COMP 3311 Database Management Systems

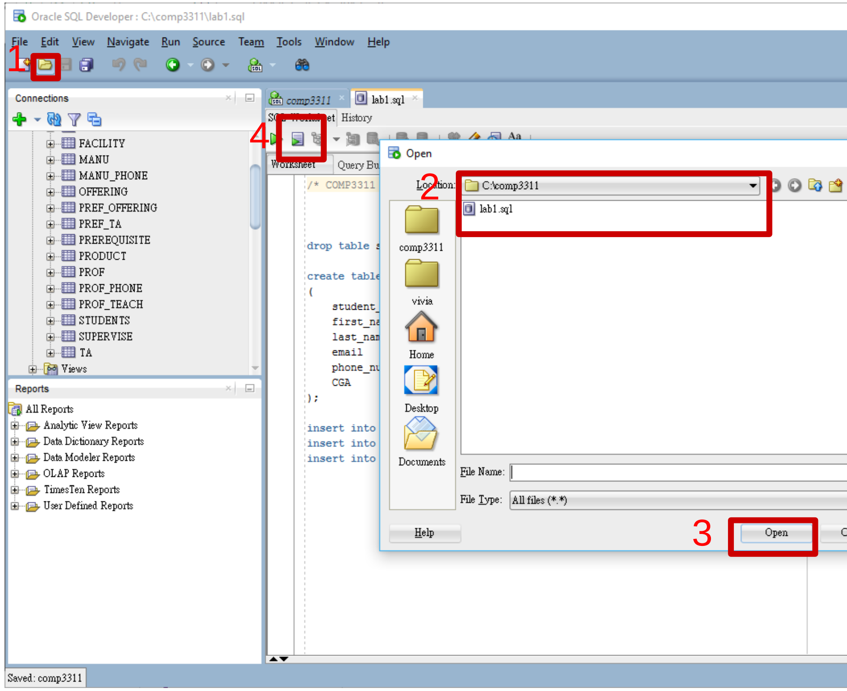
Lab 3. SQL aggregate functions and sub-queries

Objectives of the Lab

- After this lab you should be able to
  - use SQL string functions
  - use SQL number functions
  - use SQL date functions
  - use aggregate functions in SQL
  - use SQL sub-queries

Downloading and running the lab SQL script file 1

- Login Oracle database server using SQL Developer with your Oracle account
- Download (save) the lab3.sql file to local file system
  - http://course.cs.ust.hk/comp3311/labs/lab3.sql
- Open file
- Run script
- Choose connection



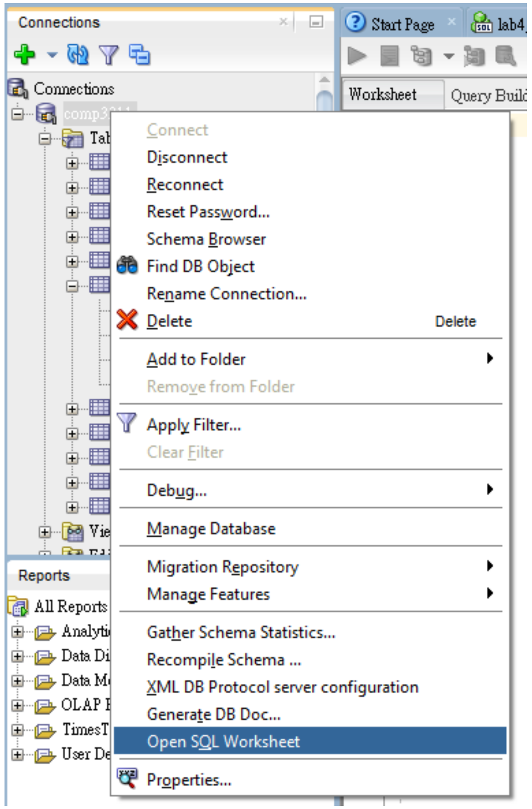
```
/* COMP3311

drop table s

create table
(
    student_
    first_na
    last_nam
    email
    phone_nu
    CGA
);

insert into
insert into
insert into
```

Downloading and running the lab SQL script file 2

- The tables students and departments created last time were dropped.
- 5 new tables are created. They are
  - students,
  - courses,
  - enroll,
  - departments, and
  - facility.
- The facility table indicates the number of projectors and computers each department has.

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

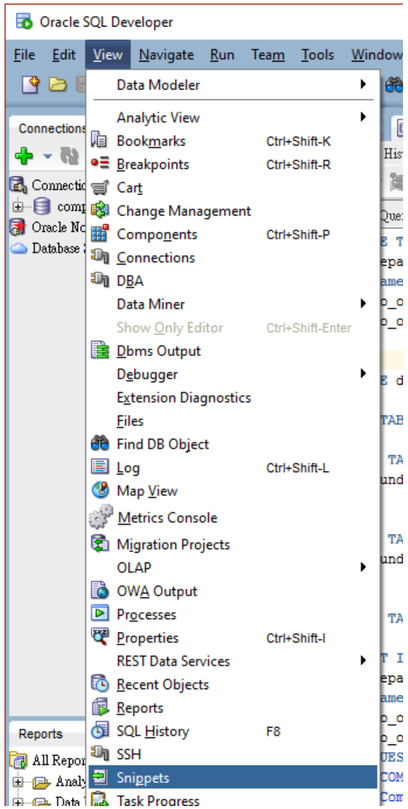


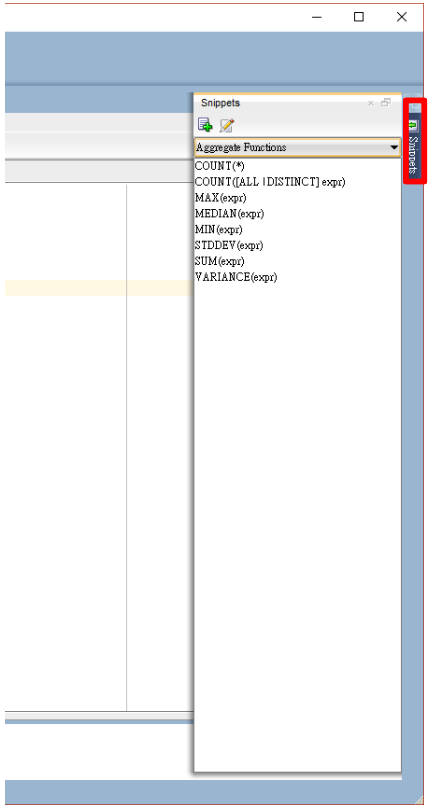
Snippets 1

- Snippets are code fragments, such as SQL functions, Optimizer hints, and miscellaneous PL/SQL programming techniques.
- Some snippets are just syntax, and others are examples.
- You can insert and edit snippets when you are using the SQL Worksheet or creating or editing a PL/SQL function or procedure.

Snippets 2

- Click the View menu, select Snippets.
- In the snippets window (on the right side), use the drop-down to select a group.
- You can display and hidden the snippets window with Snippets button is placed in the right window margin





Snippets 3

- To insert a snippet into your code
  - Drag the snippet from the snippets window and
  - Drop it into the desired place in your code; then
  - Edit the syntax so that the SQL function is valid in the current context.
  - To see a brief description of a SQL function in a tooltip, hold the pointer over the function name.

Snippets 4

- For example, you could
  - type SELECT
  - use the drop-down to select Character Functions
  - drag LOWER(char)
  - edit the LOWER function syntax : char -> last_name
  - type the rest of the statement, such as in the following:
    SELECT LOWER(last_name) FROM students;

SQL String functions 1

- String functions take strings as input and give either strings or numerical values as output
- Examples:
  - LOWER(string) – converts all the characters in the string to lowercase

    ```
    SELECT LOWER(last_name) FROM students;
    ```
  - UPPER(string) -converts all the characters in the string to uppercase

    ```
    SELECT UPPER(last_name) FROM students;
    ```

SQL String functions 2

- INITCAP(string) – set the first character of each word to be in uppercase.

```
SELECT INITCAP(name) FROM courses;
```

- SUBSTR(string,position,length) – returns a particular portion of a string.

```
SELECT SUBSTR(first_name,2,3) FROM students;
```

- CONCAT(string1,string2) – concatenates two strings. “||” on the other hand can concatenate more than 2 strings.

```
SELECT CONCAT(last_name, first_name) FROM students;
```

SQL String functions 3

- INSTR(string1, length, string2) – returns the location of string2 in string1.

  SELECT INSTR(last_name,'or') FROM students;
- LENGTH(string) – returns the length of a string.

  SELECT LENGTH(last_name) FROM students;
- LPAD(string1,length,string2) – pads string2 to the left of string1 so that the new string's length equals to length.

  SELECT LPAD('a',10,'b') FROM students;
- RPAD(string1,length,string2) – pads string2 to the right of string1 so that the new string's length equal to length.

  SELECT RPAD('a',10,'b') FROM students;

SQL String functions 4

- LTRIM(string) – removes all the left spaces from the string.
  LTRIM('   a ') gives you 'a '
- RTRIM -– removes all the right spaces from the string.
  RTRIM('   a ') gives you '   a'

SQL numeric functions

- These functions accept numeric inputs and output numeric values.
- Examples
  - MOD(number1,number2) – returns number1 mod number2.
  - POWER(number1,number2) – returns (number1)^{number2}.
  - ROUND(number1, Integer_number2) - returns a value rounded to integer_number2 places.
  - TRUNC(number1, Integer_number2) – truncates number1 to integer_number2 decimal places.

SQL numeric functions

- Examples
  - SELECT MOD(7,2) FROM dual;
  - SELECT POWER(4,3) FROM dual;
  - SELECT ROUND(5.55,1) FROM dual;
  - SELECT TRUNC(5.55,1) FROM dual;

SQL DATE functions 1

- The default date format for Oracle is 'DD-MON-YY'. 7th of March 2014 is therefore ' 07-MAR-14'.
- Examples
  - ADD_MONTHS(date, number) – adds “number” of months to the date.

```
SELECT ADD_MONTHS('07-MAR-14',2) FROM dual;
ADD_MONTH
---------
07-MAY-14
```

(dual is an Oracle built-in table for SQL queries that do not logically have table names)

SQL DATE functions 2

- LAST_DAY(date) – returns the date of the last day in the month of the specified date
- NEXT_DAY(*date*,*weekday*) – returns the date of first *weekday* that is later than *date*.

The possible values for weekday are

'SUNDAY', 'MONDAY', 'TUESDAY', 'WEDNESDAY', 'THURSDAY', 'FRIDAY', 'SATURDAY'

NEXT_DAY('05-MAR-14','SATURDAY') would return '08-MAR-14'

SQL DATE functions 3

- TO_DATE(string, date_format_string)- convert the string to the corresponding date according to the date_format_string.

```
TO_DATE('11-MAR-14','DD-MON-YY')
TO_DATE('2014-03-23','YYYY-MM-DD')
```

- TO_CHAR(date,format_mask) – convert a date to a string with respect to the format mask. The format masks can be
  - 'YYYY' : 4-digit year
  - 'MM'   : 2-digit month
  - 'MONTH' : 'JANUARY', 'FEBRUARY', etc

SQL aggregate functions 1

- An aggregation function performs calculation on a collection of input data and returns a single value for the data.
- ALL aggregate functions (except for count(*)) ignore NULL values (i.e. do not include them in the calculation)
- Examples:
  - AVG (column) – returns the average value for the values in the column.

    SELECT AVG(CGA) FROM students;
  - COUNT(column) – returns the number of records.

    SELECT COUNT(CGA) FROM students;

SQL aggregate functions 2

- Examples:
  - MAX (column) – returns the maximum value for the values in the column.

    SELECT MAX(CGA) FROM students;
  - MIN(column) – returns the minimum value for the values in the column

    SELECT MIN(CGA) FROM students;
  - STDDEV(column) – returns the sample standard deviation for the values in the column

    SELECT STDDEV(CGA) FROM students;
  - SUM (column) – adds up all the values in the column

    SELECT SUM(CGA) FROM students;

The GROUP BY clause

- The GROUP BY clause groups the data by one or more columns, so that aggregate functions can be applied.

  SELECT department_id, last_name AS LN FROM students GROUP BY department_id, last_name;

  Note: 1. The non-aggregation attributes in the SELECT clause must be a subset of the attributes in the GROUP BY clause.

  2. you can not use column alias in the GROUP BY clause Under Oracle, i.e. you can't change last_name to LN.

GROUP BY and the HAVING clause 1

- The HAVING clause is applied to the GROUP BY clause to specify the condition(s) under which the results should be returned.

  ```
  SELECT department_id, MAX(CGA) FROM
  STUDENTS GROUP BY department_id
  HAVING MAX(CGA)=12 or MAX(CGA)<10;
  ```

GROUP BY and the HAVING clause 2

- The SQL statement can also contain the WHERE clause to specify the condition(s) for the records to be chosen, the chosen records will then be filtered by the condition(s) specified by the HAVING clause.

```
SELECT department_id, MAX(CGA) FROM
STUDENTS WHERE department_id='COMP'
GROUP BY department_id
HAVING MAX(CGA)=12 or MAX(CGA)<10;
```

Sub-query (1/3)

students(student_id, first_name, last_name, CGA, department_id)

- Sub-query in the WHERE clause works as part of the row selection process
- Use a sub-query in the WHERE or HAVING clause when the criteria depends on the results of another table

```
SELECT first_name, department_id, CGA
FROM students
WHERE CGA = (
        SELECT MIN(CGA)
        FROM students
);
```

Sub-query

- Do you know what this query does?

Sub-query (2/3)

students(student_id, first_name, last_name, CGA, department_id)

- Do you know what the following sub-query does?

```
SELECT department_id, AVG(CGA)
FROM students
GROUP BY department_id
HAVING AVG(CGA) >
(
    SELECT AVG(CGA)
    FROM students
);
```

Sub-query

Sub-query (3/3)

students(student_id, first_name, last_name, CGA, department_id)

- The following query utilizes two temporary tables to store the result of the sub-queries.

```
SELECT temp_table.department_id, temp_table.AVG_CGA
FROM (SELECT department_id,avg(CGA) AS AVG_CGA
            FROM students
            GROUP BY department_ID
              ) temp_table
WHERE temp_table.AVG_CGA >
(  SELECT temp_cga.overall_avg_cga
   FROM ( SELECT AVG(CGA) as overall_avg_cga
   FROM students ) temp_cga );
```

Example of Division Query

- Find the first names of students who take all the courses:

```
select s.first_name
from students s
where not exists(
     (select course_id from courses)
      minus
      (select course_id
       from enroll e
       where e.student_id=s.student_id));
```

Conclusion

- We covered the following topics in this lab:
  - Using Snippets in SQL Developer
  - SQL string functions, number functions and date functions
  - Aggregation functions in SQL
  - Sub-queries
  - A simple example on division