COMP 3311
Database Management Systems

Lab 2

Basic SQL Statements

Lab Objectives

- After this lab you should:
  - Know how to execute simple SQL commands in SQL Developer.
  - Know how to use the SELECT-FROM-WHERE SQL clauses.
  - Know how to use the ORDER BY clause.
  - Know how to use simple Join operations.

Retrieving Records Using The SELECT Statement

- Syntax:

  select * { [distinct] *column* | *expression [alias], ...*} from *table*

- Example: select all the columns from a table

```
select *
from Department;
```

- Example: select specified columns from a table

```
select departmentId, departmentName
from Department;
```

Removing Duplicates

- The default setting for the SELECT statement is to return all the relevant records – including duplicate ones.

- For example, the following statement will return all the department ids from the Student table:

```
select departmentId
from Student;
```

- To remove duplicates, the DISTINCT keyword can be added to the SELECT statement:

```
select distinct departmentId
from Student;
```

Incorporating Arithmetic Operations In The SELECT Statement

- It is possible to include arithmetic operations like * , / , + , - in a SELECT statement.

- Example:

```
select lastName, cga, cga+2.0
from Student;

select lastName, cga, cga/2.0
from Student;
```

Note: cga/2.0 will return the same result as cga/2 in SQL, this is different from some higher-level languages like C++.

Changing The Name Of A Column Using An Alias

- The column name in the result table can be changed by using the AS keyword.

```
select lastName as ln
from Student;
```

- The SELECT statement can be used to output a column named Quarter CGA which displays the result cga/4.

```
select cga/4 as "Quarter CGA"
from Student;
```

Note: Double quotes are required around an alias if it has an embedded space.

Concatenating Results In The SELECT Statement

- The || operator can be used to concatenate two columns in a select statement.

```
select firstName || ' ' || lastName as "Full Name"
from Student;
```

- The || operator can be used to add a string to the result.

```
select firstName || ' ' || lastName || ' studies in ' || departmentId as "Description"
from Student;
```

Note: If double quotes are placed around a single word alias such as Description, then it is displayed as typed; otherwise the alias name will be displayed in all capital letters.

Example Of Concatenations

- Using concatenation, a query result can be expressed in a more easily comprehensible form.
- For example the output from the table Student can be:

  Ariana Grande (13456789) from the COMP department has CGA 2.83. His/Her email is cs_grande@connect.ust.hk.

- What is the corresponding SELECT statement?

```
select firstName ||' '|| lastName || '(' || studentId || ') ' || 'from the ' || departmentId ||
' department has CGA ' || CGA ||'.' ||' His/Her email is ' || email || '@connect.ust.hk.'
as Lab2
from Student;
```

Using The WHERE Clause

- The WHERE clause, which is always used together with the SELECT clause, is used to select specified rows from a table.

- Syntax: select * | { [distinct] *column* | *expression [alias],...*} from *table* where *condition;*

- For example, the following query retrieves only the COMP department information.

```
select *
from Department
where departmentId='COMP';
```

The string 'COMP' in the condition clause is case sensitive.

Using Comparison Operators In The WHERE Clause

- The comparison operators are

=	equal	>	greater than
>=	greater than or equal	<	less than
<=	less than or equal	<>	not equal

- Examples:

```
select *
from Student
where cga<>2.5;
```

```
select *
from Student
where cga<=1.9;
```

Using Logical Conditions In The WHERE Clause

- Boolean operators
  - AND

```
select *
from Student
where cga>=2 and departmentId='MATH';
```

  - OR

```
select *
from Student
where cga>=2 or departmentId='MATH';
```

  - NOT

```
select *
from Student
where not departmentId='MATH';
```

String Matching In The WHERE Clause (1)

- Pattern matching operators/functions
  - LIKE (for matching characters)
    - % can match zero or more characters.

```
select *
from Student
where firstName like '%u%';
```

    - _ matches exactly one character.

```
select *
from Student
where firstName like '_u%'
```

String Matching In The WHERE Clause (2)

- Pattern matching operators/functions
  - REGEXP_LIKE (for matching patterns)

    Syntax: REGEXP_LIKE(*attribute-name*, *regular-expression*, *match-parameter*)

    - Match students with a double vowel in their last name

      'i' → case insensitive

      'c' → case sensitive.

```
select *
from Student
where regexp_like(lastName, '([aeiou])\1', 'i');
```

More Operators For Conditions (1)

- Range of values operators
  - BETWEEN / NOT BETWEEN

```
select *
from Student
where cga between 2.8 and 3;
```

NOTE: Reversing the order of 3 and 4 will give no result!

- Set membership operators
  - IN / NOT IN

```
select *
from Student
where departmentId in ('ELEC', 'MATH');
```

More Operators For Conditions (2)

- Null value operator
  - IS NULL

```
select *
from Student
where cga is null;
```

NOTE: Cannot use `where cga=null`.

Changing Precedence Using Parentheses

- The AND condition has higher precedence than the OR condition.
  - Selects students from the COMP department *plus* students from the MATH department with cga>=3:

```
select *
from Student
where departmentId='COMP' or departmentId='MATH' and cga>=3;
```

  - To select students with cga>=3, from either the 'COMP' or the 'MATH' departments, add parentheses.

```
select *
from Student
where (departmentId='COMP' or departmentId='MATH') and cga>=3;
```

The ORDER BY Clause (1)

- Sort the result by one or more columns.
  - ASC ascending order (default)

```
select *
from Student
order by cga;
```

  - DESC descending order

```
select *
from Student
order by cga desc;
```

The ORDER BY Clause (2)

- Sort by an alias

```
select firstName, cga, cga*0.8 as wcga
from Student
order by wcga;
```

- Sort by multiple columns

```
select *
from Student
order by departmentId asc, firstName desc;
```

Cartesian Product And Join Operation

- Cartesian product in the absence of a JOIN predicate.

```
select firstName, lastName, departmentName
from Student, Department;
```

The Student table has 13 entries, the department table has 5 entries, and the query result has 65 entries.

- Join Operation

```
select firstName, lastName, departmentName
from Student, Department
where Student.departmentId=Department.departmentId;
```

Note: Attributes names need to be qualified with the relation name if they are ambigous

Join Operation With Conditions

- A condition in the WHERE clause with a join condition further restricts the tuples selected.

```
select firstName, lastName, departmentName
from Student, Department
where Student.departmentId=Department.departmentId
        and Student .departmentId='COMP'
        and cga>2.5;
```

Note that attributes names need to be qualified with the relation name if they are ambiguous. For example, departmentId is an attribute of both the Student and the Department relations in the above example.

Natural Join Operation

- The Natural-Join operation merges the rows of two tables if the column(s) with identical name(s) match on their values.
- For the tables Student and Department, there is one such column departmentId.
- Only rows with identical values in the column departmentId will be merged, so students with departmentId ='COMP' will merge with the department with departmentId = 'COMP'.

```
select firstName, lastName, departmentName
from Student natural join department;
```

Summary

- We covered the following topics in this lab:
  - The **SELECT** statement.
  - Arithmetic operations in the **SELECT** statement.
  - Alias and concatenation of results.
  - The **WHERE** clause, the comparison operators and the logical operators.
  - The **ORDER BY** clause.
  - The Join operation.

Lab Exercise

- You must complete the lab exercise and upload the result to Canvas by **11:59 p.m. today**.

Ask for help if you need it!

IMPORTANT NOTES

Save your modified InsertMyself.sql script file either to the M drive or to a USB drive as any personal files on the lab computers will be automatically deleted periodically.

To access the database server from outside campus you need to use the HKUST VPN. See http://itsc.ust.hk/apps/vpn/ for instructions on how to connect to the HKUST VPN.