الفصل الحادي عشر
قواعد البيانات والـ Mysql

في البداية سنتعرف على مصطلح الـ RDBMS، ونعني بذلك قواعد البيانات العلائقية، والتي من خصائصها سهولة الوصول إلى البيانات المخزنة فيها، وسرعة إتمام عمليات الاستعلام المختلفة، وبالاضافة إلى المميزات الأخرى فإن هذا النوع يعتبر الأكثر استخداماً في جميع التطبيقات سواء المستخدمة في الإنترنت أو ذات الطابع البرمجي الخاص، وبطبيعة الحال فإن الـ Mysql من هذا النوع.

ومن المهم معرفة بعض الأساسيات في الـ RDBMS، والتي من شأنها تسهيل عملية فهمك التام لطريقة عملها والتعامل معها..

١- الجداول Tables:

تعتبر أكبر جزء في قاعد البيانات، وهي عبارة عن أعمدة وصفوف تحتوي على قيم معينة.

٢- الأعمدة Columns:

لكل عمود في الجدول اسم خاص يختلف عن أسماء الأعمدة الأخرى في نفس الجدول، ويجب أن يكون لكل عمود نوع خاص به يصف نوع البيانات التي ستخزن فيه، وكم يظهر في الصورة، فإن عمود الرقم من النوع الرقمي Integer، أما الحقلين الآخرين فهي نصوص Text.

٣- الصفوف Rows:

كل صف من صفوف الجدول يحتوي على قيم مختلفة ويمثل معلومات متكاملة عن قطاع معين، وفي مثالنا يمثل معلومات متكاملة عن شخص معين.

٤- القيم Values:

وهي ما تحتوي عليه تقاطعات الصفوف بالأعمدة.

٥- المفاتيح Keys:

وتعتبر من أساليب تسهيل الوصول إلى المعلومات في قواعد البيانات، وفي مثالنا السابق نرى أن العمود Id يحتوي على أرقام متسلسلة لا تتكرر نهائياً بل إنها تتكون بشكل تلقائي عند إدراج أي صف جديد للجدول، وبالتالي فإنها تعتبر المفتاح المناسب لكل صف من صفوف الجدول لضمان عدم الالتباس في اختيار الصفوف.

فلو افترضنا أن لدينا جدولين في قاعدة بيانات، يحتوي الجدول الأول على معلومات عن الدروس مفصلة على عدة حقول لتلك الدروس، على سبيل المثال:
الرقم (id)، الدرس (lesson)، رقم الكاتب (Key_author)..
ويحتوي الجدول الثاني على بيانات الأعضاء كما يلي:
الرقم (Key_author)، الاسم (name)..

والمطلوب هو طريقة لربط الجدولين، بحيث أن رقم الكاتب في جدول الدروس (Key_author) يدل على اسم الكاتب في جدول الأعضاء (name).

بالتدقيق في المثال يتضح أن الحقلين (أو العمودين) Key_author في كلا الجدولين هو مفتاح الربط بينهما، ولذلك يمكن الوصول إلى اسم الكاتب اعتماداً على رقمه من جدول الدروس، وبالتالي الربط بين الجدولين.

لن أتحدث طويلاً عن مقدمات قواعد البيانات Mysql، ولكن بهذه المقدمة البسيطة يمكن على الأقل تصور بعض الأساسيات حول قواعد البيانات عموماً والـ Mysql خصوصاً، ومن وجهة نظري فالاهم هو كيفية التعامل مع قواعد البيانات

بما يخدم احتياجاتنا مع الـ PHP، ولذلك سأتطرق في هذا الفصل إلى نقطة هامة جداً وهي إدارة قواعد البيانات، وأعني بذلك عملية إنشاء قواعد البيانات والجداول والتحكم في الحقول والبيانات وغيرها، لتكون الأساس للتعامل مع قواعد البيانات لاحقاً عن طريق الـ PHP، ولعمل ذلك يوجد عدة طرق من أهمها الطريقة التقليدية المباشرة بالاعتماد على نظام الدوس في ذلك وبدون استخدام أي برامج أخرى للإدارة.

الاتصال بالـ Mysql، والتعامل معها:

كما قلنا أن الطريقة التقليدية هي الاتصال بقواعد البيانات عن طريق سيرفر الـ Mysql وبدون استخدام أي مكونات أخرى، ولعمل ذلك نحتاج أن نعرف مسار سيرفر الـ Mysql على الجهاز المستخدم بعد عملية التثبيت، كما قمنا بذلك في فصل المقدمة، وعادة يكون المسار كالتالي (C:\mysql\bin)، وبذلك يمكن تشغيل البرنامج mysql.exe من داخل الـ Dos.

عموماً طريقة الاتصال بقاعدة البيانات هي كالتالي:

```
mysql –h HostName –u UserName –p
```

مع استبدال الـ HostName باسم السيرفر لديك، سواء كان السيرفر على نفس الجهاز وفي هذه الحالة تكتب localhost، أو أن السيرفر الذي تود الاتصال به ليس على نفس الجهاز وبذلك تكتب المسار الكامل لاسم السيرفر (HostName)، ومع استبدال الـ UserName باسم المستخدم الخاص بالـ Mysql لديك، بعد ذلك سيتم طلب كلمة المرور الخاصة بقاعدة البيانات بعد الضغط على Enter، قم بإدخالها وسيتم فتح الاتصال بالـ Mysql، كما يمكن كتابة mysql فقط ليتم فتح الاتصال بقاعدة البيانات فقط إذا كنت تعمل على نفس الجهاز وليس جهاز أخر.

سيظهر المؤشر الخاص بأوامر الـ Mysql كالتالي:

```
mysql>
```

وبهذا نكون وصلنا إلى المكان المطلوب لكتابة أوامر الـ Mysql والتحكم بها.

الأمر الأول الذي سنقوم بكتابته يقوم باستعراض قواعد البيانات الموجودة على السيرفر والأمر هو:

```
show databases;
```

بعد كتابة هذا الأمر (بعد مؤشر الـ mysql>)، سيتم استعراض قواعد البيانات في السيرفر الذي قمنا بالاتصال به، وفي حالة عدم وجود أي قاعدة بيانات قمت بإعدادها من قبل، فان من الطبيعي أن تجد قاعدتي بيانات موجودة بشكل تلقائي عند تثبيت السيرفر Mysql، وتلك القاعدتان هما test – mysql.

ولمحاولة فهم الموضوع بشكل أكبر، سنقوم بالتطرق إلى مثال يبين كيفية إنشاء قاعدة بيانات، وكيفية الدخول لها والتعامل معها وإنشاء الجداول، ومن ثم حذفها..

بعد استعراض قواعد البيانات بالأمر السابق، سنقوم بانشاء قاعدة بيانات باسم PHP، ولعمل ذلك قم بكتابة الأمر التالي:

```
create database PHP;
```

لو قمنا بكتابة الأمر السابق (show database) سنرى أن قواعد البيانات أصبحت ٣ بإضافة القاعدة PHP إلى القاعدتين test – mysql، ولاستخدام أي منها نقوم بكتابة الأمر التالي في مثالنا مع القاعدة PHP:

```
use PHP;
```

وهذه يعني الدخول في قاعدة البيانات PHP واستخدام المؤشر (mysql>) لكتابة الأوامر المتعلقة بالتعامل مع قاعدة بيانات بعينها.

أول هذه الأوامر هو أمر إنشاء جدول في قاعدة البيانات، وهذه الأمر يحتاج إلى تفصيل دقيق لبعض الخصائص مثل أسماء الحقول وأنواع البيانات فيها، وبعض

الأشياء الأخرى، عموماً قم بكتابة الأمر التالي وسأقوم بشرح كافة التفاصيل بعد المثال:

```
(create table users
 ،id Int not null auto_increment Primary Key
 ،name text not null
counter int
);
```

شرح المثال:

- قمنا بكتابة (create table users) وهذا يعني إنشاء جدول باسم users.
- القوس) يعني بداية تسمية حقول الجدول وخصائص تلك الحقول.
- السطر الأول من أسماء الحقول هو (id) والرمز (int) يعني وصف نوع البيانات التي ستخزن في الحقل (id)، وهي في هذه الحالة تعني نوع البيانات الرقمية، أما الرمز (not null) فيعني عدم إمكانية أن يكون هذا الحقل فارغاً، بل يجب أن يحتوي على قيمة، والـ (auto_increment) يجعل الحقل يحتوي على قيم متسلسلة يستحيل تكرارها، وسيبدأ من الرقم ١ ويبدأ بالزيادة بمقدار واحد في كل مرة يتم إدخال صف جديد إلى هذا الجدول، وفي النهاية الرمز (Primary Key) يعني أن الحقل هو المفتاح الرئيسي لهذا الجدول أو بمعنى إنه سيتم التفريق بين صفوف الجدول اعتماداً على هذا الحقل ولهذا وضعنا (auto_increment) لضمان عدم اختلاط البيانات.
- السطر الثاني يحتوي على اسم الحقل (name) ونوع البيانات (text) أي نصي، ونفس الرمز السابق الذي ذكرناه وهو (not null).
- السطر الثالث يحتوي على اسم الحقل (counter) ونوع البيانات (int)، ولاحظ أننا لم نذكر (not null) وبالتالي يمكن أن يكون هذا الحقل فارغاً لا يحتوي على أي قيمة، ولن يكون هناك أي تعارض أو مشكلة بعكس الحقلين السابقتين.
- في السطر قبل الأخير، أي قبل علامة الإغلاق (، سيكون بدون فاصلة.
- السطر الأخير يحتوي على أقفال عملية إنشاء الجدول بالعلامة)؛.

عموماً هذا المثال يعطي نبذة بسيطة عن كيفية إجراء مثل هذه الأوامر، وسنتطرق إلى بقية الأوامر في الأسطر القليلة القادمة.

يمكنك استعراض الجداول الموجودة في قاعدة بيانات عن طريق الأمر:

```
show tables;
```

ولو قمت بتطبيق ذلك على المثال السابق فسترى أن الجدول users موجود في قاعدة البيانات PHP التي قمنا بإنشائها.

يمكن كذلك استعراض خصائص الجدول السابق users الذي قمنا بإنشائه في المثال السابق، عن طريق الأمر التالي:

```
describe users;
```

سترى أن حقول الجدول وخصائص كل جدول ظهرت لك بشكل واضح.

- التعامل مع بيانات الجداول:

بقي أن نذكر الطرق التي يمكن من خلالها إدخال البيانات إلى الجدول users، بل وكيفية التعامل مع تلك البيانات بالتعديل والحذف وغير ذلك، وكما قلنا سابقاً إن هذه الأساسيات مفيدة جداً في البرمجة بلغة الـ PHP، بل إن فهم هذه الطرق هو المفتاح الأساسي للتعامل مع قواعد البيانات عن طريق الـ PHP،

عموماً إن أول تلك الأوامر هو إضافة صف جديد إلى الجدول، وهذا ما يبينه المثال التالي:

```
insert into users set
name = “Ahmad”;
counter = 3
;
```

مع ملاحظة أن users هو اسم الجدول، name اسم الحقل (العمود) الأول، counter اسم الحقل (العمود) الثاني، كما تلاحظ أن الحقل id لم نتطرق له،

لأننا في إعدادنا للجدول ذكرنا أن الحقل (id auto_increment) أي ستضاف إليه القيم بشكل تلقائي وبشكل منظم، كما قلنا في كل مرة يزيد العداد بقيمة ١، و بطبعية الحال يمكنك القياس على هذا المثال باستبدال ما يجب استبداله من اسم الجدول (users) واسماء الحقول (name – counter) وكذلك البيانات بما يناسب الذي تريد القيام به.

هذا بالنسبة لإضافة بيانات جديدة إلى جدول معين، أما بالنسبة لاستعراض البيانات في الجدول فكما يلي:

```
select * from users;
```

ومعني select (اختر)، ولذلك ستجد أن جميع البيانات التي في الجدول users قد تم سردها، واذا كنت ملتزماً بالمثال السابق حرفياً فستجد أن البيانات التي أضفناها في المثال السابق ظهرت على شكل صف من صفوف الجدول، وبالتالي كلما أضفت صفاً جديداً إلى الجدول وقمت باستعراض البيانات تجد أن بياناتك قد تم تخزينها، وينطبق الكلام السابق حول الاستبدال هنا أيضاً، فيمكن استبدال اسم الجدول users بأي اسم لجدول في قاعدة البيانات المستخدمة، وللتأكد من أسماء الجداول قم باستخدام الطريقة السابق ذكرها وهي (show tables).

النقطة الأخيرة التي سأتطرق لها هي ما يجب معرفته حول الأمر select وهو كثرة استخدامه في التعامل عن طريق الـ PHP، وبالتالي يجب عليك فهم طريقة كتابته بشكل كامل، بالاضافة إلى خيارات الاختيار إن صح التعبير، وهي ما يتم كتابته بعد الجملة السابقة من خيارات تحدد طريقة اختيار البيانات من شروط وترتيب وحدود وهذا ما ساذكره في الاسطر القليلة القادمة.

فلفترض أن الجدول السابق يحتوي على أكثر من صف من البيانات بالشكل التالي:

أما البيانات التي نود جلبها فهي كما يلي لكل نقطة على حدة:

١- بيانات الأعضاء الذين ليس لهم أي موضوع.

٢- بيانات الأعضاء الذين لهم مواضيع أكثر من ٥ مرتبين من الأكثر إلى الأقل.

٣- بيانات العضو Ahmed.

٤- بيانات جميع الأعضاء مرتبين حسب الاسم.

٥- بيانات العضو الأكثر مواضيعاً.

سنأخذ كل حالة على حدة:

الحالة الأولى:

يمكن التعامل معها كما يلي:

```
select * from users where counter=0;
```

الزيادة التي قمنا بوضعها هي (where counter=0) أي بحيث أن الحقل (counter) يساوي صفر، وبالتالي سيتم إهمال أي صف من البيانات التي لا يحتوي الحقل (counter) فيها على القيمة صفر، وسيتم جلب البيانات التي يحتوي هذا الحقل فيها على صفر.

الحالة الثانية:

```
select * from users where counter >= 5 order by counter;
```

في هذا المثال أضفنا الشرط (where counter =< ٥) وهو واضح كما في المثال السابق ولكن تم تغيير الشرط لا أقل ولا أكثر، أما الإضافة الأخرى فهي طريقة الترتيب وهي (order by counter) وتعني (قم بترتيب البيانات المختارة بحسب الحقل counter)، وهناك طريقة أخرى للتحكم في الترتيب اما تصاعدي أو تنازلي وذلك بإضافة كلمة asc ليكون الترتيب تنازلياً كما هو الحال في المثال السابق، فسواء ذكرت ذلك أو سيتم اعتبارها تنازلياً بشكل تلقائي، أما الأهم فهو

طريقة الترتيب التصاعدي من الأقل إلى الأكبر ويتم ذلك عن طريق كتابة الكلمة desc بعد الترتيب مباشرة لتصبح كما يلي:

```
select * from users where counter >= 5 order by counter desc;
```

الحالة الثالثة:

```
select * from users where name = “Ahmed”;
```

لاحظ أن الفرق الوحيد هنا هو استخدام علامات التنصيص، لأن نوع البيانات نصية.

الحالة الرابعة:

```
select * from users order by name;
```

وقد أوردت هذا المثال لبيان أنه يمكن استخدام أحد الخيارات لجلب البيانات وترك باقي الخيارات، فيمكن كما في المثال استخدام خيار الترتيب (order) وعدم استخدام الخيارات الباقية (where – limit)، أما الخيار where فقد تطرقنا له سابقاً وتعرفنا على فائدته، والخيار الأخر limit هو ما سيتم التطرق إليه في المثال التالي الخاص بالحالة الخامسة:

الحالة الخامسة:

```
select * from users order by counter limit 1;
```

والـ limit تعني عدد الصفوف المختارة، أي لو قمنا بكتابة المثال السابق بدون الـ limit ستجد أن جميع البيانات سيتم اختيارها، ولكن باستخدام الـ limit نقوم بتحديد عدد الصفوف التي سيتم اختيارها استناداً إلى طريقة ترتيبنا للبيانات، فكما تلاحظ قمنا بترتيب البيانات بحسب الحقل counter ولم نذكر (desc) ولذلك فالبيانات يتم ترتيبها من الأكبر إلى الأصغر، وبالتالي فاختيارنا للحقل الأول يقضي باختيار بيانات الشخص الأكثر كتابة للمواضيع.

بقي أن نذكر طريقي التعديل والحذف ليكتمل الفصل، وسنبدأ بطريقة التعديل على البيانات الموجودة في الجدول users من قاعدة البيانات PHP، والمثال التالي يوضح الطريقة التي سيتم شرحها بعد المثال:

```
update users set
 ،name = “Naser”
counter = 30
where name=”Ahmad”;
```

الجملة update تعني تحديث أو (قم بتحديث)، والـ users هو اسم الجدول الذي نعمل عليه، وفي السطر الثاني قمنا بإسناد القيمة Naser إلى الحقل name، والسطر الذي يليه قمنا بإسناد القيمة ٣٠ إلى الحقل counter، ولكن لو توقفنا هنا بدون ذكر الصف الذي سيتم التعديل عليه، سيتم تعديل كافة الصفوف في الجدول مهما كان عددها، ولذلك كتبنا في النهاية ”where name=”Ahmad، بمعنى أن التغيرات السابقة ستحدث فقط على الصف من البيانات التي يحتوي فيها الحقل name على القيمة Ahmad.

ربما يكون المثال غير واضح بشكل كافٍ، ولكن مع التمرس والمحاولة ستجد أن المسألة منطقية وواضحة بشكل كبير، عموماً لم يبق لدينا إلا طريقة الحذف، سواء كان لكل البيانات في الجدول، أو لصف معين من البيانات وسنرى ذلك في المثالين التاليين، وهما ما سنختم به هذا الفصل:

```
delete from users;
```

الأمر السابق كفيل بإلغاء جميع الصفوف في الجدول users كما هو واضح، ولذلك كن متأكداً من أن التجارب التي تقوم بها هي على بيانات غير هامة.

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

وهذا الحذف سيتم على الصف الذي يتحقق عليه الشرط، وفي هذه الحالة على الصف من البيانات التي يحتوي فيها الحقل id على القيمة ١.

الدوال [Function]:

يوجد في PHP العديد من الدوال التي تقوم بوظيفة معينة (محددة) كذلك توجد إمكانية إنشاء دوال تؤدي وظيفة خاصة وحديثا هنا عن هذا النوع من الدوال (كيفية إنشاء دوال)

الدالة: تقوم بتنفيذ شئ معين حيث تأخذ (متغيرات – معطيات) ثم تقوم بمعالجة هذه المتغيرات وتخرج قيمة أخرى.

- الشكل العام – التركيب:

```
Function(المعطيات – المتغيرات – البارامتر) اسم الدالة
{
هنا يتم كتابة الكود
Return (المعطيات – المتغيرات – البارامتر) ;
}
```

- تعريف الدالة:

لكي نقوم بتعريف دالة نكتب كلمة function بعدها اسم الدالة وبعد الاسم نكتب المعطيات – المتغيرات بين قوسين.

مثال:

```
<?
Function aa($s)
?>
```

حيث aa هو اسم الدالة، وبالتأكيد يمكن أن يكون أي اسم.

($s) هو (المتغير – المعطى – البارامتر)، أي اسم من هذه كما تحب أن تسميه.

مع ملاحظة عدم وضع فاصلة منقوطة بعد هذا السطر.

بعد ذلك نقوم بكتابة كود الدالة (عمل الدالة) بين العلامتين { } ، كما يجب أن ننهي الدالة بكلمة return لإعلام الدالة بأن وظيفتها قد انتهت بالإضافة إلى ذكر اسم المتغير المذكور في تعريف الدالة سابقاً..

مثال:

```
<?
Return($s) ;
?>
```

- استخدامات الدالة:

يمكن وضع الدالة في أي مكان في شيفرة php في أولها أو آخرها بمعنى أنه يمكن استدعاء دالة تم تعريفها في أخر الشيفرة أو العكس.

- إظهار نتيجة الدالة (طباعة الدالة):

نستخدم الأمر الخاص بالطباعة echo أو print وبعده طبعاً اسم الدالة..

مثال:

```
<?
echo aa(5);
print aa(5);
?>
```

مثال كامل:

```
<?
//تعريف الدالة
function aa($a)
{
$a=$a*$a*$a*$a;
return($a);
}
//طباعة ناتج الدالة عند إدخال الرقم ٥ فيها
echo aa(5);
?>
```

هذه الدالة تقوم بحساب عدد مرفوع لأس أربعة بمعنى أن العدد مضروب في نفسه أربع مرات اسم الدالة aa وعند طباعة مخرجات الدالة لرقم، كتبنا أمر الطباعة قبل اسم الدالة والرقم المراد حساب الأس الرابع له بين قوسين (٥) وهكذا إذا وضعنا أي رقم آخر سوف تقوم الدالة بحساب الأس الرابع للرقم مباشر وفي مثالنا هذا يتم طبع الرقم ٦٢٥.

نقطة أخرى هي أننا قمنا بتمرير قيمة ثابتة إلى الدالة، ولذلك يمكننا أن نمرر للدالة متغير كما في المثال التالي:

```
<?
function as($a)
{
$a=$a*$a*$a*3  ;
return($a) ;
}
$z=10 ;
echo as ($z) ;
?>
```

في هذا المثال تقوم الدالة بضرب العدد في نفسه ثلاث مرات ثم في الرقم ٣، ونلاحظ أننا مررنا المتغير $z إلى الدالة as وكتبناها جميعها في سطر طباعة نتيجة الدالة بالأمر echo. ولذلك تقوم الدالة في هذا المثال بضرب الرقم ١٠ في نفسه ثلاث مرات ثم في ٣ يكون الناتج ٣٠٠٠ ومن ثم يتم طباعة الناتج، وبطبيعة الحال كلما غيرنا قيمة المتغير اختلفت نتيجة الدالة.

- العمليات الرياضية:

هي نفسها العمليات التي درستها في المرحلة الابتدائية من (جمع +، طرح -، ضرب ❖، قسمة /) والزائد عليهم الذي لم تدرسه تقريباً هو باقي القسمة (٪)..

مثال شامل على كل العمليات في الـ PHP:

```
<?
$a = 6;
$b=2;
$c= $a +$b;
//سوف نحصل على ناتج الجمع ٨

$c= $a -$b;
//سوف نحصل على ناتج الطرح ٤

$c= $a * $b;
//سوف نحصل على ناتج الضرب ١٢

$c= $a /$b;
//سوف نحصل على ناتج القسمة ٣

$a = 7;
$b=2;
$c= $a % $b;
//سوف نحصل على باقي القسمة ١
?>
```

- عمليات Assignment:

(=)

احفظ القيمة في المتغير، بمعنى خزن القيمة ٣ في المتغير $a:

```
<?
$a = 3;
print $a;
//يطبع ٣
?>
```

(+=)

إضافة قيمة إلى قيمة في نفس المتغير: Solomon

```
<?
$a = 3;
$a += 3;
print $a;
//يطبع ٦
?>
```

(-=)

اطرح المقدار واحد من المقدار ثلاثة في المتغير $a:

```
<?
$a = 3;
$a -= 1;
print $a;
//يطبع ٢
?>
```

(*=)

يضرب القيمة ٣ بالقيمة ٢ ويكون الناتج مخزن في نفس المتغير:

```
<?
$a = 3;
$a *= 2;
print $a;
//يطبع الناتج ٦
?>
```

(/=)

يقسم قيمة على قيمة أخرى:

```
<?
$a = 6;
$a /= 2;
print $a;
//يطبع ناتج القسمة ٣
?>
```

(.=)

دمج سلسلة حرفية:

```
<?
$a = "This is ";
$a.= "a test.";
print $a;
//يطبع الجملة التالية:
// This is a test.
?>
```

- عوامل الإضافة و الطرح:

لو افترضنا أننا لدينا المتغير $a=3$ و أردنا إضافة واحد إليه بحيث يصبح ٤ أو طرح واحد منه بحيث يصبح ٢ ، لدينا العوامل التالية:

$a++ أرجع قيمة a ثم أضف واحد إليها

++$a أضف واحد إليها ثم أرجع القيمة

$a-- أرجع القيمة ثم اطرح واحد منها

--$a اطرح واحد ثم أرجع القيمة

value++

يتم إضافة واحد إلى الرقم خمسة:

```
<?
$a = 5;
print ++$a;
//يطبع القيمة ٦
?>
```

++value

يرجع القيمة نفسها وفي استخدام ثانٍ تزيد القيمة واحداً:

```
<?
$a = 5;
```

```
print $a++;
//طباعة الرقم ٦
print "<br>";
print $a;
//طباعة الرقم ٥
?>
```

value--

يطرح من القيمة واحداً:

```
<?
$a = 5;
print --$a;
//يطبع الرقم ٤
?>
```

--value

يرجع القيمة نفسها وفي استخدام ثانٍ يطرح منها واحداً:

```
<?
$a = 5;
print $a--;
//يطبع الرقم ٤
print "<br>";
print $a;
//يطبع الرقم ٥
?>
```

- عمليات المقارنة **Comparasion Operators**:

$a == $b المتغيران متساويان..

$a === $b المتغيران متساويان و من نفس النوع..

$a != $b المتغير الأول لا يساوي الثاني..

$a !== $b المتغير الأول لا يساوي الثاني وليس من نفس النوع..

$a < $b أكبر من..

$a > $b أصغر من..

$a =< $b أكبر من أو يساوي..

$a => $b أصغر من أو يساوي..

== (تساوي)

تساوي القيمة المخزنة في المتغير الأول بالقيمة المخزنة في المتغير الثاني:

```
<?
$x = 7;
$y = "7";
 ". $y; تساوي if ($x == $y) print $x. "
 //يطبع ٧ تساوي ٧
?>
```

=== (تساوي ومن نفس النوع)

تساوي القيمة المخزنة في المتغير الأول بالقيمة المخزنة في المتغير الثاني وتكون القيم من نفس النوع (حرفية – عددية):

```
<?
$x = 7;
$y = 7;
if ($x === $y) print $x. " is identical to ". $y;
 is identical to 7 //يطبع ٧
?>
```

!= (لا تساوي)

إذا كانت القيم المخزنة في المتغيرين غير متساوية:

```
<?
$x = 8;
$y = 4;
 ". $y; لا تساوي if ($x != $y) print $x. "
 //يطبع ٨ لا تساوي ٤
?>
```

!== (لا تساوي ولا من نفس النوع)

إذا كانت القيم المخزنة في المتغيرين غير متساوية وليست من نفس النوع:

```
<?
$x = 8;
$y = 9;
 ". $y; ليست من نفس نوع if ($x !== $y) print $x. " i
 //يطبع ٨ ليست من نفس نوع ٩
?>
```

< (أقل من)

مقارنة بين قيمتين واحدة أقل من الأخرى:

```
<?
$x = 5;
$y = 9;
  ". $y; أقل من if ($x < $y) print $x. "
 //يطبع ٥ أقل من ٩
?>
```

> (أكبر من)

مقارنة بين قيمتين واحدة أكبر من الأخرى:

```
<?
$x = 9 ;
$y = 5;
  ". $y; اكبر من if ($x > $y) print $x. "
 //يطبع  ٩ أكبر من ٥
?>
```

<= (أقل من ويساوي)

مقارنة بين قيمتين واحدة أقل من الأخرى أو مساوية لها:

```
<?
$x = 5;
```

```
$y = 5;
if ($x <= $y) print $x;
 //يطبع القيمة ٥
?>
```

(أكبر من ويساوي) =>

مقارنة بين قيمتين واحدة أكبر من الأخرى و مساوية لها:

```
<?
$x = 7;
$y = 5;
if ($x >= $y) print $x;
 //يطبع القيمة ٧
?>
```

العمليات المنطقية Logical Operations:

لكي تكون قيمة الشرط صحيحة فيجب أن تنطبق القواعد التالية الخاصة بكل عامل منطقي على حدة، والعوامل هي:

(and) يجب تحقق الاثنين a and b

(or) يجب تحقق كلاهما أو إحداهما a or b

(Xor) يجب تحقق إحداهما وليس كلاهما a xor b

(!) نفي تحقق الشرط نفي لقيمة a !$a

ملاحظة: يمكن كتابة الـ (and) بالشكل التالي (&) والـ (or) بالشكل التالي (|) والـ (Xor) بالشكل التالي (^)..

And (و)

إذا تحقق الشرطان، بمعنى المتغير الأول يساوي ٧ والمتغير الثاني يساوي ٥ نفذ أمر الطباعة واطبع صحيح:

```
<?
$x = 7;
$y = 5;
if (($x == 7) and ($y == 5)) print "صحيح";
//يتم طباعة صحيح
?>
```

Or (أو)

إذا كان أحد الشرطين صحيح أو الاثنين صحيحين نفذ أمر الطباعة:

```
<?
$x = 7;
$y = 5;
if (($x == 7) or ($y == 8)) print "True";
// True يطبع
?>
```

Xor

إذا تحقق أحد الشرطين وليس الاثنين معاً ينفذ أمر الطباعة:

```
<?
$x = 7;
$y = 5;
if (($x == 7) xor ($y == 8)) print "True";
// True تحقق شرط واحد فقط فيتم طباعة كلمة
?>
```

! (النفي)

إذا كانت جملة الشرط غير صحيحة نفذ أمر الطباعة:

```
<?
$y = 5;
if (! ($y == 10)) print "True";
    يطبع True لأن المتغير القيمة المخزنة فيه غير صحيحة
?>
```

&&

المعامل && لـه نفـس وظيفـة (and) لكـن الاخـتلاف في ترتيـب تنفيـذ أولويـات العمليات:

```
<?
$x = 7;
$y = 5;
if (($x == 7) && ($y == 5)) print "True";
// True يطبع
?>
```

||

المعامل || له نفس وظيفة (or) لكن الاختلاف في ترتيب تنفيذ أولويات العمليات:

```
<?
$x = 7;
$y = 5;
if (($x == 7) || ($y == 5)) print "True";
// True يطبع
?>
```

١- الدالة mysql_connect:

```
integer mysql_connect(string host، string username،
string password);
```

تقوم هذه الدالة بالاتصال مع قاعدة البيانات وتعيد لك رقم يفيدك إذا كان لديك أكثر من اتصال بقواعد البيانات، احتفظ به لاستخدامه في دوال أخرى تالية إذا كان هناك حاجة لذلك كما قلنا، أما الوضع الطبيعي فلا يحتاج إلا إلى الاتصال بالطريقة السابقة فقط وبدون الاحتفاظ بأي رقم، فقط مرر للدالة اسم الخادم واسم المستخدم وكلمة المرور، ولكن يتوجب عليك بعد الانتهاء أن تغلق الاتصال باستخدام الدالة mysql_close

مثال:

```
<?
$link = mysql_connect("db.azzozhsn.f2s.com"، "mag"، "Pass");
?>
```

٢- الدالة mysql_pconnect:

```
integer mysql_pconnect(string host، string username، 
strin password);
```

هذه الدالة تقوم بما تقوم به الدلة السابقة إلا أنه لا يتوجب عليك إغلاق الاتصال، مثال:

```
<?
$link = mysql_pconnect("db.azzozhsn.f2s.com"، "mag"، 
"Pass"); 
?>
```

٣- الدالة mysql_select_db:

```
boolean mysql_select_db(string database، 
integer link);
```

تقوم هذه الدالة باختيار قاعد البيانات المحدد لها. مثال:

```
<?
mysql_select_db(string database، 
integer link); 
?>
```

٤- الدالة mysql_db_query:

```
boolean mysql_db_query(string database، string query، 
integer link);
```

تقوم هذه الدالة بتنفيذ سطر SQL على قاعدة البيانات المفتوحة بالمعطى database مثال:

```
<?
$link = mysql_connect("db.azzozhsn.f2s.com"، "mag"، 
"Pass");  
$Query = "DELETE FROM magazine";
$result = mysql_db_query("mag"، $Query، 
$link); 
?>
```

٥- الدالة mysql_close:

```
boolean mysql_close(integer link);
```

تقوم هذه الدالة بقطع (إغلاق) قاعدة البيانات، مرر لها رقم الاتصال المعاد من الدالة mysql_connect

مثال:

```
<?
//الاتصال بقاعدة البيانات.
$link = mysql_connect("localhost", "mag", "Pass");
//إغلاق الاتصال بقاعدة البيانات.
mysql_close($link);
?>
```

٦- الدالة mysql_query:

```
integer = mysql_query(string query, integer link);
```

تقوم هذه الدالة بما تقوم به الدالة mysql_db_query تقريباً إلا أن الدالة mysql_query يقتصر عملها على قاعدة البيانات المحددة بالدالة mysql_select_db.

في حالة عدم تمرير رقم الاتصال فستعمل الدالة على الاتصال الأخير.

مثال:

```
<?
$link = mysql_connect("localhost", "mag", "Pass");
$query = "DELETE FROM magazine";
$result = mysql_query($query, $link);
?>
```

٧- الدالة mysql_errno:

```
integer mysql_errno(integer link);
```

تقوم هذه الدالة بإعادة رقم آخر خطأ حدث في التعامل مع قاعدة البيانات.

٨- الدالة mysql_error:

```
string mysql_error(integer link);
```

تعيد هذه الدالة رسالة الخطأ الحاصل في قاعدة البيانات.

٩ - الدالة mysql_create_db:

```
integer link);،boolean mysql_create_db(string databasename
```

تقوم هذه الدالة بإنشاء قاعدة بيانات جديدة مرر لها اسم قاعدة البيانات ورقم الاتصال العائد من الدالة mysql_connect أومن الدالة mysql_pconnect..

مثال:

```
<?
//حيث إن الفراغ هو الباسوورد az المتصل بقاعدة بيانات اسمها
"");  ، "az"،$link = mysql_pconnect("localhost"
//إنشاء قاعدة بيانات جديدة
"mag")) ،if (! mysql_create_db($link
{
") فشل إنشاء قاعدة البيانات الجديدة print("
   exit();
}
?>
```

١٠- الدالة mysql_drop_db:

```
integer link);،boolean mysql_drop_db(string databasename
```

تقوم هذه الدالة بحذف قاعدة البيانات المحددة بالمعطى databasename..

١١- الدالة mysql_list_dbs:

```
integer mysql_list_dbs(integer link);
```

تقوم هذه الدالة بإعادة موشر لكل قواعد البيانات الموجودة في الخادم لغرض استعمالها مع الدالة mysql_fetch_row وأمثالها.

١٢- الدالة mysql_field_seek:

```
integer field);،boolean mysql_field_seek(integer result
```

تقوم هذه الدالة بتحديد الحقل الممرر إليها رقمه. مثال:

```
<?
//حيث إن الفراغ هو الباسوورد az المتصل بقاعدة بيانات اسمها
"");  ،"az"،$dbLink = mysql_pconnect("localhost"
// Authors اختيار قاعدة البيانات
$dbLink);  ،mysql_select_db("Authers"
// Adress اختيار جميع الحقول من الجدول
$Query = "SELECT * FROM adress";
$dbLink);  ،$result = mysql_query($Query
//الانتقال إلى الحقل الثاني اعتماداً على عملية الاختيار
1);  ،mysql_field_seek($reslut
?>
```

١٣- الدالة mysql_field_name:

```
integer feild);،string mysql_field_name(integer result
```

تعيد هذه الدالة اسم الحقل المحدد بالرقم الممرر إليها والذي يبدأ بالرقم صفر للحقل (العمود) الأول. مثالها سيأتي بعد قليل.

١٤- الدالة mysql_field_type:

```
integer feild);،string mysql_field_type(integer result
```

تعيد هذه الدالة نوع الحقل المحدد بالرقم الممرر إليها والذي يبدأ بالرقم صفر للحقل (العمود) الأول. المثال سيأتي بعد قليل أيضاً..

١٥- الدالة mysql_field_len:

```
integer feild);،string mysql_field_len(integer result
```

تعيد هذه الدالة طول الحقل بالبايت المحدد بالرقم المرر إليها والذي يبدأ بالرقم صفر للحقل (العمود) الأول. المثال بعد قليل..

١٦- الدالة mysql_field_flags:

```
integer feild);،string mysql_field_flags(integer result
```

تعيد هذه الدالة وصف الحقل المحدد بالرقم المرر إليها والذي يبدأ بالرقم صفر للحقل (العمود) الأول.

١٧- الدالة mysql_list:

```
integer link);، string table،mysql_list(string database
```

المثال الشامل:

```
<?
//حيث إن الفراغ هو الباسوورد az المتصل بقاعدة بيانات اسمها
"");  ، "az"،$link = mysql_pconnect("localhost"
//ترتيب الحقول وجلبها
integer link);  ، "table"،$result = mysql_list_field("mag"
//حلقة تكرار للمرور على كل حقل
for ($a = 0; $a < mysql_field_num($result); $a++)
{
$i);  ،    print(mysql_field_name($result
$i));  ،    print(mysql_field_type($result
$i));  ،    print(mysql_field_len($result
i));  ،    print(mysql_field_flags($result
}
?>
```

١٨- الدالة mysql_fetch_field:

```
<?
 integer field);،object mysql_fetch_field(integer result
?>
```

استخدم هذه الدالة لتحصل على معلومات حول حقول الجدول المراد، الحقول ترقم بدايةً من صفر وصف الحقل مشروح في الجدول التالي:

الخاصة	الوصف
blob	إذا كانت TRUE فالحقل عبارة عن عن حقل بيانات كبير
maxlength	الطول الأقصى للحقل
multiple_key	تكون TRUE إذا كان الحقل مفتاحاً
name	اسم الحقل
not_null	تكون TRUE إذا كان الحقل لا يمكن أن يكون فارغاً
numric	تكون TRUE إذا كان الحقل يرقم تلقائياً
primary_key	تكون TRUE إذا كان الحقل يمثل مفتاحاً رئيساً
unque_key	تكون TRUE إذا كان الحقل يمثل مفتاحاً ثانوياً
zerofill	تكون TRUE إذاكان الحقل يملأ بالقيمة ٠

١٩ - الدالة mysql_fetch_lengths:

```
<?
array mysql_fetch_lengths(integer result);
?>
```

استخدم هذه الدالة لتعيد مصفوفة تحتوي على الطول الأقصى لكل حقل محدد في المعطي result.

```
<?
//Connect to server as azzozhsn no password
"");،"azzozhsn"،$link = mysql_pconnect("localhost"
```

```
//Select th magazine database
$link);،mysql_select_db("magazine"
//Get name and id from magazine
 id FROM magazine';،$Query = 'SELECT name
 $link);،$result = mysql_query($Query
$length = mysql_fetch($result);
//Print length of the third column
print($lengths[2]);
?>
```

٢٠ - الدالة mysql_fetch_array:

```
<?
array mysql_fetch_array(integer result);
?>
```

هذه الدالة تعيد مصفوفة تحتوي على قيم سجل وتنقل المؤشر إلى السجل التالي.

مثال:

```
<?
//Connect to server as azzozhsn no password
"");،"azzozhsn"،$link = mysql_pconnect("localhost"
//Select th magazine database
$link);،mysql_select_db("magazine"
//Get name and id from magazine
 id FROM magazine';،$Query = 'SELECT name
 $link);،$result = mysql_query($Query
//Get every row
 MYSQL_ASSOC)){،while($row=mysql_fetch_array($result
 //Print mane and id
    print({$row["id"]}={$row["name"]});
}
?>
```

٢١- الدالة mysql_fetch_object:

```
<?
object mysql_fetch_object(integer result)
?>
```

هذه الدالة تشبه الدالة mysql_fetch_array إلا أنها تعيد كائن. عند استدعاء الدالة ينتقل المؤشر إلى السجل التالي في الجدول، وإذا وصل إلى نهاية الجدول ثم استدعيت الدالة مرة أخرى فإنها تعيد القيمة FALSE.

مثال:

```
<?
while($row=mysql_fetch_object(result)){
 //print id and name
 $row->name")، print ("$row->id
}
?>
```

٢٢- الدالة mysql_fetch_row:

هذه الدالة تعيد مصفوفة تحتوي على قيم حقول سجل من الجدول وكل استدعاء يعيد قيمة الحقول في السجل التالي في الواقع هذه الدالة تشبه الدالتين السابقتين.

مثال:

```
<?
while($row=mysql_fetch_row(result)){
 //print id and name
 $row[1]")، print ("$row[0]
}
?>
```

٢٣- الدالة mysql_change_user:

```
<?
، string db، string password،musql_change_user(string user
 integer link);
?>
```

استخدم هذه الدالة لتغيير مستخدم قاعدة بيانات المتصل بها. المعطيان db، link اختيارية وفي حالة فقدهما يستعاض عنهما بالاتصال الحالي. هذه الدالة تتطلب إصدار MySQL 3.23.3 أو ما بعدها.