關聯式資料庫管理系統
結構化查詢語言(SQL)
使用MySQL

資料查詢 – SELECT 命令

資料彙總與資料分組

(Aggregating and Grouping Data)

資料查詢 – SELECT 敘述

子句(Element)	Expression	Role
SELECT	<select list>	給定查詢的資料項目 Defines which columns to return
FROM	<table source>	給定資料來源 Defines table(s) to query
WHERE	<search condition>	給定查詢/過濾資料條件 Filters rows using a predicate
GROUP BY	<group by list>	資料分組設定 Arranges rows by groups
HAVING	<search condition>	給定分組資料查詢/過濾條件 Filters groups using a predicate
ORDER BY	<order by list>	給定查詢結果排序方式 Sorts the output

群組/彙總函數(Group/Aggregate Functions)

- 群組/彙總函數主要提供資料彙總功能,如:計算平均、資料加總…等
  - 多筆記錄(rows)執行一次，傳回一個結果

EMP

DEPTNO	SAL
10	2450
10	5000
10	1300
20	800
20	1100
20	3000
20	3000
20	2975
30	1600
30	2850
30	1250
30	950
30	1500
30	1250

"maximum salary in the EMP table"

MAX(SAL)
5000

群組/彙總函數(Group/Aggregate Functions)

- 群組函數

函數	說明
COUNT(*)	回傳資料的筆數
COUNT(column)	回傳欄位不為空值的筆數
COUNT(DISTINCT column)	回傳欄位去除重複列不為空值的筆數
MAX(column)	回傳欄位中的最大值
MIN(column)	回傳欄位中的最小值
SUM(column)	回傳欄位的加總
AVG(column)	回傳欄位的平均值

COUNT(*)

- 傳回資料表中的資料列數

```
mysql> SELECT *
    -> FROM emp
    -> WHERE deptno=10;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7839 | KING   | PRESIDENT | NULL | 1981-11-18 | 5000 | NULL |     10 |
|  7782 | CLARK  | MANAGER   | 7839 | 1981-06-09 | 2450 | NULL |     10 |
|  7934 | MILLER | CLERK     | 7782 | 1982-01-23 | 1300 | NULL |     10 |
+-------+--------+-----------+------+------------+------+------+--------+

mysql> SELECT count(*)
    -> FROM emp
    -> WHERE deptno=10;
+----------+
| count(*) |
+----------+
|        3 |
+----------+
```

COUNT(column|expr)

- 傳回欄位或運算式不為空值的資料列數

```
mysql> SELECT comm
    -> FROM emp
    -> WHERE deptno=30;
+------+
| comm |
+------+
| NULL |
| 1400 |
|  300 |
|    0 |
| NULL |
|  500 |
+------+
```

```
mysql> SELECT COUNT(comm)
    -> FROM emp
    -> WHERE deptno=30;
+-------------+
| COUNT(comm) |
+-------------+
|           4 |
+-------------+
```

Number of columns
(not include null value)

COUNT(DISTINCT column|expr)

- 傳回欄位或運算式中去除重複資料的資料列數,但不包含空值

```
mysql> SELECT DISTINCT comm
    -> FROM emp
    -> WHERE deptno=30;
+------+
| comm |
+------+
| NULL |
| 1400 |
|  300 |
|    0 |
|  500 |
+------+
```

```
mysql> SELECT COUNT(DISTINCT comm)
    -> FROM emp
    -> WHERE deptno=30;
+----------------------+
| COUNT(DISTINCT comm) |
+----------------------+
|                    4 |
+----------------------+
```

Number of distinct column value (not include null value)

MAX(column|expr) Function

- 傳回欄位或運算式中最大值

```
mysql> SELECT sal
    ->   FROM emp
    ->  WHERE deptno=30;
+------+
| sal  |
+------+
| 2850 |
| 1250 |
| 1600 |
| 1500 |
|  950 |
| 1250 |
+------+
```

```
mysql> SELECT MAX(sal)
    ->   FROM emp
    ->  WHERE deptno=30;
+----------+
| MAX(sal) |
+----------+
|     2850 |
+----------+
```

註：NULL Value皆忽略不計入

MIN(column|expr) Function

- 傳回欄位或運算式中最小值

```
mysql> SELECT sal
    -> FROM emp
    -> WHERE deptno=30;
+------+
| sal  |
+------+
| 2850 |
| 1250 |
| 1600 |
| 1500 |
|  950 |
| 1250 |
+------+
```

```
mysql> SELECT MIN(sal)
    -> FROM emp
    -> WHERE deptno=30;
+----------+
| MIN(sal) |
+----------+
|      950 |
+----------+
```

註：NULL Value皆忽略不計入

SUM(column|expr) Function

- 將欄位或運算式加總(數值資料)

```
mysql> SELECT sal
    -> FROM emp
    -> WHERE deptno=30;
+------+
| sal  |
+------+
| 2850 |
| 1250 |
| 1600 |
| 1500 |
|  950 |
| 1250 |
+------+
```

```
mysql> SELECT SUM(sal)
    -> FROM emp
    -> WHERE deptno=30;
+----------+
| SUM(sal) |
+----------+
|     9400 |
+----------+
```

註：NULL Value 皆忽略不計入

AVG(column|expr) Function

- 計算欄位或運算式的平均值(數值資料)

```
mysql> SELECT sal
    ->   FROM emp
    ->  WHERE deptno=30;
+------+
| sal  |
+------+
| 2850 |
| 1250 |
| 1600 |
| 1500 |
|  950 |
| 1250 |
+------+
```

```
mysql> SELECT AVG(sal)
    ->   FROM emp
    ->  WHERE deptno=30;
+-----------+
| AVG(sal)  |
+-----------+
| 1566.6667 |
+-----------+
```

註：NULL Value皆忽略不計入

AVG(column|expr) Function

- 計算公式

```
AVG(column) = SUM(column) / COUNT(column)
```

```
mysql> SELECT comm
    -> FROM emp
    -> WHERE deptno=30;
+------+
| comm |
+------+
| NULL |
| 1400 |
|  300 |
|    0 |
| NULL |
|  500 |
+------+
```

```
mysql> SELECT AVG(comm)
    -> FROM emp
    -> WHERE deptno=30;
+-----------+
| AVG(comm) |
+-----------+
|  550.0000 |
+-----------+
```

註：NULL Value皆忽略不計入

AVG with IFNULL Function

- 平均值計算公式

```
平均值(Average) = SUM(column) / COUNT(*)  或
                = AVG(IFNULL(column|expr, 0))
```

```
mysql> SELECT SUM(comm)/COUNT(*) AVG
    -> FROM emp
    -> WHERE deptno=30;
+--------+
| AVG    |
+--------+
| 366.67 |
+--------+
```

```
mysql> SELECT AVG(IFNULL(comm,0)) AVG
    -> FROM emp
    -> WHERE deptno=30;
+----------+
| AVG      |
+----------+
| 366.6667 |
+----------+
```

使用群組函數(Using Group Functions)

- 使用群組函數做資料彙總

```
mysql> SELECT SUM(SAL), MIN(SAL), MAX(SAL), AVG(SAL), COUNT(*)
    -> FROM emp WHERE deptno=30;
+----------+----------+----------+-----------+----------+
| SUM(SAL) | MIN(SAL) | MAX(SAL) | AVG(SAL)  | COUNT(*) |
+----------+----------+----------+-----------+----------+
|     9400 |      950 |     2850 | 1566.6667 |        6 |
+----------+----------+----------+-----------+----------+
```

資料分組(Grouping Data)

- 依資料內容來分組，分組後再做資料彙總
  - 查詢各部門員工之薪資總和

deptno	sal
10	5000
10	1300
10	2450
20	2975
20	1100
20	3000
20	800
20	3000
30	1250
30	2850
30	1500
30	1600
30	1250
30	950

部門 10 →

部門 20 →

部門 30 →

deptno	SUM(sal)
10	8750
20	10875
30	9400

```
SELECT deptno, sum(sal)
From emp
Group by deptno;
```

資料分組(Creating Groups of Data)

- 使用GROUP BY子句將表格(table)中的記錄(rows)依資料內容分為不同的群組
- 若無 GROUP BY 子句，則整個表格就是一組

EMP

DEPTNO	SAL
10	2450
10	5000
10	1300
20	800
20	1100
20	3000
20	3000
20	2975
30	1600
30	2850
30	1250
30	950
30	1500
30	1250

2916.6667

2175

1566.6667

"average salary in EMP table for each department"

DEPTNO	AVG(SAL)
10	2916.6667
20	2175
30	1566.6667

GROUP BY 子句

- GROUP BY子句將表格(table)中的記錄(rows)依GROUP BY後資料項的相同內容分為一個群組

```
SELECT        column, group_function
FROM          table
[WHERE        condition]
[GROUP BY     group_by_expression]
[HAVING       group_condition]
[ORDER BY     column];
```

- GROUP BY 分組後再做各組的資料彙總
  - ***group_by_expression*** 即為排序的依據(ASC)
  - 可以使用ORDER BY改變預設排序的結果

```
SELECT    deptno, AVG(sal) AS AvgSal
FROM      emp
GROUP BY  deptno;
```

資料分組使用GROUP BY 子句

```
mysql> SELECT deptno, SUM(sal)
    -> FROM emp
    -> GROUP BY deptno;
+--------+----------+
| deptno | SUM(sal) |
+--------+----------+
|     10 |     8750 |
|     20 |    10875 |
|     30 |     9400 |
+--------+----------+
```

有排序(升冪ASC)

若有一般欄位和群組函同時出現在SELECT LIST中時，請勿將一般欄位放置於GROUP BY子句中

```
mysql> SELECT deptno, SUM(sal)
    -> FROM emp;
ERROR 1140 (42000): Mixing of GROUP columns (MIN(),MAX(),COUNT()...)
with no GROUP columns is illegal if there is no GROUP BY clause
```

資料分組 – NULL

- NULL值在 GROUP BY 子句的資料分組

```
mysql> SELECT comm, COUNT(*)
    -> FROM emp
    -> GROUP BY comm;
+------+----------+
| comm | COUNT(*) |
+------+----------+
| NULL |       10 |
|    0 |        1 |
|  300 |        1 |
|  500 |        1 |
| 1400 |        1 |
+------+----------+
```

NULL 自己也算資料分組的一組

資料分組 – 使用ORDER BY

- 改變預設排序的結果

```
mysql> SELECT deptno, SUM(sal)
    -> FROM emp
    -> GROUP BY deptno
    -> ORDER BY 2;
+--------+----------+
| deptno | SUM(sal) |
+--------+----------+
|     10 |     8750 |
|     30 |     9400 |
|     20 |    10875 |
+--------+----------+
3 rows in set (0.00 sec)
```

升冪 ASC

資料分組－多欄位(Grouping of Multiple Column)

- 分成更小的群組資料

```
mysql> SELECT deptno, job, count(*)
    -> FROM emp
    -> GROUP BY deptno, job;
+--------+-----------+----------+
| deptno | job       | count(*) |
+--------+-----------+----------+
|     10 | CLERK     |        1 |
|     10 | MANAGER   |        1 |
|     10 | PRESIDENT |        1 |
|     20 | ANALYST   |        2 |
|     20 | CLERK     |        2 |
|     20 | MANAGER   |        1 |
|     30 | CLERK     |        1 |
|     30 | MANAGER   |        1 |
|     30 | SALESMAN  |        4 |
+--------+-----------+----------+
```

總人員應和 COUNT(*)= 14 相同

篩選分組資料(Filtering Group Results)

EMP

DEPTNO	SAL
10	2450
10	5000
10	1300
20	800
20	1100
20	3000
20	3000
20	2975
30	1600
30	2850
30	1250
30	950
30	1500
30	1250

5000

3000

2850

"maximum salary per department greater than $2900"

DEPTNO	MAX(SAL)
10	5000
20	3000

```
SELECT   deptno, max(sal) SalMax
FROM     emp
GROUP BY deptno
HAVING   max(sal)>2900;
```

分組資料過濾條件(HAVING子句)

- 使用HAVING子句設定條件來篩選回傳的分組資料

```
SELECT        column, group_function
FROM          table
[WHERE        condition]
[GROUP BY     group_by_expression]
[HAVING       group_condition]
[ORDER BY     column];
```

  - WHERE子句中不可以出現群組函數

使用HAVING子句篩選分組資料

- 依職務分類, 列出人數>3的類別

```
mysql> SELECT job, COUNT(*) CNT
    -> FROM emp
    -> GROUP BY job
    -> ORDER BY CNT;
+-----------+-----+
| job       | CNT |
+-----------+-----+
| PRESIDENT |   1 |
| ANALYST   |   2 |
| MANAGER   |   3 |
| SALESMAN  |   4 |
| CLERK     |   4 |
+-----------+-----+
```

```
mysql> SELECT job, COUNT(*) CNT
    -> FROM emp
    -> GROUP BY job
    -> HAVING COUNT(*) > 3;
+----------+-----+
| job      | CNT |
+----------+-----+
| CLERK    |   4 |
| SALESMAN |   4 |
+----------+-----+
```

篩選分組資料的錯誤(Error in Restricting)

```
mysql> SELECT job, COUNT(*) CNT
    ->   FROM emp
    ->  WHERE COUNT(*) > 3
    ->  GROUP BY job;
ERROR 1111 (HY000): Invalid use of group function
```

- WHERE子句中不可以出現群組函數

查詢命令的子句執行的順序
Logical Query Processing

- SELECT 命令的執行順序

```
5: SELECT   <select list>
1: FROM     <table source>
2: WHERE    <search condition>
3: GROUP BY <group by list>
4: HAVING   <search condition>
6: ORDER BY <order by list>
```

資料查詢 – SELECT 執行順序

```
SELECT deptno, round(avg(sal),0), sum(sal), count(*)
FROM emp
WHERE SAL > 1500
GROUP BY deptno
HAVING avg(sal)>2500
ORDER BY COUNT(*) DESC;
```

EMPNO	SAL	DEPTNO
7369	800	20
7499	1600	30
7521	1250	30
7566	2975	20
7654	1250	30
7698	2850	30
7782	2450	10
7788	3000	20
7839	5000	10
7844	1500	30
7876	1100	20
7900	950	30
7902	3000	20
7934	1300	10

Step 1 FROM

Step 2. Where

```
5   SELECT deptno, round(avg(sal),0),
            sum(sal), count(*)
1   FROM emp
2   WHERE SAL > 1500
3   GROUP BY deptno
4   HAVING avg(sal)>2500
6   ORDER BY COUNT(*) DESC;
```

FROM emp

EMPNO	SAL	DEPTNO
7369	800	20
7499	1600	30
7521	1250	30
7566	2975	20
7654	1250	30
7698	2850	30
7782	2450	10
7788	3000	20
7839	5000	10
7844	1500	30
7876	1100	20
7900	950	30
7902	3000	20
7934	1300	10

Where sal > 1500

EMPNO	SAL	DEPTNO
7499	1600	30
7566	2975	20
7698	2850	30
7782	2450	10
7788	3000	20
7839	5000	10
7902	3000	20

Step 3. Group by

```
5   SELECT deptno, round(avg(sal),0),
            sum(sal), count(*)
1   FROM emp
2   WHERE SAL > 1500
3   GROUP BY deptno
4   HAVING avg(sal)>2500
6   ORDER BY COUNT(*) DESC;
```

EMPNO	SAL	DEPTNO
7499	1600	30
7566	2975	20
7698	2850	30
7782	2450	10
7788	3000	20
7839	5000	10
7902	3000	20

Group by deptno

EMPNO	SAL	DEPTNO
7782 7839	2450 5000	10 10
7566 7788 7902	2975 3000 3000	20 20 20
7499 7698	1600 2850	30 30

Step 4. Having

Having avg(sal)>2500

```
5  SELECT deptno, round(avg(sal),0),
           sum(sal), count(*)
1  FROM emp
2  WHERE SAL > 1500
3  GROUP BY deptno
4  HAVING avg(sal)>2500
6  ORDER BY COUNT(*) DESC;
```

EMPNO	SAL	DEPTNO
7782	2450	10
7839	5000	10
7566	2975	20
7788	3000	20
7902	3000	20
~~7499~~	~~1600~~	~~30~~
~~7698~~	~~2850~~	~~30~~

avg(sal)

3725

2991.666666

~~2225~~

Having avg(sal)>2500

EMPNO	SAL	DEPTNO
7782	2450	10
7839	5000	10
7566	2975	20
7788	3000	20
7902	3000	20

Step 5. Select

```
5   SELECT deptno, round(avg(sal),0),
            sum(sal), count(*)
1   FROM emp
2   WHERE SAL > 1500
3   GROUP BY deptno
4   HAVING avg(sal)>2500
6   ORDER BY COUNT(*) DESC;
```

EMPNO	SAL	DEPTNO
7782	2450	10
7839	5000	10
7566	2975	20
7788	3000	20
7902	3000	20

select deptno, round(avg(sal), 0), sum(sal), COUNT(*)

10, 3725, 7450, 2

20, 2992, 8975, 3

deptno	round(avg(sal))	sum(sal)	count(*)
10	3725	7450	2
20	2992	8975	3

Step 6. Order by

```
5  SELECT deptno, round(avg(sal),0),
           sum(sal), count(*)
1  FROM emp
2  WHERE SAL > 1500
3  GROUP BY deptno
4  HAVING avg(sal)>2500
6  ORDER BY COUNT(*) DESC;
```

deptno	Round(avg(sal))	Sum(sal)	Count(*)
10	3725	7450	2
20	2992	8975	3

ORDER BY COUNT(*) DESC

deptno	Round(avg(sal))	Sum(sal)	Count(*)
20	2992	8975	3
10	3725	7450	2

WHERE 跟 HAVING 的比較

- WHERE 過濾篩選分組前表格中的紀錄(Rows)
  - 篩選原始紀錄
- HAVING 過濾篩選分組後的分組資料
  - 篩選分組資料

WHERE 跟 HAVING 的比較

- CASE 1

```
SELECT     job, SUM(sal) PAYROLL
FROM       emp
WHERE      job NOT LIKE 'SALES%'
GROUP BY   job
HAVING     SUM(sal)>5000
ORDER BY   SUM(sal);
```

- CASE 2

```
SELECT     job, SUM(sal) PAYROLL
FROM       emp
GROUP BY   job
HAVING     SUM(sal)>5000 and job NOT LIKE 'SALES%'
ORDER BY   SUM(sal);
```

分組資料的字串連結函數

- GROUP_CONCAT 群組函數

```
GROUP_CONCAT([DISTINCT] expr [,expr ...]
             [ORDER BY {unsigned_integer | column | expr}
                 [ASC | DESC] [,column ...]]
             [SEPARATOR string])
```

- DISTINCT　去除連結字串重複資料
- ORDER BY　排序連結字串
- unsigned_integer　無負號
- column, expr　欄位名稱或運算式
- SEPARATOR　設定連結字串間隔
- string　連結字串間隔字串

分組資料的字串連結函數

- 使用GROUP_CONCAT 群組函數列出各部門所有的職務列表

```
mysql> SELECT deptno, GROUP_CONCAT(job SEPARATOR ',') JOBS
    -> FROM emp
    -> GROUP BY deptno;
+--------+-----------------------------------------------------+
| deptno | JOBS                                                |
+--------+-----------------------------------------------------+
|     10 | PRESIDENT,CLERK,MANAGER                             |
|     20 | MANAGER,CLERK,ANALYST,CLERK,ANALYST                 |
|     30 | SALESMAN,MANAGER,SALESMAN,SALESMAN,SALESMAN,CLERK |
+--------+-----------------------------------------------------+
```

分組資料的字串連結函數

- 列出和部門有關的職務(去除重複資料+排序)

```
mysql> SELECT deptno,
    ->   GROUP_CONCAT(DISTINCT job ORDER BY job ASC SEPARATOR ',') JOBS
    ->   FROM emp
    ->   GROUP BY deptno;
+--------+-------------------------+
| deptno | JOBS                    |
+--------+-------------------------+
|     10 | CLERK,MANAGER,PRESIDENT |
|     20 | ANALYST,CLERK,MANAGER   |
|     30 | CLERK,MANAGER,SALESMAN  |
+--------+-------------------------+
3 rows in set (0.01 sec)
```

作業練習

建立查詢指令以顯示下列各題描述之資料:

1. 顯示所有員工的最高、最低、總和及平均薪資，依序將表頭命名為 Maximum, Minimum, Sum 和 Average，請將結果顯示為四捨五入的整數。
2. 顯示每種職稱的最低、最高、總和及平均薪水。
3. 顯示每種職稱的人數。
4. 顯示資料項命名為Number of Managers來表示擔任主管的人數。
5. 顯示資料項命名為DIFFERENCE的資料來表示公司中最高和最低薪水間的差額。
6. 顯示每位主管的員工編號及該主管下屬員工最低的薪資，排除沒有主管和下屬員工最低薪資少於1000的主管，並以下屬員工最低薪資作降冪排列。
7. 顯示在1980,1981,1982,1983年進公司的員工數量，並給該資料項一個合適的名稱。