COMP 3311 Database Management Systems

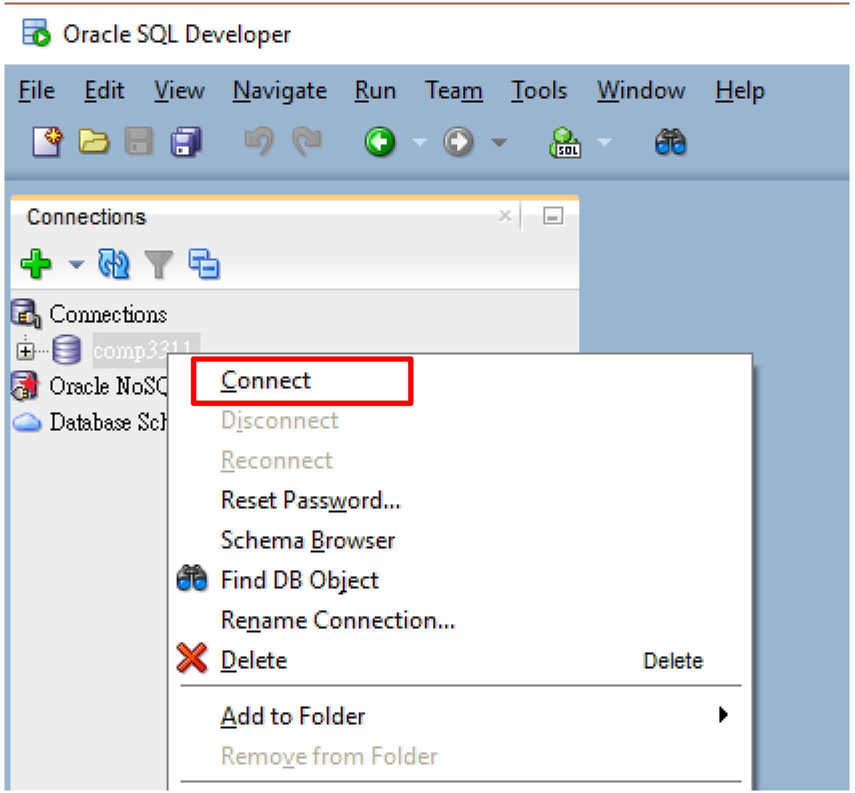
Lab 2. Basic SQL statements

Objectives of the Lab

- After this lab you should be able to
  - Know how to issue simple SQL commands to the SQL Developer
  - Know how to use the SELECT-FROMWHERE SQL clause.
  - Know how to use the ORDER BY options in SQL clauses.
  - Know how to use simple Join clauses

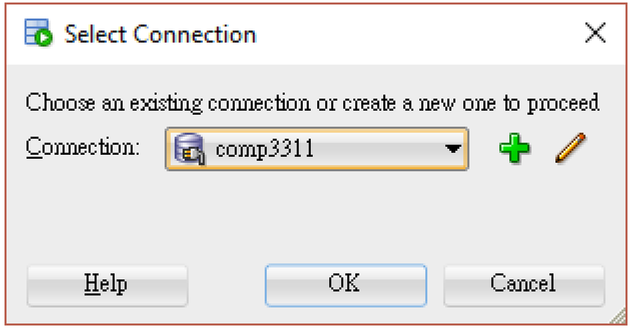
Re-connect to database server

- Connect to UST network if necessary
- Right click on the connection
- Click Connect
- Enter your Oracle username and password



Running a SQL script file 1

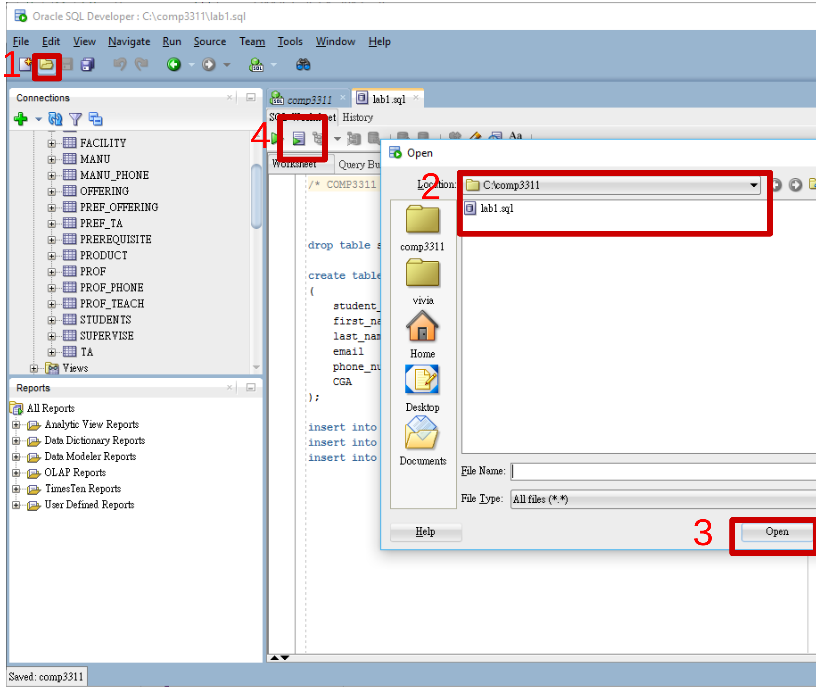
- Download (save) the lab2.sql file to local file system
  - http://course.cs.ust.hk/comp3311/labs/lab2.sql
- Open file
- Run script
- Choose connection


Select Connection

Choose an existing connection or create a new one to proceed

Connection: comp3311

Help | OK | Cancel



```
/* COMP3311

drop table s

create table
(
    student_
    first_na
    last_nam
    email
    phone_nu
    CGA
);

insert into
insert into
insert into
```

Running a SQL script file 2

- The table students created last time was dropped, a new students and a departments table is created.
- Basically lab2.sql creates a table called ‘students’ with 7 attributes and a table called department with 3 attributes.
- And lab2.sql inserts 5 different instances of students into the table and 3 different instances of departments into the table.
- Don’t worry if you do not understand the SQL statements for the time being. We shall cover them in details in the future labs.

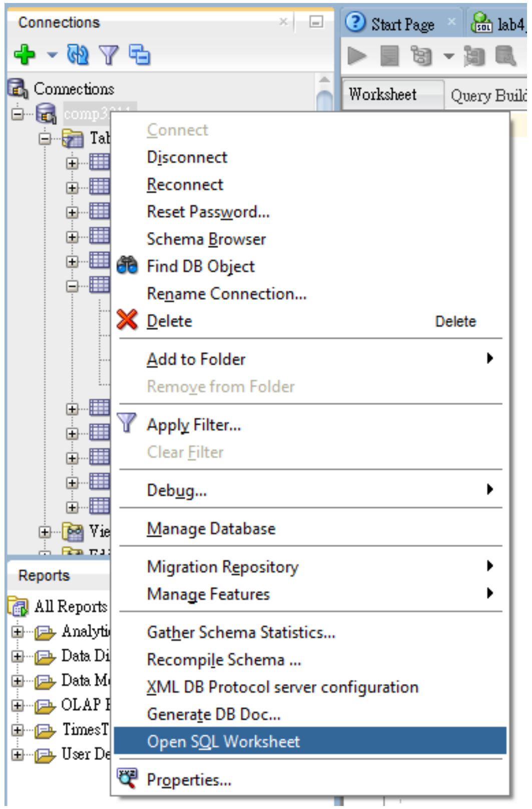
Running a SQL script file 3

You should get below result in Script Output after successfully run lab2.sql

```
Table STUDENTS dropped.

Table DEPARTMENTS dropped.

Table STUDENTS created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

Table DEPARTMENTS created.

1 row inserted.

1 row inserted.

1 row inserted.
```

Run SQL statement

- switch to worksheet or open new worksheet
- type statements in worksheet
- keep the cursor on this line and run statement
- result will be shown in Query Result or Script Output



Retrieving records using the SELECT statement

- Syntax:

  SELECT * | { [DISTINCT] *column* | *expression* [alias],..} FROM *table*

- Example, select all the columns from a table:

  ```
  SELECT * FROM departments;
  ```

- Example, select some specified columns

  ```
  SELECT department_id, name FROM departments;
  ```

Incorporating arithmetic operations to the SELECT statement

- It is possible to include arithmetic operations like * , / , + , - to the SELECT statement.
- For example:

```
SELECT last_name, CGA, CGA+2.0 FROM students;
SELECT last_name, CGA, CGA/2.0 FROM students;
```

- Note that 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 Alias

- We can change the column name of a table in the returned results by using the AS operator.

```
SELECT last_name AS ln FROM students;
```

- Use the SELECT statement to output a column named “ Quarter CGA ” which displays the result CGA/4.

```
SELECT CGA/4 AS "Quarter CGA" FROM Students;
```

Removing duplicates

- The default setting for the SELECT statement is to return all the relevant records – including duplicated ones. For example, the following statement will return all the department_ids from the students table:

```
SELECT department_id FROM students;
```

- To remove duplications, we can add the “ DISTINCT ” switch to the SELECT statement:

```
SELECT DISTINCT department_id FROM students;
```

Concatenating results in the SELECT statements

- Concatenating two columns in a select statement by using the “ || ” operator.

```
SELECT first_name || last_name AS "Full Name" FROM students;
```

- Adding a string to the results.

```
SELECT last_name|| ' studies in ' || department_id AS "Description"
FROM students;
```

Example of concatenations

- By using concatenations, we can express the results from a query in a more easy-to-comprehend form.
- For example we can artificially make an output from the table students to be:
    - Rita Lai(3456789) from the COMP department obtains CGA 10.5. His/Her email is cs_lrx@stu.ust.hk .
- What is the corresponding SELECT statement?
    - SELECT first_name||' '|| last_name || '(' || student_id || ') ' || 'from the ' || department_id || ' department obtains CGA ' || CGA ||'.' ||' His/Her email is ' || email || '@stu.ust.hk .' AS lab2 FROM students;

Specifying the output by using the WHERE clause

- The WHERE clause does not exist by itself, it is almost always in connection with the SELECT statement.
- Syntax:
    - SELECT * | { [DISTINCT] *column* | *expression* [alias],..} FROM *table* WHERE *conditions*;
- For example, we can retrieve only the information from the COMP department.

  ```
  SELECT * FROM departments WHERE department_id = 'COMP';
  ```

  The string ‘COMP’ in the condition clause is case sensitive.

Using Comparison Operator with the WHERE clause

- =,>,>=,<,<=,<>
  - Examples:

```
SELECT * from students WHERE CGA<>10.5;
SELECT * from students WHERE department_id='COMP';
```

Logical conditions

- AND
    - WHERE cga>=10 AND department_id ='MATH'
- OR
    - WHERE cga>10 OR department_id='MATH'
- NOT
    - WHERE department_id NOT IN ('COMP' , 'ELEC')

More conditions

- BETWEEN
  - WHERE cga BETWEEN 10 AND 12

    (reversing the order of 10 and 12 will give you nothing)
- IN
  - WHERE department_id in ('ELEC' , 'MATH')
- LIKE
  - WHERE first_name LIKE '%i%'
  - WHERE first_name LIKE '_i%'

    %: can have zero or more characters

    _: exactly one character.
- IS NULL
  - WHERE last_name IS NULL

Changing precedence using Parentheses 1

- THE AND condition has higher precedence than the OR condition
- The following selects students from the COMP department plus the students from the MATH department with CGA>11:
  - SELECT * FROM students WHERE department_id= 'COMP' OR department_id= 'MATH' AND CGA>11;

Changing precedence using Parentheses 2

- What if we want to select students with CGA >11, from either the ‘COMP’ or the ‘MATH’ departments? (Add a pair of parentheses)
  - SELECT * FROM students WHERE (department_id= 'COMP' OR department_id= 'MATH') AND CGA>11;

The ORDER BY clause

- Sort the result by one or more columns
  - ASC : ascending order (default)
  - DESC: descending order
- Examples:

```
SELECT * FROM students ORDER BY cga;
SELECT * FROM students ORDER BY cga DESC;
```

More about the ORDER BY clause

- Sort by an alias

  SELECT first_name, CGA*0.8 AS wCGA FROM students ORDER BY wCGA;

- Sort by multiple columns

  SELECT * FROM students ORDER BY department_id ASC, first_name DESC;

SQL JOINS

- CROSS product in the absence of JOIN predicate:
  SELECT first_name, last_name FROM students, departments;
  The students table has 5 entries, the departments table has 3 entries, and we have 15 entries for this query.

- JOIN
  - SELECT first_name, last_name from students, departments where students.department_id=departments.department_id;

SQL JOINS : Natural Join 1

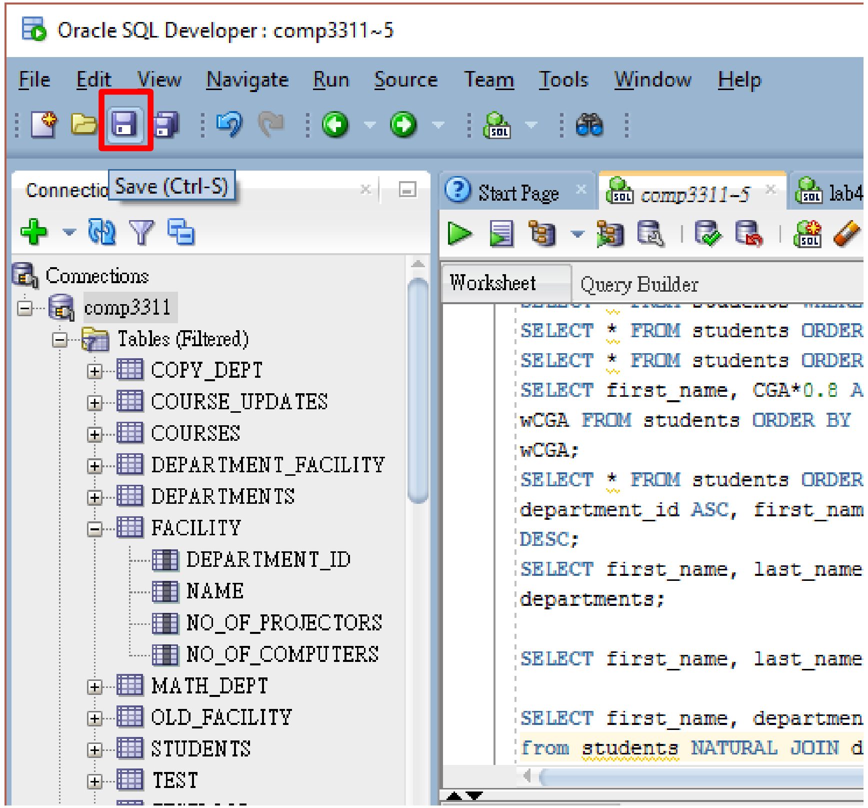
- Joins two tables.
- The Natural-Join operation matches the rows of the two tables by looking at the column(s) with identical name(s).
- The rows from the two tables are merged if the column entry/entries match(es).

SQL JOINS : Natural Join 2

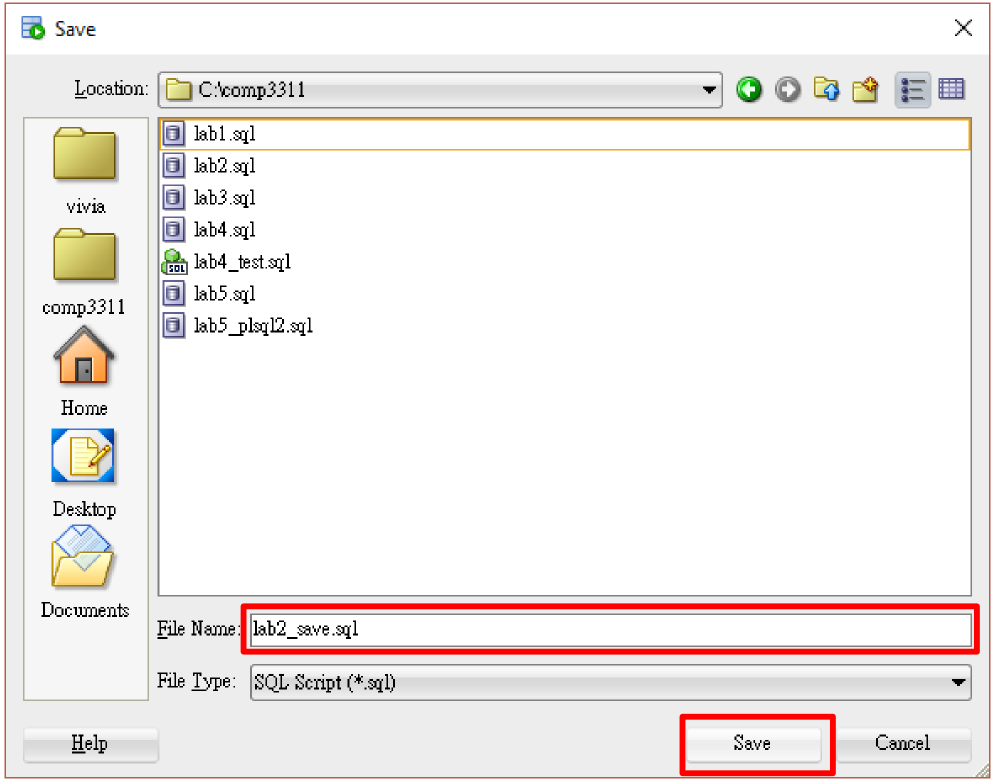
- For the tables students and departments, there is one such column department_id
- Only rows with identical entries in the column department_id will be merged, so students with department_id=‘COMP’ will merge with the department with department_id=‘COMP’.

  SELECT first_name, department_id, name from students NATURAL JOIN departments;

Saving SQL worksheet



```
SELECT * FROM students ORDER
SELECT * FROM students ORDER
SELECT first_name, CGA*0.8 A
wCGA FROM students ORDER BY
wCGA;
SELECT * FROM students ORDER
department_id ASC, first_nam
DESC;
SELECT first_name, last_name
departments;

SELECT first_name, last_name

SELECT first_name, departmen
from students NATURAL JOIN d
```



Conclusions

- 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 JOINs.

Exercise

- Create queries according to the following requirements:

    1. Display the first_name and email of the students from the COMP department.
    2. Display the first_name of all the students whose first_name contains ‘ r ’ as the 4th character.
    3. Display the first_name of all the students whose first_name contains either an ‘ a’ or an ‘e’.
    4. Display the information for the students who are from (the COMP or the ELEC department) and the CGA is not 8.34 or 12.

Suggested Solutions

1. SELECT first_name, email FROM students WHERE department_id='COMP';
2. SELECT last_name FROM students WHERE first_name LIKE '_ _ _ r%';
3. SELECT last_name FROM students WHERE first_name LIKE '%a%' or first_name LIKE '%e%';
4. SELECT * FROM students where department_id in ('COMP','ELEC') AND cga NOT in (8.34,12)