدره n حرف، مما يضمن أن جميع القيم في هذا العمود تكون بنفس الطول.

```sql
summary TEXT
```

TEXT: لتخزين نصوص طويلة جدًا تتجاوز الحدود المقررة لأنواع البيانات الأخرى، مما يتيح تخزين محتوى كبير.

جدول المناسبات

قم بإنشاء جدول المناسبات

رابط الكود النهائي:إنشاء جدول الأحداث والمناسبات

نكتب الكود الآتي في موقع db-fiddle.com في قسم SCHEMA

```sql
CREATE TABLE Events (
    EventID INT PRIMARY KEY AUTO_INCREMENT,
    EventName VARCHAR(100),
    EventDate DATE,                    -- تاريخ الحدث
    StartTime TIME,                    -- وقت بداية الحدث
    EndTime TIME,                      -- وقت نهاية الحدث
    EventTimestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP, -- السجل
    EventDateTime DATETIME,            -- تاريخ ووقت الحدث معًا
    EventYear YEAR,                    -- التي يحدث فيها الحدث
    Location VARCHAR(100)              -- موقع الحدث
);


INSERT INTO Events (EventName, EventDate, StartTime, EndTime, Ev
VALUES
('Art Exhibition', '2024-08-10', '18:00:00', '22:00:00', '2024-0
('Jazz Concert', '2024-09-05', '20:00:00', '23:30:00', '2024-09-
('Theater Play', '2024-10-12', '19:30:00', '21:30:00', '2024-10-
('Tech Conference', '2025-01-15', '09:00:00', '17:00:00', '2025-

```

نقوم بإنشاء جدول المناسبات في قاعدة البيانات:

هنا توضيح لأنواع البيانات المستخدمة في الجدول Events:

- **INT**
  نوع بيانات رقمي يستخدم لتخزين الأعداد الصحيحة. في هذه الحالة، يستخدم لتخزين EventID، وهو معرف فريد لكل حدث. تم تعيينه كـ PRIMARY KEY و AUTO_INCREMENT لضمان أن كل حدث يحصل على معرف فريد يتزايد تلقائيًا.

- **VARCHAR(100)**
نوع بيانات يستخدم لتخزين النصوص. يمكن أن يحتوي على سلسلة من الأحرف بطول يصل إلى 100 حرف. يستخدم لتخزين أسماء الأحداث (EventName) والمواقع (Location).
- **DATE**
نوع بيانات يستخدم لتخزين التاريخ. يتضمن فقط السنة والشهر واليوم. يستخدم لتخزين تاريخ الحدث (EventDate).
- **TIME**
نوع بيانات يستخدم لتخزين الوقت. يتضمن الساعة والدقيقة والثانية. يستخدم لتخزين وقت بداية الحدث (StartTime) ووقت نهاية الحدث (EndTime).
- **TIMESTAMP**
نوع بيانات يستخدم لتخزين الطابع الزمني. يتضمن التاريخ والوقت. عادةً ما يتم استخدامه لتسجيل وقت إنشاء أو تعديل السجل. هنا يستخدم لتخزين وقت إضافة السجل الفعلي (EventTimestamp) ويحدد قيمته الافتراضية بـ CURRENT_TIMESTAMP.
- **DATETIME**
نوع بيانات يجمع بين التاريخ والوقت. يستخدم لتخزين تاريخ ووقت حدث معين معًا (EventDateTime).
- **YEAR**
نوع بيانات يستخدم لتخزين السنة فقط. يستخدم لتحديد السنة التي يحدث فيها الحدث (EventYear).

ملاحظات في دفترك

```sql
EventDate DATE
```

Date يُستخدم لتخزين تاريخ معين في قاعدة البيانات. يتم تمثيله بثلاثة أجزاء: السنة (YYYY)، والشهر (MM)، واليوم (DD). مثال: 2024-08-10

```sql
StartTime TIME,
EndTime TIME
```

TIME هو نوع بيانات يُستخدم لتخزين الوقت فقط دون تاريخ. يُمثل الوقت على شكل ساعات ودقائق وثوانٍ. مثال: 18:00:00

```sql
EventTimestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
```

TIMESTAMP هو نوع بيانات يُستخدم لتخزين الطابع الزمني، ويشمل كل من التاريخ والوقت. يتم تعيين القيمة الافتراضية إلى CURRENT_TIMESTAMP، مما يعني أنه سيتم تلقائيًا تسجيل الوقت الحالي عند إدخال سجل جديد. مثال:2024-08-06 11:51:20

```sql
EventDateTime DATETIME
```

DATETIME هو نوع بيانات يُستخدم لتخزين كل من التاريخ والوقت معًا. يُمثل التاريخ والوقت على شكل سنة وشهر ويوم وساعة ودقيقة وثانية. مثال: 2024-08-10 18:00:00

```sql
EventYear YEAR
```

YEAR هو نوع بيانات يُستخدم لتخزين السنة فقط، بدون تاريخ أو وقت محدد. يُمثل السنة على شكل رقم مكون من أربعة أرقام، مثال: 2024.

جدول البوكيمون

قم بإنشاء جدول البوكيمونات

رابط الكود النهائي: إنشاء جدول البوكيمون

نكتب الكود الآتي في موقع db-fiddle.com في قسم SCHEMA

```sql
CREATE TABLE Pokemon (
    PokemonID INT PRIMARY KEY AUTO_INCREMENT,
    PokemonName VARCHAR(50) NOT NULL,
    PokemonType ENUM('Fire', 'Water', 'Grass', 'Electric', 'Rock
    IsStrong BOOLEAN DEFAULT FALSE,
    IsPowerful BOOLEAN DEFAULT FALSE,
```

```sql
    IsWaterType BOOLEAN DEFAULT FALSE,
    BaseAttack INT,
    BaseDefense INT,
    BaseSpeed INT
);


INSERT INTO Pokemon (PokemonName, PokemonType, IsStrong, IsPower
VALUES
('Charizard', 'Fire', TRUE, TRUE, FALSE, 84, 78, 100),
('Blastoise', 'Water', TRUE, TRUE, TRUE, 83, 100, 78),
('Bulbasaur', 'Grass', FALSE, FALSE, FALSE, 49, 49, 45),
('Pikachu', 'Electric', TRUE, FALSE, FALSE, 55, 40, 90),
('Gyarados', 'Water', TRUE, TRUE, TRUE, 125, 79, 81);


```

SQL

نقوم بإنشاء جدول البوكيمون في قاعدة البيانات:

هنا توضيح لأنواع البيانات المستخدمة في الجدول `Pokemon`:

- **PokemonID** – `INT` – معرف فريد لكل بوكيمون، يتم زيادته تلقائيًا مع كل إدخال جديد. يستخدم كمفتاح أساسي (`PRIMARY KEY`).
- **PokemonName** – `VARCHAR(50)` – اسم البوكيمون، نص بطول يصل إلى 50 حرفًا. لا يمكن أن يكون فارغًا (`NOT NULL`).
- **PokemonType** – `ENUM('Fire', 'Water', 'Grass', 'Electric', 'Rock', 'Psychic', 'Ice', 'Dragon')` – نوع البوكيمون، يمكن أن يكون أحد الأنواع المحددة فقط: `Fire`, `Water`, `Grass`, `Electric`, `Rock`, `Psychic`, `Ice`, `Dragon`. لا يمكن أن يكون فارغًا (`NOT NULL`).
- **IsStrong** – `BOOLEAN` – يشير إلى ما إذا كان البوكيمون يعتبر قويًا. القيمة الافتراضية هي `FALSE`.
- **IsPowerful** – `BOOLEAN` – يشير إلى ما إذا كان البوكيمون يعتبر قويًا جداً. القيمة الافتراضية هي `FALSE`.
- **IsWaterType** – `BOOLEAN` – يشير إلى ما إذا كان البوكيمون من نوع الماء. القيمة الافتراضية هي `FALSE`.
- **BaseAttack** – `INT` – القيمة الأساسية لهجوم البوكيمون. عدد صحيح يمثل مدى قوة هجوم البوكيمون.
- **BaseDefense** – `INT` – القيمة الأساسية لدفاع البوكيمون. عدد صحيح يمثل مدى قوة دفاع البوكيمون.
- **BaseSpeed** – `INT` – القيمة الأساسية لسرعة البوكيمون. عدد صحيح يمثل مدى سرعة البوكيمون.

ملاحظات في دفترك

```sql

PokemonType ENUM('Fire', 'Water', 'Grass', 'Electric', 'Rock', 'P

```

ENUM هو نوع بيانات في SQL يحدد مجموعة من القيم الممكنة فقط. في حقل ENUM، يمكنك تحديد قائمة بالقيم الثابتة المسموح بها، ويمكن للعمود أن يحتوي على واحدة فقط من هذه القيم.

```sql
IsStrong BOOLEAN DEFAULT FALSE,
IsPowerful BOOLEAN DEFAULT FALSE,
IsWaterType BOOLEAN DEFAULT FALSE,
```

BOOLEAN هو نوع بيانات في SQL يُستخدم لتخزين القيمتين TRUE أو FALSE.

هذا الدليل قيد التحديث المستمر وسيتم إضافة تمارين أسبوعياً.

تواصل معي

يسرني التواصل معكم عبر البريد الإلكتروني

mail at LearnWithNaw dot net

كيف أستطيع تسهيل أعمالك؟