Week 2 SQL Tutorial

## Basic Wildcard Text Searching

Output the first names of all employees whose first name begins with a ‘J’:

SELECT DISTINCT first\_name FROM employee WHERE first\_name LIKE 'J%';

Output the first names of all employees whose first name begins with a ‘J’, followed by any two characters, followed by ‘nne’:

SELECT DISTINCT first\_name FROM employee WHERE first\_name LIKE 'J\_\_nne'

Output the first names of all employees whose first name has the string ‘anne’ ANYWHERE in it:

SELECT DISTINCT first\_name FROM employees WHERE first\_name LIKE '%ANNE%';

## Boolean Operations

* AND Displays a record if **all** the conditions separated by AND is TRUE
* OR Displays a record if **any** of the conditions separated by OR is TRUE
* NOT Displays a record if the condition is **NOT** TRUE

## More on Boolean Operations and Order of Operation

* AND comes before OR
* Parenthesis comes before anything else

To return the current salary (where the ‘to\_date’, or end\_date is not yet listed) for employee number 10001:

SELECT emp\_no, salary FROM salary WHERE emp\_no = 10001 AND to\_date IS NULL;

To return the current salary (where the ‘to\_date’, or end\_date is not yet listed) for employees numbered 10001 and 10003:

SELECT emp\_no, salary FROM salary WHERE (emp\_no = 10001 OR emp\_no = 10003) AND to\_date IS NULL;

Note that without the parenthesis, the result returned would be WRONG! Why? Because of the order in which the Boolean operators are evaluated (And BEFORE OR).

## Using Operators

To return the salary employee number 10001 would receive if we gave her a 3% raise:

SELECT emp\_no, (salary\*1.03) FROM salary WHERE emp\_no = 10001 AND to\_date IS NULL;

To determine which employees would still make under $100,000 if we gave everyone a 3% raise over their current salary:

SELECT emp\_no FROM salary WHERE to\_date IS NULL AND salary\*1.03 < 100000;

Note: making sure that ‘to\_date’ is NULL assures that we are looking at everyone’s CURRENT salary. (Salaries that have a ‘to\_date’ specified are salaries that employees have earned in the PAST).

Output everyone’s total compensation if we gave a $3,000 bonus:

SELECT (salary + 3000) FROM salary;

Output the percentage increase in total compensation that $3,000 bonus would represent for each person:

SELECT emp\_no, (3000/salary)\*100 FROM salary WHERE to\_date IS NULL;

Round the result obtained above to the whole percentage point:

SELECT ROUND((3000/salary)\*100) FROM salary;

Output a list of employee numbers of the employees for whom a $3,000 bonus would represent a 2% or less increase in overall compensation:

SELECT emp\_no FROM salary WHERE (3000/salary) < .02;

Other math functions can be found in the online MySQL 5.1 documentation at <http://dev.mysql.com/doc/refman/5.1/en/mathematical-functions.html>

## Dealing with Dates

MySQL built-in function to return the current date:

SELECT CURRENT\_DATE();

MySQL built-in function to return the current date and time:

SELECT NOW();

Output every employee’s age:

SELECT first\_name, last\_name, FLOOR(DATEDIFF(CURRENT\_DATE(), birth\_date)/365) FROM employee

Analysis: This statement is somewhat complicated, so let’s break it down into steps:

1. DATEDIFF is a built-in MySQL function that returns the difference in days between two dates – in this case, between the current date (courtesy of MySQL’s CURRENT\_DATE function) and the employee’s birth date.
2. Step 1 result is returned in days. We normally talk about age in years, however, so we need to convert days to years. We do this by dividing the day results by 365 days/year.
3. We are still not done because we have a decimal result and people typically report their ages in a whole number of years. We could use the ROUND() function, but that would report someone who is over halfway to their next birthday as a year older, which is not how ages are reported. Instead, we use the FLOOR() function, which drops everything but the integer, which is what we want.

**Adding and Subtracting Time Intervals**

Find the date 10 months ago:

SELECT DATE\_SUB(NOW(), INTERVAL 10 MONTH)

The interval can be DAY, WEEK, YEAR, etc. (See <https://www.w3schools.com/sql/func_mysql_date_sub.asp> for more options)

Note that the interval is singular. This is a common source for errors.

Find the date 26 weeks in the future:

SELECT DATE\_ADD(NOW(), INTERVAL 26 WEEK)

**Timestampdiff function**

Syntax: TIMESTAMPDIFF(unit,datetime\_expr1,datetime\_expr2);

Units can be FRAC\_SECOND (microseconds), SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, or YEAR.

Example:

SELECT timestampdiff(YEAR, '1971-03-28', NOW());

+------------------------------------------+

| timestampdiff(YEAR, '1971-03-28', NOW()) |

+------------------------------------------+

| 47 |

+------------------------------------------+

1 row in set (0.00 sec)

Timestampdiff returns an integer equal to the whole interval value between date 1 and date2, and is great for simplifying age calculations.

**Day of Week Calculations**

Output the day of the week that employee ‘10001’ was born on.

SELECT DAYOFWEEK(birth\_date) FROM employee WHERE emp\_no = ‘10001’;

Note that 1 = Monday, 2 = Tuesday, etc.

Many other date and time functions are available. Consult the online manual at <http://dev.mysql.com/doc/refman/5.1/en/date-and-time-functions.html> for more information.

Capitalization:

* SQL reserve words (SELECT, FROM, WHERE, etc.) are case-insensitive
  + Select \* from employee = SELECT \* FROM employee = sELecT \* from employee
* Tables and databases are case sensitive ONLY if the underlying operating system is
  + We are using Linux and as such MySQL database and table names ARE case-sensitive
  + “Select \* from EMPLOYEE” will prompt an error that table employees.employee does not exist
  + If we were using Windows (as our MySQL hosting system on the VM, that is), this would be fine
* Attribute names are NOT case-sensitive:
  + “SELECT emp\_no” is exactly the same as “SELECT EMP\_NO”
* The case-sensitivity of your data depends on the database’s “collation type”
  + Collation sets the conditions for text comparison
  + The default collation, utf8\_general\_ci, is case-insensitive (hence the “ci”) and thus “ABC” = “abc”
  + You can change to a “cs” (case sensitive) collation if you need case sensitivity in your application
  + See the [MySQL 5.5 online documentation](http://dev.mysql.com/doc/refman/5.5/en/charset-database.html) for more information on collation