**What is an AUTO INCREMENT Field?**

Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.

Often this is the primary key field that we would like to be created automatically every time a new record is inserted.

**MySQL AUTO\_INCREMENT Keyword**

MySQL uses the AUTO\_INCREMENT keyword to perform an auto-increment feature.

By default, the starting value for AUTO\_INCREMENT is 1, and it will increment by 1 for each new record.

The following SQL statement defines the "Personid" column to be an autoincrement primary key field in the "Persons" table:

CREATE TABLE Persons (

Personid int NOT NULL AUTO\_INCREMENT,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Age int,

PRIMARY KEY (Personid)

);

To let the AUTO\_INCREMENT sequence start with another value, use the following SQL statement:

ALTER TABLE Persons AUTO\_INCREMENT=100;

When we insert a new record into the "Persons" table, we do NOT have to specify a value for the "Personid" column (a unique value will be added automatically):

INSERT INTO Persons (FirstName,LastName) VALUES ('Lars','Monsen');

**MySQL Date Data Types**

MySQL comes with the following data types for storing a date or a date/time value in the database:

* DATE - format YYYY-MM-DD
* DATETIME - format: YYYY-MM-DD HH:MI:SS
* TIMESTAMP - format: YYYY-MM-DD HH:MI:SS
* YEAR - format YYYY or YY

**Note:** The date data type are set for a column when you create a new table in your database!

**Working with Dates**

Look at the following table:

**Orders Table**



Now we want to select the records with an OrderDate of "2008-11-11" from the table above.

We use the following SELECT statement:

SELECT \* FROM Orders WHERE OrderDate='2008-11-11' The result-set will look like this:



**MySQL CREATE VIEW Statement**

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

A view is created with the CREATE VIEW statement.

# CREATE VIEW Syntax

CREATE VIEW *view\_name* AS SELECT *column1*, *column2*, ...

FROM *table\_name*

WHERE *condition*;

**MySQL CREATE VIEW Examples**

The following SQL creates a view that shows all customers from Brazil:

|  |
| --- |
| **Example** |
| CREATE VIEW [Brazil Customers] AS |

SELECT CustomerName, ContactName

FROM Customers

WHERE Country = 'Brazil';

We can query the view above as follows:

|  |
| --- |
| **Example** |
| SELECT \* FROM [Brazil Customers]; |

The following SQL creates a view that selects every product in the "Products" table with a price higher than the average price:

|  |
| --- |
| **Example** |
| CREATE VIEW [Products Above Average Price] AS |

SELECT ProductName, Price

FROM Products

WHERE Price > (SELECT AVG(Price) FROM Products);

We can query the view above as follows:

**Example**

SELECT \* FROM [Products Above Average Price];

**MySQL Updating a View**

A view can be updated with the CREATE OR REPLACE VIEW statement.

# CREATE OR REPLACE VIEW Syntax

CREATE OR REPLACE VIEW *view\_name* AS

SELECT *column1*, *column2*, ...

FROM *table\_name*

WHERE *condition*;

The following SQL adds the "City" column to the "Brazil Customers" view:

**Example**

CREATE OR REPLACE VIEW [Brazil Customers] AS

SELECT CustomerName, ContactName, City

FROM Customers

WHERE Country = 'Brazil';

**MySQL Dropping a View**

A view is deleted with the DROP VIEW statement.

# DROP VIEW Syntax

DROP VIEW *view\_name*;

The following SQL drops the "Brazil Customers" view:

|  |
| --- |
| **Example** |
| DROP VIEW [Brazil Customers]; |