National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

CS-220: Database Systems

Class: BSCS-6C

Lab11 – Logical Operators & Basic SQL Functions

Date: 30th Nov, 2017

Time: 09:00am-12:00 pm

Instructor: Dr.Sharifullah Khan

Lab Engineer: Ms Sadia Amir

National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Querying Relational Database

Introduction

Structured Query Language (SQL) is a high level query language which has inbuilt operators and functions for different presentation/manipulation of the data that is retrieved.

Objectives

After performing this lab students should be able to:

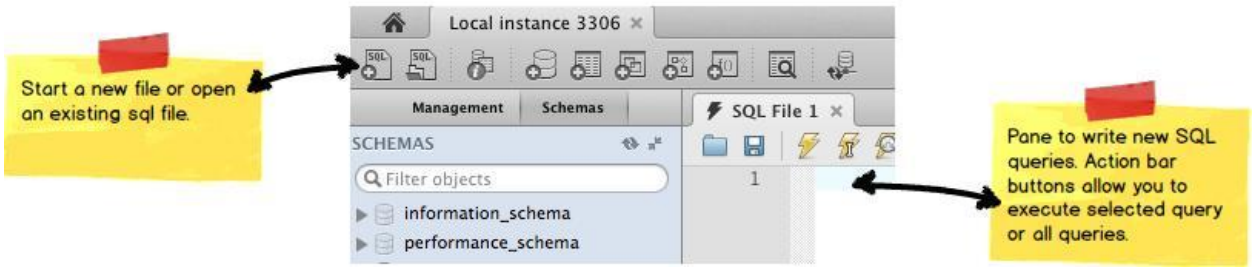
1. Design SQL queries to retrieve data using SELECT clause and using logical operators.
2. Explore various inbuilt single row functions of SQL.

Tools/Software Requirement

- MySQL Community Server 5.6
- MySQL Workbench 6.1
- Sakila Database

Description

1. Open MySQL Workbench and open the default connection instance.
2. A new query window would open from where you can write and execute queries.



3. You can save the query file and can also add comments using //, /* */ symbols.
4. On executing queries, results are displayed in the lower part of the screen.
5. Error or success messages are displayed in action output pane at the bottom.
6. Continue playing with the Workbench and SQL queries till you are comfortable with the querying mechanism and have learnt the shortcuts to execute queries.

This lab is about querying databases. This lab will cover SQL keywords, functions and logical operators. You will execute these queries using Sakila database. Modify the examples wrt Sakila database and practice all operators/functions on the data in Sakila.

ORDER BY

The SQL ORDER BY clause allows you to sort the records in your result set. The SQL ORDER BY clause can only be used in **SQL SELECT statements**.

The syntax for the SQL ORDER BY clause is:

```
SELECT columns
FROM tables
WHERE predicates
ORDER BY column ASC/DESC;
```

The SQL ORDER BY clause sorts the result set based on the columns specified. If the ASC or DESC value is omitted, it is sorted by ASC.

ASC indicates ascending order. (default)
DESC indicates descending order.

Example 1:

```
SELECT supplier_city
FROM suppliers
WHERE supplier_name = 'IBM'
ORDER BY supplier_city;
```

Example 2:

When sorting your result set in descending order, you use the DESC attribute in your ORDER BY clause as follows:

```
SELECT supplier_city
FROM suppliers
WHERE supplier_name = 'IBM'
ORDER BY supplier_city DESC;
```

This SQL ORDER BY example would return all records sorted by the supplier_city field in descending order.

SQL ORDER BY - Using both ASC and DESC attributes together

When sorting your result set using the SQL ORDER BY clause, you can use the ASC and DESC attributes in a single **SQL SELECT statement**.

For example:

```
SELECT supplier_city, supplier_state
FROM suppliers
WHERE supplier_name = 'IBM'
ORDER BY supplier_city DESC, supplier_state ASC;
```

This SQL ORDER BY would return all records sorted by the supplier_city field in descending order, with a secondary sort by supplier_state in ascending order.

SQL Operator Precedence

1. Parentheses - if you group conditional SQL statements together by parentheses, SQL Server evaluates the contents of these first.
2. Arithmetic - multiplication (using the operators *, /, or %)
3. Arithmetic - addition (using the operators + or -)
4. Other - String Concatenate (+)
5. Logical - NOT
6. Logical - AND
7. Logical - OR

Note: (The order of evaluation at the same precedence level is from left to right.)

1. Execute the following queries using the Sakila schema.

```
Select customer_id, 100*amount-10
from payment;

Select customer_id, 100*(amount-
10) from payment;
```

Logical - AND

The AND condition allows you to create an SQL statement based on 2 or more conditions being met. It can be used in any valid SQL statement - select, insert, update, or delete.

The syntax for the AND condition is:

```
SELECT columns
FROM tables
WHERE column1 = 'value1'
AND column2 = 'value2';
```

The AND condition requires that each condition be must be met for the record to be included in the result set. In this case, column1 has to equal 'value1' and column2 has to equal 'value2'.

Example #1:

The first example that we'll take a look at involves a very simple example using the AND condition.

```
SELECT *
FROM suppliers
WHERE city = 'New York'
and type = 'PC Manufacturer';
```

This would return all suppliers that reside in New York and are PC Manufacturers. Because the * is used in the select, all fields from the supplier table would appear in the result set.

Logical - OR

The OR condition allows you to create an SQL statement where records are returned when any one of the conditions are met. It can be used in any valid SQL statement - select, insert, update, or delete.

The syntax for the OR condition is:

```
SELECT columns
FROM tables
WHERE column1 = 'value1'
or column2 = 'value2';
```

The OR condition requires that any of the conditions be must be met for the record to be included in the result set. In this case, column1 has to equal 'value1' OR column2 has to equal 'value2'.

Example #1:

The first example that we'll take a look at involves a very simple example using the OR condition.

```
SELECT *
FROM suppliers
WHERE city = 'New York'
or city = 'Newark';
```

This would return all suppliers that reside in either New York or Newark. Because the * is used in the select, all fields from the suppliers table would appear in the result set.

Example #2:

The next example takes a look at three conditions. If any of these conditions is met, the record will be included in the result set.

```
SELECT supplier_id
FROM suppliers
WHERE name = 'IBM'
or name = 'Hewlett Packard'
or name = 'Gateway';
```

This SQL statement would return all supplier_id values where the supplier's name is either IBM, Hewlett Packard or Gateway.

Logical Operator Precedence

Logical operators are executed in the following specified order.

1. Logical - NOT
2. Logical - AND
3. Logical - OR

Combination of And & Or

The AND and OR conditions can be combined in a single SQL statement. It can be used in any valid SQL statement - select, insert, update, or delete.

When combining these conditions, it is important to use brackets so that the database knows what order to evaluate each condition.

Example #1:

The first example that we'll take a look at an example that combines the AND and OR conditions.

```
SELECT *
FROM suppliers
WHERE (city = 'New York' and name =
'IBM') or (city = 'Newark');
```

This would return all suppliers that reside in New York whose name is IBM and all suppliers that reside in Newark. The brackets determine what order the AND and OR conditions are evaluated in.

Example #2:

The next example takes a look at a more complex statement.

```
SELECT supplier_id
FROM suppliers WHERE
(name = 'IBM')
or (name = 'Hewlett Packard' and city = 'Atlantic City')
or (name = 'Gateway' and status = 'Active' and city = 'Burma');
```

This SQL statement would return all supplier_id values where the supplier's name is IBM or the name is Hewlett Packard and the city is Atlantic City or the name is Gateway, the status is Active, and the city is Burma.

Single Row Functions

What is a function?

A function is similar to an operator in operation. A function is a name that performs a specific task. A function may or may not take values (arguments) but it always returns a value as the result. If function takes values then these values are to be given within parentheses after the function name. The following is the general format of a function.

function [(argument-1, argument-2,...)]

If the function doesn't take any value then function name can be used alone and even parentheses are not required.

Single-row functions return a single result row for every row of a queried table or view. These functions can appear in select lists, WHERE clauses, START WITH and CONNECT BY clauses, and HAVING clauses.

Arithmetic functions perform take numeric data; date functions take date type data and string functions take strings. Conversion functions are used to convert the given value from one type to another. Miscellaneous functions perform operations on any type of data. Group functions are used to perform operations on the groups created by GROUP BY clause.

Character Functions

Character functions operate on values of datatype CHAR or VARCHAR.

LOWER

Returns a given string in lower case.

```
select LOWER(first_name)
from actor;
```

UPPER

Returns a given string in UPPER case.

```
select UPPER(first_name)
from actor;
```

LENGTH

Returns the length of a given string.

```
select length(first_name)
from actor;
```

CONCAT(str1,str2,...)

Returns the string that results from concatenating the arguments. May have one or more arguments.

```
SELECT CONCAT('My', 'S', 'QL');
+                                                    +
| CONCAT('My', 'S', 'QL')                            |
                                                     +
MySQL
```

CONCAT_WS(separator,str1,str2,...)

CONCAT_WS() stands for Concatenate With Separator and is a special form of CONCAT(). The first argument is the separator for the rest of the arguments. The separator is added between the strings to be concatenated. The separator can be a string, as can the rest of the arguments. If the separator is NULL, the result is NULL.

```
SELECT CONCAT_WS(',','First name','Last Name' );
+                                                          +
| CONCAT_WS(',','First name','Last Name' )                 |
-----------------------------------------------------------+ +
| First name, Last Name                                    |
```

LPAD(str,len,padstr)

Returns the string str, left-padded with the string padstr to a length of len characters. If str is longer than len, the return value is shortened to len characters.

```
SELECT LPAD('hi',4,'??');
+                                                          +
| LPAD('hi',4,'??')                                        |
-----------------------------------------------------------+ +
| ??hi                                                     |
-----------------------------------------------------------+ +
```

RPAD(str,len,padstr)

Returns the string str, right-padded with the string padstr to a length of len characters. If str is longer than len, the return value is shortened to len characters.

```
SELECT RPAD('hi',5,'?');
+                                                          +
| RPAD('hi',5,'?')                                         |
-----------------------------------------------------------+ +
| hi???                                                    |
+---------------------------------------------------------- +
```

LTRIM(str)

Returns the string str with leading space characters removed.

```
SELECT LTRIM('  lecture');
```

```
-------------------------------------------------------+ +
| LTRIM('  lecture')                                    |
-------------------------------------------------------+ +
| lecture                                               |
-------------------------------------------------------+ +
```

REPEAT(str,count)

Returns a string consisting of the string str repeated count times. If count is less than 1, returns an empty string. Returns NULL if str or count are NULL.

```
SELECT REPEAT('MySQL', 3);
+                                                      +
| REPEAT('MySQL', 3)                                   |
--------------------------------------------------------+ +
| MySQLMySQLMySQL                                      |
--------------------------------------------------------+ +
```

RTRIM(str)

Returns the string str with trailing space characters removed.

```
SELECT RTRIM('barbar   ');
+                                                     +
| RTRIM('barbar   ')                                  |
-------------------------------------------------------+ +
| barbar                                              |
-------------------------------------------------------+ +
```

SUBSTRING(str,pos)
SUBSTRING(str FROM pos)
SUBSTRING(str,pos,len)
SUBSTRING(str FROM pos FOR len)

The forms without a len argument return a substring from string str starting at position pos. The forms with a len argument return a substring len characters long from string str, starting at position pos. The forms that use FROM are standard SQL syntax. It is also possible to use a negative value for pos. In this case, the beginning of the substring is pos characters from the end of the string, rather than the beginning. A negative value may be used for pos in any of the forms of this function.

```
SELECT SUBSTRING('Quadratically',5);
+                                                      +
| SSUBSTRING('Quadratically',5)                        |
----------------------------------------------------------+ +
| ratically                                            |
----------------------------------------------------------+ +
> SELECT SUBSTRING('foobarbar' FROM 4);
----------------------------------------------------------+ +
| SUBSTRING('foobarbar' FROM 4)                        |
----------------------------------------------------------+ +
| barbar                                               |
----------------------------------------------------------+ +
> SELECT SUBSTRING('Quadratically',5,6);
----------------------------------------------------------+ +
| SUBSTRING('Quadratically',5,6)                       |
----------------------------------------------------------+ +
| ratica                                               |
----------------------------------------------------------+ +
```

SUBSTRING_INDEX(str,delim,count)

Returns the substring from string str before count occurrences of the delimiter delim. If count is positive, everything to the left of the final delimiter (counting from the left) is returned. If count is negative, everything to the right of the final delimiter (counting from the right) is returned. SUBSTRING_INDEX() performs a case-sensitive match when searching for delim.

```
SELECT SUBSTRING_INDEX('www.mysql.com', '.', 2);
+-----------------------------------------------------+
```

```
| SUBSTRING_INDEX('www.mysql.com', '.', 2)                |
+                                                         +
| www.mysql                                               |
-----------------------------------------------------------+ +
```

Arithmetic Operators:

Arithmetics operators and functions are available in MYSQL which are used for arithmetic operations. You can explore them online.

LAB TASKS

Given the following database schema:

Student (snum: integer, sname: char(30), major: char(25), level: char(2), age: integer)

Faculty (fid: integer, fname: char(30), deptid: integer)

Class (cname: char(40), meets_at: char(20), room: char(10), fid: integer | fid REFS Faculty.fid)

Enrolled (snum: integer, cname: char(40) | snum REFS student.snum, cname REFS class.name)

Data of the updated Enrolled table is given in text format and also loaded at LMS. Write SQL expressions for each of the following queries and execute them, Use **like** operator for string matching and **Join** cluase:

1. Add new fields: dateFrom and dateTo in the enrolled table.
2. Insert data in enrolled table.
3. Print the level in lower case and the average age of those students who enrolled in 2008 for that level for all levels that include 'R'.
4. Find the names of all classes (initial capital) and their enrollment strength that have enrollment greater than 2.
5. Find the names of faculty members (in upper case) and their department number together who teach in room 'R128' in descending order.
6. Fin the names of students who enrolled in June 2010 and majoring in any 'Science' in ascending order.
7. Find the names of faculty members that either teach to class 'database systems'or not.
8. Find distinct student ages of those students who completed the course in the year in which he enrolled the couse, in 'Database' class.
9. Find the name of students and completion month and year who enrolled in 'Data Structures'.
10. Find class names ether taught by faculty member: 'Barbara Wilson' or not
11. Find the name of teacher who does not teach to class: 'Marketing Research'.
12. Find the names of faculty members that taught to class 'database Systems' in 2009.

13. Find the name of course that was run only for three months.
14. Find the names of students who enrolled but could not complete their courses.
15. Find the names of students who completed their course in June.

Deliverable

Submit a PDF document including the SQL queries to answer above-mentioned information needs as well as snapshot of their outcome when executed over MySQL using the Workbench.