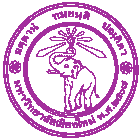
****

**CHIANG MAI UNIVERSITY**

**Bachelor of Science (DIGITAL INDUSTRY INTEGRATION)**

**College of Arts, Media and Technology**

**1st Semester / Academic Year 2021**

**Database System and Database System Design**

Name ปกรณ์ ศิลปประภา Student ID 632110345 Section [DII Bootcamp] Database 1/64

1. Pre-defined Functions

SQL has provided a group of pre-defined function. The syntax to use the pre-defined function is given as follows.

SELECT *FUNCTION(column\_name), …*  
FROM *table\_name*

WHERE *conditions*;

The list of function and the description is given as follows.

|  |  |
| --- | --- |
| **Function** | **Description** |
| AVG | Calculate the average value of the inputted column. |
| COUNT | Determine the number of row in a table. |
| MAX | Determine the maximum value of the inputted column. |
| MIN | Determine the minimum value of the inputted column. |
| SUM | Calculate the summation value of the inputted column. |

1.1 Run the following SQL. This SELECT statement will count the total number of row in the table.

SELECT *COUNT(\*)*  
FROM *student*;

1.2 Run the following SQL. This SELECT statement will determine the average value of height in the student table.

SELECT *AVG(height)*  
FROM *student*;

1.3. Write down your own SQL statement to count the number of student whose age is more than 20 years old from the student table. Run the SQL.

|  |
| --- |
| SELECT COUNT(age) FROM student WHERE age > 20; |

1.4 Write down your own SQL statement to determine the highest height of the student in the student table. Run the SQL.

|  |
| --- |
| SELECT MAX(height) FROM student; |

1.5 Write down your own SQL statement to determine the shortest height of the student in the student table. Run the SQL.

|  |
| --- |
| SELECT MIN(height) FROM student; |

1.6 Write down your own SQL statement to determine the summation of age the student table. Run the SQL.

|  |
| --- |
| SELECT SUM(age) FROM student; |

1.7 Write down your own SQL statement to determine the average height of student whose age is more than 20 years old from the student table. Run the SQL.

|  |
| --- |
| SELECT AVG(height) FROM student WHERE age > 20; |

1.8 Write down your own SQL statement to determine the average height, maximum age and minimum student ID from the student table. Run the SQL.

|  |
| --- |
| SELECT AVG(height),MIN(age),MIN(Id) FROM student; |

1.9 Write down your own SQL statement to retrieve the student id, first name and last name of the students who have the highest height.

SELECT Id,firstName,lastName,height

FROM student

WHERE height = (SELECT MAX(height) FROM student);

1.10 Write down your own SQL statement to retrieve the student id, first name and last name of the students who have the age higher than the average.

SELECT Id,firstName,lastName,age

FROM student

WHERE age > (SELECT AVG(age) FROM student);

2. DISTINCT operator

In previous lab, one task asks you to determine the number of student per advisor. If you have done this problem, you might find it a challenge. The difficulty of this problem is to remove the redundancy of the advisor. The SELECT statement treats each data in different tuple as a distinct data even it has the same value. However, the SQL also provide the mechanism to convert the repeated value into a single one by using DISTINCT operator.

SELECT DISTICT(*column\_name), …*  
FROM *table\_name*

WHERE *column\_name LIKE search\_text*;

2.1 Try the SELECT statement to retrieve all of the data from the student table. You can see that the MySQL will return all of the data in the student table.

SELECT *advisor*  
FROM *student;*

2.2 Run the following SQL. This SELECT statement will return only the unique value in the advisor column.

SELECT *DISTINCT(advisor)*  
FROM *student;*

2.3 Write down your own SQL statement to count the distinct value of age from the student table. Run the SQL.

|  |
| --- |
| SELECT COUNT(DISTINCT(age)) FROM student; |

2.4 Write down your own SQL statement to retrieve the distinct value of height from the student table. Run the SQL.

|  |
| --- |
| SELECT DISTINCT(height) FROM student; |

2.5 Write down your own SQL statement to count the distinct value of advisor from the student table where the advisee has height higher than the average. Run the SQL.

|  |
| --- |
| SELECT DISTINCT(advisor) FROM student WHERE height > (SELECT AVG(height) FROM student); |

3. Sorting

Since the SQL is a language to manipulate a large quantity of data, it is equipped with the sorting command to help with the management. The syntax of sort the data in the database is as follows.

SELECT  *column\_name\_1, column\_name\_2, …, column\_name\_n*  
FROM *table\_name*

ORDER BY *column\_name\_1 STYLE, column\_name\_2 STYLE,…*

When there is more than one column given in the ORDER BY clause, the SQL will sort the column in order. For example, ORDER BY *col1, col2.* The SQL will sort the col1 and then sort the col2.

There are two sorting method.

* Descending order. This style will sort the data in the given column from higher value to lower value. The command to indicate that the data is sort by the given column is DESC.
* Ascending order. This style will sort the data in the given column from smaller value to higher value. The command to indicate that the data is sort by the given column is ASC. This sorting style is a default for sorting.

3.1 Run the following SQL. This SELECT statement will return all of the data in sorted manner.

SELECT *\**  
FROM *student*

*ORDER BY studentid;*

3.2 Run the following SQL. This SELECT statement will return all of the data in sorted manner.

SELECT *\**  
FROM *student*

*ORDER BY studentid ASC;*

3.3 Run the following SQL. This SELECT statement will return all of the data in sorted manner.

SELECT *\**  
FROM *student*

*ORDER BY studentid DESC;*

3.4 Write down your own SQL statement to retrieve the all of the data from the student table where the data is sorted by height in descending order. Run the SQL.

|  |
| --- |
| SELECT \* FROM student ORDER BY height DESC; |

3.5 Write down your own SQL statement to retrieve the all of the data from the student table where the data is sorted by age in ascending order. Run the SQL.

|  |
| --- |
| SELECT \* FROM student ORDER BY age ASC; |

3.6 Write down your own SQL statement to retrieve the all of the data from the student table where the student is higher than 170 and the data is sorted by height in ascending order. Run the SQL.

|  |
| --- |
| SELECT \* FROM student WHERE height > 170 ORDER BY height ASC; |

3.7 Write down your own SQL statement to retrieve the all of the data from the student table where the data is sorted by age in ascending order and height in descending order, respectively. Run the SQL.

|  |
| --- |
| SELECT \* FROM student ORDER BY height,age ASC; |

4. Grouping

Some attribute of the data may limit to a certain set of values. To investigate into the data using the set as criteria, we can use the GROUP BY command to group the data in the table. The syntax of the grouping command is given as follows.

SELECT *column\_name\_1, column\_name\_2, …, column\_name\_n, …*  
FROM *table\_name*

GROUP BY *column\_name*;

4.1 Try the SELECT statement to retrieve the grouped value in the advisor column from the student table. Run the SQL.

SELECT *advisor*  
FROM *student*

GROUP BY advisor;

4.2 Try the SELECT statement to retrieve the grouped value in the advisor column and the number of row in each group from the student table. Run the SQL.

SELECT advisor*,count(\*)*  
FROM *student*

GROUP BY advisor;

4.3 Try the SELECT statement to retrieve the grouped value in the advisor column and the number of row in each group from the student table. Run the SQL.

SELECT advisor,height,count(\*)

FROM student

GROUP BY advisor,height;

4.4 Write down your own SQL statement to retrieve the age and count the number of student from the student table where the data is grouped by age and count only the student that is higher than 170. Run the SQL.

SELECT age,height,COUNT(\*) FROM student WHERE height > 170 GROUP BY age,height;

4.5 Write down your own SQL statement to retrieve the advisor, the number of advisees, and the average height of the student from the student table. Run the SQL.

SELECT advisor,AVG(height),COUNT(\*)  FROM student GROUP BY advisor,height;

5. Grouping with conditions

Similar to the WHERE clause, the GROUP BY is incorporated with the condition to refine the grouping. The command is the HAVING. The syntax of the HAVING command is given as follows.

SELECT *column\_name\_1, column\_name\_2, …, column\_name\_n*  
FROM *table\_name*

GROUP BY *column\_name\_1, column\_name\_2, …, column\_name\_n*;

HAVING conditions;

5.1 Try the SELECT statement to group the student using age and count the number of student in each group from the student table. Run the SQL.

SELECT age,count(\*)

FROM student

GROUP BY age;

5.2 Try the SELECT statement to group the student using age and count the number of student in each group from the student table. Moreover, the SQL will display only the group that has number of student more than 1. Run the SQL.

SELECT age,count(\*)

FROM student

GROUP BY age

HAVING count(\*)>1;

5.3 Try the SELECT statement to group the student using age and count the number of student in each group from the student table. Moreover, the SQL will display only the group that has number of student more than 1 and the height of the student that we count must be higher than 170. Run the SQL.

SELECT age,count(\*)

FROM student

WHERE height > 170

GROUP BY age

HAVING count(\*)>1;

5.4 Write down your own SQL statement to retrieve the advisor and the average height of student of each advisor from the student table. Run the SQL.

SELECT advisor,AVG(height) FROM student GROUP BY advisor

5.5 Write down your own SQL statement to retrieve the average age and count the number of student from the student table where the data is grouped by advisor. The student in the counting must have “a” in the name and we will show only the group that has the number of student more than 10. Run the SQL.

SELECT advisor,AVG(age),COUNT(\*)

FROM student

WHERE firstName LIKE '%a%'

GROUP BY advisor,age HAVING COUNT(\*)>10;

1. Database creation

The SQL statement to create the database is given as follows.

CREATE DATABASE database\_name;

* 1. Run the following SQL. This statement will create a database named [yournickname][last 4 digit of your student id].

CREATE DATABASE yam1020;

1. Table creation

The SQL statement to create a table in a database is given as follows.

CREATE TABLE table\_name

(

column\_name1 data\_type(size),

column\_name2 data\_type(size),

column\_name3 data\_type(size),

....,

PRIMARY KEY (column\_name)

);

The PRIMARY KEY constraint is used to indicate the primary key of the table.

The common datatype in SQL SERVER can be looked on https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver15 .

* 1. Run the following SQL in the database from 6.1.

CREATE TABLE myFirstDB

(

id int,

firstname varchar(30),

lastname varchar(30),

age int,

PRIMARY KEY(temp\_id)

);

* 1. Write down your own SQL statement to create a table named student2 in the **database from 6.1**. The statement must contain the following attribute id as varchar of size 9, firstname as varchar of size 30, lastname as varchar of size 30, age as integer, faculty as varchar 30, height as float and date\_of\_birth as datetime. The primary key of this table is id. Run the SQL.

CREATE TABLE Student2

(

id VARCHAR(6),

firstname varchar(30),

lastname varchar(30),

age int,

faculty VARCHAR(30),

height FLOAT,

date\_of\_birth DATETIME,

PRIMARY KEY(id)

);

* 1. Insert the data into the newly created table using the following data and given instruction.
* Use you last 4 digits of your student ID, mod by 26 and add 1 to the result.
* Insert the records from the given data 5 records whose id is earlier than your result.
* Insert the records from the given data 5 records whose id is later than your result.
* If the insert records are out-of-bound, students need to use the records at the beginning or the ending.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| id | firstname | lastname | age | faculty | height | date\_of\_birth |
| 001 | Dylan | Sharp | 29 | Software Engineering | 167 | 9-24-2002 |
| 002 | Rory | Willis | 20 | Software Engineering | 180 | 12-23-2002 |
| 003 | Archie | Hussain | 19 | Digital Game | 183 | 10-31-2001 |
| 004 | Noah | Ross | 23 | Digital Game | 179 | 3-13-2001 |
| 005 | Ryan | Moore | 23 | Software Engineering | 181 | 6-5-2001 |
| 006 | Silas | Mercado | 22 | Digital Game | 184 | 1-30-2001 |
| 007 | Kai | Sanders | 28 | Modern Management and Information Technology | 172 | 6-13-2001 |
| 008 | Finley | Mueller | 27 | Animation and Visual Effect | 179 | 8-18-2001 |
| 009 | Triston | Whitfield | 22 | Animation and Visual Effect | 168 | 1-23-2002 |
| 010 | Rashad | Rowland | 21 | Digital Game | 182 | 5-18-2001 |
| 011 | Kian | Wells | 27 | Animation and Visual Effect | 180 | 9-21-2002 |
| 012 | Bradley | Lloyd | 23 | Animation and Visual Effect | 184 | 5-4-2002 |
| 013 | Owen | Moore | 25 | Animation and Visual Effect | 183 | 5-11-2002 |
| 014 | Zachary | Cooke | 25 | Software Engineering | 182 | 2-6-2002 |
| 015 | Arthur | Wilkinson | 21 | Modern Management and Information Technology | 171 | 5-17-2001 |
| 016 | Donte | Hurley | 21 | Software Engineering | 169 | 7-9-2002 |
| 017 | Bryce | Woodward | 19 | Digital Industry Integration | 178 | 3-30-2002 |
| 018 | Isaias | Norris | 23 | Animation and Visual Effect | 183 | 1-18-2002 |
| 019 | Bryan | Maddox | 24 | Digital Industry Integration | 169 | 9-26-2002 |
| 020 | Eddie | Mcmillan | 25 | Software Engineering | 173 | 11-26-2002 |
| 021 | Reggie | Ellis | 28 | Modern Management and Information Technology | 181 | 4-10-2001 |
| 022 | Steff | Wilson | 22 | Modern Management and Information Technology | 178 | 11-13-2002 |
| 023 | Jessie | Thomas | 24 | Animation and Visual Effect | 176 | 1-23-2002 |
| 024 | Jaime | King | 29 | Modern Management and Information Technology | 172 | 5-21-2001 |
| 025 | Val | Russell | 24 | Modern Management and Information Technology | 170 | 9-18-2001 |
| 026 | Rory | Hendricks | 29 | Modern Management and Information Technology | 184 | 6-10-2001 |
| 027 | Danni | Bailey | 26 | Animation and Visual Effect | 182 | 10-7-2001 |

1. Update a data tuple in a table

The SQL statement to change values of tuples a table in a database is given as follows.

UPDATE table\_name

SET column\_name\_1=value1, column\_name\_2=value2,…

WHERE conditions;

The conditions are used to locate the tuple to be updated.

|  |
| --- |
| Use the database DBdii, not database from 7.3 |

* 1. Before you execute the UPDATE statement, run the following SELECT statement. Run the following SQL and check the value in the table.

SELECT \*

FROM student2

WHERE age = 20;

Run the following UPDATE statement

UPDATE student2

SET firstName = 'hello'

WHERE age=20;

Then, run the following SELECT statement again.

SELECT \*

FROM student2

WHERE age = 20;

Study the result and try to understand the operation.

* 1. Write down your own SQL statement to change the first name of the student whose age is the highest in table student2 to “noobie”. Run the SQL.

UPDATE student2

SET firstName = 'noobie'

WHERE age=(SELECT MAX(height) FROM student2 );

* 1. Write down your own SQL statement to change the first name and last name of the student whose height is greater than 170.0 and age is more than 20 years old to “hello” and world in student table. Run the SQL.

UPDATE student2

SET firstName = 'hello',lastname = 'hello'

WHERE height > 170.0 AND age > 20;

* 1. Given student2, write down your own SQL statement to change the height of the student whose first name contains ‘a’ into 177.77. Run the SQL.

UPDATE student2

SET height = 177.77

WHERE firstname LIKE '%a%';

* 1. Given student2, write down your own SQL statement to change the date of birth of the student whose id contains 211 into today. Run the SQL.

UPDATE student2

SET date\_of\_birth = 2564-08-11

WHERE id = '211';

1. Delete a data tuple in a table

The SQL statement to delete a data tuple in a table is given as follows.

DELETE FROM table\_name

WHERE conditions;

Remarks: Please reinsert the student table again.

* 1. Before you execute the UPDATE statement, run the following SELECT statement. Run the following SQL and check the value in the table.

SELECT \*

FROM student2

WHERE age = 20; [If there is no record with age of 20m you may select other age]

Run the following UPDATE statement

DELETE FROM student2

WHERE age=20;

Then, run the following SELECT statement again

SELECT \*

FROM student2

WHERE age = 20;

* 1. Write down your own SQL statement to delete a student whose height is higher than 177 in the student table. Run the SQL.

DELETE FROM student2 WHERE height>177;

* 1. Write down your own SQL statement to delete a student whose id is “582115042” in the student table. Run the SQL.

DELETE FROM student2 WHERE id = '582115042';

1. Table delete

The SQL statement to delete a table in a database is given as follows.

DROP TABLE table\_name;

* 1. Run the following SQL. This statement will delete a database from 6.1.

DROP TABLE student2;

Study the result and try to understand the operation.

1. Database delete

The SQL statement to delete the database is given as follows.

DROP DATABASE database\_name;

* 1. Run the following SQL. This statement will delete a database.

DROP DATABASE yam1020;