Lecture 9: SQL basic 3

BADM/ACCY 352

Spring 2017

Instructor: Yi Yang, PhD

Last lecture

- How to rank, limit results
- Aggregate functions and subquery

- HW2 is due tonight.
- Lab2 on Wednesday.

Explain the following queries.

```sql
SELECT P_CODE, P_PRICE FROM PRODUCT
WHERE P_PRICE > ( SELECT AVG(P_PRICE) FROM PRODUCT)
ORDER BY P_PRICE DESC;
```

```sql
SELECT MAX(P_PRICE) FROM PRODUCT
WHERE P_PRICE != (
    SELECT MAX(P_PRICE)
    FROM PRODUCT
    );
```

GROUPING DATA: GROUP BY

- Sometimes we need to group rows into smaller collections.
- SQL can group certain rows by a column, and each unique value of the column is a group.
- Syntax:

```
SELECT column_name, aggregate_funcation(column_name)
FROM table_name
WHERE conditions
GROUP BY column_name
HAVING conditions;
```

GROUPING DATA: GROUP BY

- For a sales database, you want to know the total sale revenue per day, month, quarter, year, or per store, region.
- For a student database, you want to know student GPA per semester.
- For a stock database, you want to know the stock return per month, quarter, year.

GROUP BY with aggregate function example

emp_bonus

Employee	Bonus
A	1000
B	2000
A	500
C	700
B	1250

GROUP BY
employee column

Employee	Sum(Bonus)
A	1500
B	3250
C	700

```
SELECT employee, SUM(bonus)
FROM emp_bonus
GROUP BY employee;
```

P_CODE	P_DESCRIPT	P_PRICE	V_CODE
11QER/31	Power painter, 15 psi., 3-nozzle	109.99	25595
13-Q2/P2	7.25-in. pwr. saw blade	14.99	21344
14-Q1/L3	9.00-in. pwr. saw blade	17.49	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	39.95	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	43.99	23119
2232/QTY	B&D jigsaw, 12-in. blade	109.92	24288
2232/QWE	B&D jigsaw, 8-in. blade	99.87	24288
2238/QPD	B&D cordless drill, 1/2-in.	38.95	25595
23109-HB	Claw hammer	9.95	21225
23114-AA	Sledge hammer, 12 lb.	14.40	NULL
54778-2T	Rat-tail file, 1/8-in. fine	4.99	21344
89-WRE-Q	Hicut chain saw, 16 in.	256.99	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	5.87	NULL
SM-18277	1.25-in. metal screw, 25	6.99	21225
SW-23116	2.5-in. wd. screw, 50	8.45	21231
WR3/TT3	Steel matting, 4'x8'x1/6", .5" ...	119.95	25595

What if you want to compute the average product price per vendor?

```
SELECT V_CODE, AVG(P_PRICE) FROM PRODUCT
GROUP BY V_CODE;
```

RESULTS:

V_CODE	AVG(P_PRICE)
NULL	10.135000
21225	8.470000
21231	8.450000
21344	12.490000
23119	41.970000
24288	155.593333
25595	89.630000

NOTE the first row

HAVING vs. WHERE

- HAVING is used with aggregate functions to apply constraint conditions.
- WHERE can't be used with aggregated table.

emp_bonus

Employee	Bonus
A	1000
B	2000
A	500
C	700
B	1250

GROUP BY

Employee	Sum(Bonus)
A	1500
B	3250
C	700

HAVING

Employee	Sum(Bonus)
A	1500
B	3250

```
SELECT employee, SUM(bonus) FROM emp_bonus
GROUP BY employee WHERE SUM(bonus) > 1000;
```

NO

```
SELECT employee, SUM(bonus) FROM emp_bonus
GROUP BY employee HAVING SUM(bonus) > 1000;
```

YES

Let's walk through an example

INV_NUMBER	CUS_CODE	INV_DATE
1001	10014	2014-01-16 00:00:00
1002	10011	2014-01-16 01:00:00
1003	10012	2014-01-16 01:00:00
1004	10011	2014-01-17 00:00:00
1005	10018	2014-01-17 01:00:00
1006	10014	2014-01-17 02:00:00
1007	10015	2014-01-17 02:00:00
1008	10011	2014-01-17 02:00:00

1. List all invoices in the INVOICE table.
2. Count the number of invoices that are after '2014-01-17'.
3. Count the total number of invoices per customer, and the invoices satisfy the condition 'after 2014-01-17'.
4. List the cus_code who has more than one invoices after 2014-01-17.

```sql
#1
SELECT * FROM INVOICE;

#2
SELECT COUNT(*) FROM INVOICE WHERE INV_DATE >= '2014-01-17';

#3
SELECT CUS_CODE, COUNT(*) FROM INVOICE
WHERE INV_DATE >= '2014-01-17'
GROUP BY CUS_CODE;

#4
SELECT CUS_CODE, COUNT(*) FROM INVOICE
WHERE INV_DATE >= "2014-01-17"
GROUP BY CUS_CODE
HAVING COUNT(*) > 1;
```

Group by multiple columns

EMP_NUM	EMP_NAME	EMP_GENDER	EMP_DEP
101	Alice	Female	HR
102	Bob	Male	HR
103	Chelsea	Female	IT
104	Doug	Male	IT

1. Count the number of EMP by gender.

```
select count(*), EMP_GENDER from EMP group by EMP_GENDER;
```

2. Count the number of EMP by department.

```
select count(*), EMP_DEP from EMP group by EMP_DEP;
```

3. Count the number of EMP by gender for each department.

```
select count(*), EMP_DEP, EMP_GENDER from EMP
group by EMP_GENDER, EMP_DEP;
```

Put it together

```
SELECT columns
FROM table
[WHERE condition]
[GROUP BY column]
[HAVING condition]
[ORDER BY column [ASC | DESC]]
[LIMIT count]
```

```
SELECT CUS_CODE, COUNT(*) FROM INVOICE
WHERE INV_DATE >= "2014-01-17"
GROUP BY CUS_CODE
HAVING COUNT(*) > 1
ORDER BY CUS_CODE
LIMIT 10;
```

iClicker question

COUNT(*)	EMP_MGR
3	NULL
4	100
6	105
4	108

- Firstly, what does this query do?

```
select COUNT(*), EMP_MGR from EMP GROUP BY EMP_MGR;
```

- If I want to remove the row with NULL value in EMP_MGR column, which of the following is correct?

A:
```
select COUNT(*), EMP_MGR from EMP GROUP BY EMP_MGR
where NOT EMP_MGR IS NULL;
```

B:
```
select COUNT(*), EMP_MGR from EMP GROUP BY EMP_MGR
having NOT EMP_MGR IS NULL;
```

C:
```
select COUNT(*), EMP_MGR from EMP GROUP BY EMP_MGR
having NOT EMP_MGR = NULL;
```

SQL clinic: HAVING vs. WHERE

- Technically, you can use HAVING clause without group by. It “looks the same” as using WHERE clause.
- WHERE clause filters data before SELECT
- HAVING clause filters data after SELECT

```
SELECT EMP_LNAME FROM EMP HAVING EMP_NUM > 107;
```

Syntax Error!!!

```
SELECT EMP_LNAME FROM EMP WHERE EMP_NUM > 107;
```

So, which clause is more efficient if you have many many rows in a table?

Learning objective

- Know how to write query with GROUP BY.

Date/time in SQL

- **MySQL** uses the following data types for storing a date or a date/time value in the database:
  - DATE - format YYYY-MM-DD
  - DATETIME - format: YYYY-MM-DD HH:MI:SS
- December 30th, 2016 would be stored as 2016-12-30 in DATE type.
- 3:30 in the afternoon on December 30th, 2016 would be stored as 2016-12-30 15:30:00.

Date functions

- Some useful functions on DATE data type.
- DAY(): DAY('2000-12-31') = 31
- WEEK(): WEEK('2000-12-31') = 51
- MONTH(): MONTH('2000-12-31') = 12
- QUARTER(): QUARTER('2000-12-31') = 4
- YEAR(): YEAR('2000-12-31') = 2000

- Similarity, you can use Hour(), Minute(), Second() on DATETIME type for finer-granularity calculation.

Other Useful Date functions

- CURRENT_DATE():
  - returns current date
  - how do I get current year, current month?
- TIMESTAMPDIFF():
  - returns a value after subtracting a datetime expression from another.
  - Compute the age of an employee:

```
SELECT EMP_DOB,
TIMESTAMPDIFF(YEAR, EMP_DOB, CURRENT_DATE())
FROM EMP;
```

Date functions with group by

INVOICE table

INV_NUMBER	CUS_CODE	INV_DATE
1001	10014	2014-01-16 00:00:00
1002	10011	2014-01-16 01:00:00
1003	10012	2014-01-16 01:00:00
1004	10011	2014-01-17 00:00:00
1005	10018	2014-01-17 01:00:00
1006	10014	2014-01-17 02:00:00
1007	10015	2014-01-17 02:00:00
1008	10011	2014-01-17 02:00:00

```
SELECT INV_NUMBER, DAY(INV_DATE)
FROM INVOICE;
```

INV_NUMBER	DAY(INV_DATE)
1001	16
1002	16
1003	16
1004	17
1005	17
1006	17
1007	17
1008	17

Return the number of invoice per day.

How about per month, per year? Or maybe per hour?

```
SELECT DAY(INV_DATE), COUNT(*)
FROM INVOICE
GROUP BY DAY(INV_DATE);
```

RESULTS:

DAY(INV_DATE)	COUNT(*)
16	3
17	5

```
SELECT DAY(INV_DATE), HOUR(INV_DATE), COUNT(*)
FROM INVOICE
GROUP BY DAY(INV_DATE), HOUR(INV_DATE);
```

RESULTS:

DAY(INV_DATE)	HOUR(INV_DATE)	COUNT(*)
16	0	1
16	1	2
17	0	1
17	1	1
17	2	3

Learning objective

- Know how to use date functions with aggregate functions and Group By clause.