**MySQL**

**Part 1**

**Part 1**

Data types for columns

\*When choosing a data type for a column that will contain a string, use VARCHAR over CHAR\*

\*tables that have a lot of NULL values suggest a poor design – make sure you specify NOT NULL when choosing column properties\*

\*make any numeric type UNSIGNED if it won’t ever store negative numbers such as; user\_id (primary key), zip code, telephone numbers\*

Example:

|  |  |
| --- | --- |
| **Column name** | **Type** |
| User\_id | MEDIUMINT UNSIGNED NOT NULL |
| First\_name | VARCHAR(20) NOT NULL |
| Last\_name | VARCHAR(40) NOT NULL |
| Pass | CHAR(40) NOT NULL |
| Reg\_date | DATETIME NOT NULL |

Data types:

|  |  |  |
| --- | --- | --- |
| **Type** | **Size** | **Description** |
| CHAR(length) | Length bytes | A fixed length field from 0-255 characters long |
| VARCHAR (length) | String length + 1 or 2 bytes | A variable-length field from 0-65,535 characters long |
| TINYTEXT | String length + 1 bytes | A string with a max length of 255 characters |
| TEXT | String length + 2 bytes | A string with a max length of 65,535 characters |
| MEDIUMTEXT | String length + 3 bytes | A string with a max length of 16,777,215 characters |
| LONGTEXT | String length + 4 bytes | A string with a max length of 4,294,967,295 characters |
| TINYINT(length) | 1 byte | Range of -128 to 127 or 0-255 unsigned |
| SMALLINT(length) | 2 bytes | Range of -32,768 to 32,767 or 0 to 65,535 unsigned |
| MEDIUMINT(length) | 3 bytes | Range of -8,388,608 to 8,388,607 or 0 to 16,777,215 unsigned |
| INT(length) | 4 bytes | Range of -2,147,483,648 to 2,147,483,647 or 0 to 4,294,967,295 |
| BIGINT(length) | 8 bytes | Range of -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 or 0 to 18,446,744,073,709,551,615 unsigned |
| FLOAT(length, decimals) | 4 bytes | A small number with a floating decimal point |
| DOUBLE(length, decimals) | 8 bytes | A large number with a floating decimal point |
| DATE | 3 bytes | In the format YYYY-MM-DD |
| DATETIME | 8 bytes | In the format of YYYY-MM-DD HH:MM:SS |
| TIMESTAMP | 4 bytes | In the format of YYYYMMDDHHMMSS; acceptable range starts in 1970 and ends in the year 2038 |
| TIME | 3 bytes | In the format of HH:MM:SS |
| ENUM | 1 or 2 bytes | Short for enumeration, which means that each column can have one of several possible values |
| SET | 1,2,3,4 or 8 bytes | Like ENUM except that each column can have more than one of several possible values |

How to create a database

CREATE DATABASE database name

CREATE it also used for making tables:

CREATE TABLE tablename(

Column1name description,

Column2name description

…)

\*capitalise SQL keywords – although this is not necessary, it makes MySQL easier to read\*

\*You should terminate every SQL command with a semi-colon\*

To tell MySQL what database you want to work with:

USE databasename

Creating a database example:

CREATE DATABASE sitename;

USE sitename;

Creating tables example:

CREATE TABLE users(

User\_id MEDIUMINT UNSIGNED NOT NULL AUTO-INCREMENT,

First\_name VARCHAR(20) NOT NULL,

Last\_name VARCHAR(40) NOT NULL,

Email VARCHAR(60) NOT NULL,

Pass CHAR(40) NOT NULL,

Registration\_date DATETIME NOT NULL,

PRIMARY KEY(user\_id)

);

To confirm the existence of the table:

SHOW TABLES;

SHOW COLUMNS FROM users;

Inserting records

After a database and its tables have been created, you can start populating them using the INSERT command. 2 ways:

1. INSERT INTO tablename(column1, column2…) VALUES(value1,value2);
2. INSERT INTO tablename VALUES(value1, value2, NULL, value3);

The first way adds rows of records to only the columns specified. A column name not specified will be given the value NULL (or default value if one was given). The second way add rows of records in the order that columns appear in the table specified. You have to specify a value for every column, even it NULL.

\*\*The first way is considered better practice – so use this!\*\*

MySQL also allows you to insert multiple rows at one time, separating each record with a comma:

INSERT INTO tablename(column1, column4)

VALUES(valueA, valueB),

(valueC, valueD),

(valueE, valueF);

*Important to note:*

* Numeric values should not be quoted
* String values (for CHAR, VARCHAR and TEXT column types) must always be quoted
* Data and time values must always be quoted
* Functions cannot be quoted
* The word NULL must not be quoted

Insert data example:

INSERT INTO users (first\_name, last\_name, email, pass, registration\_date)

VALUES (‘Ali’, ‘Issaee’, ‘email@example.com’, SHA1(‘mypass’), NOW());

The SHA1() function is used to encrypt data and is always 40 characters long (which is why password is defined as CHAR(40)). This is useful when storing sensitive data. However, you cannot decrypt so don’t use on data you may need to see again e.g. credit card numbers.

The NOW() function returns the current data and time (on the server).

**Part 2**

Selecting data

SELECT columns FROM tablename

To select all columns:

SELECT \* FROM tablename

Using conditionals

Conditionals allow you to limit what rows are returned using the WHERE command

SELECT columns FROM table WHERE condition(s)

|  |  |
| --- | --- |
| = | Equals |
| < <= > >= | Less than, less than and equal to, greater than, greater than and equal to |
| != | Not equal to |
| IS NOT NULL | Has a value |
| IS NULL | Does not have a value |
| IS TRUE | Has a true value |
| IS FALSE | Has a false value |
| BETWEEN | Within a range |
| NOT BETWEEN | Outside of a range |
| IN | Within a list of values (use this instead of multiple or statements) |
| NOT IN | Not found within a list of values |
| OR or || | Where at least one of two conditionals are true |
| AND or && | Where both conditionals are true |
| NOT or ! | Where the condition is not true |
| XOR | Where only one of two conditions is true |

Example:

SELECT \* FROM items WHERE

(price BETWEEN 10.00 AND 20.00) AND

(quantity > 0);

SELECT \* FROM cities WHERE

(zip\_code = 90210) OR (zip\_code = 90211);

SELECT \* FROM cities WHERE zip\_code IN(90210, 90211);

Using LIKE and NOT LIKE

This is used primarily with strings in conjunction with 2 wildcards:

* ( \_ ) – which matches a single character
* ( % ) – which matches zero or more characters

SELECT first\_name, last\_name FROM users WHERE

Email NOT LIKE ‘%@hotmail.com’;

Sorting query requests

Use ORDER BY to order results. Use DESC to return reverse order

SELECT \* FROM tablename

ORDER BY column DESC

\*ORDER BY should come after any conditions\*

Limiting query results

1. LIMIT x
2. LIMIT x, y
3. LIMIT 5 – shows 5 records
4. LIMIT 10, 10 – shows 10 records, starting at record 10

\*LIMIT should always come last in a query!\*

Updating data

To update data, e.g. is email changes, then use:

UPDATE tablename

SET columnname = ‘new data’

WHERE columnname = ‘record you want to change’

\*If you do not use a WHERE clause, the changes would be applied to every record in the column specified\*

Make sure to confirm where the change is made:

SELECT \* FROM users WHERE condition(s).

Full example:

UPDATE users

SET email = ‘mike@authors.com’

WHERE user\_id = 18;

SELECT \* FROM users WHERE user\_id = 18 //confirm update

Deleting data

DELETE FROM tablename WHERE condition(s)

**Part 3**

Using functions

|  |  |
| --- | --- |
| **Text functions** | |
| **Function name** | **Desc** |
| CONCAT() | Concatenate (returns string) |
| CONCAT\_WS() | Concatenate with separator => CONCAT\_WS(sep,t1,t2,t3) =>t1sept2sept3sep |
| LENGTH(t) | Number of characters in *t* |
| LEFT(x,y) | The leftmost y characters from *t* |
| RIGHT(t,x) | The rightmost x characters from *t* |
| TRIM(t) | *t* with excess spaces from the beginning and end removed |
| UPPER() | Capitalise |
| LOWER() | Lower case |
| REPLACE(t1,t2,t3) | The string t1 with instances of t2 replaced with t3 |
| SUBSTRING(t,x,y) | *y* characters from *t* beginning with *x* (indexed from 1) |

|  |  |
| --- | --- |
| **Numeric functions** | |
| **Function name** | **Desc** |
| ABS() | The absolute value |
| CEILING() | The next highest integer |
| FLOOR() | The integer value |
| FOMRAT(n1,n2) | N1 formatted as a number with n2 decimal places and commas insterted every 3 spaces |
| MOD(n1,n2) | The remainder of dividing n1 by n2 |
| RAND() | A random number between 0 & 1.0 |
| ROUND(n1,n2) | N1 rounded to n2 decimal places |
| SQRT() | The square root |

|  |  |
| --- | --- |
| **Date & time functions** | |
| **Function name** | **Desc** |
| DATE(dt) | The date value of dt |
| HOUR(dt) | The hour value of dt |
| MINUTE(dt) | The minute value of dt |
| SECOND(dt) | The second value of dt |
| DAYNAME(dt) | The name of the day for dt |
| DAYOFMONTH(dt) | The numeric day value of dt |
| MONTHNAME(dt) | The name of the month of dt |
| MONTH(dt) | The numerical math value of dt |
| YEAR(dt) | The year value of dt |
| CURDATE() | The current date |
| CURTIME() | The current time |
| NOW() | The current data & time |
| UNIX\_TIMESTAMP(dt) | The number of seconds since the epoch until the current moment or until the date specified |
| UTC\_TIMESTAMP(dt) | The number of seconds since the epoch until the current moment or until the date specified, in coordinated universal time (UTC) |

Formatting the data & time

* DATE\_FORMAT(datetime, formatting)
* TIME\_FORMAT(datetime, formatting)

|  |  |
| --- | --- |
| **FORMAT() parameters** | |
| %e | Day of the month (1-31) |
| %d | Day of the month – 2 digits (01-31) |
| %D | Day with suffix (1st – 31st) |
| %W | Weekday bane (Sunday – Saturday) |
| %a | Abbreviated weekday name (Sun – Sat) |
| %c | Month number (1-12) |
| %m | Month number – 2 digits (01-12) |
| %M | Month name (January – December) |
| %b | Month name abbreviated (Jan – Dec) |
| %Y | Year (2017) |
| %y | Year (17) |
| %l (lower case L) | Hour (1-12) |
| %h | Hour – 2 digits (01-12) |
| %k | Hour, 24hr clock (0-23) |
| %H | Hour, 24 hr clock – 2 digits (00-23) |
| %i | Minutes (00-59) |
| %s | Seconds (00-59) |
| %r | Time (8:17:02 pm) |
| %T | Time, 24 hr (20:17:02) |
| %p | AM or PM |

E.g.

SELECT DATE\_FORMAT(NOW(), ‘%M %e, %Y %l: %i’);

* MONTH DD, YYYY – HH:MM

SELECT TIME\_FORMAT(CURTIME(), ‘%T’);

* Display current time using 24hr notation

\*You should almost always use MySQL functions to format any dates coming from the database (as opposed to formatting the dates within PHP after retrieving them from the database) \*

\*The only way to access the date or time on the client (user machine), is to use Javascript. It cannot be done with PHP or MySQL\*

**TEST**

**Part 1**

1. List the most common data types in SQL and define them
2. What property should you give a numeric value which will never store a negative number
3. How do you make sure that your database/tables doesn’t have any NULL values
4. Create a database and one table for it
5. Confirm the existence of the table
6. How do you insert records? Insert multiple records at once
7. What does the SHA1() function do?
8. What does the NOW() function do?

**Part 2**

1. How do you select data from a table
2. List all conditionals and explain what they do
3. What keyword is used for condition statements
4. Give example of LIKE & NOT LIKE
5. What are the 2 wildcards associated with LIKE & NOT LIKE and what do they do
6. How do you order results
7. How do you limit your search
8. Update a record in the table and confirm update
9. Delete a record from table

**Part 3**

1. Give 3 examples of text functions
2. Give 3 examples of numeric functions
3. Give 3 examples of date & time functions
4. How do you format the date & time
5. Give examples using parameter table above, to format the date and time in different ways