SQL Group By

Business Database Fundamentals

Contributions by Arthur Adamopoulos and Vince Bruno

Scalar Functions

- Functions that return a single value
- Usable where you would normally use a value.
- Normally used within the column definitions or within where clauses etc.
- MySQL Function Reference: http://dev.mysql.com/doc/refman/5.1/en/functions.html

Scalar Functions*

- NULL Condition checks
  - IFNULL() Null if/else construct
  - NULLIF() Return NULL if expr1 = expr2
- Trimming strings
  - LTRIM() Remove leading spaces
  - RTRIM() Remove trailing spaces
  - TRIM() Remove leading and trailing spaces
- SOUNDEX
  - Creates a code for the the way the value sounds.

Scalar Functions*

- String Case manipulation
  - UPPER() Convert to uppercase
  - LOWER() Return the argument in lowercase
- CONCAT() Return concatenated string
- STRCMP() Compare two strings
- SUBSTR() Return the substring as specified

Working with Dates*

- DATE_ADD() Add two dates
- DATE_FORMAT() Format date as specified
- DATEDIFF() Subtract two dates
- DAY() Return the day of the month (0-31), also DAYOFWEEK() and DAYNAME()
- MONTH() Return the month from the date passed
- YEAR() Return the year
- WEEK() Return the week number

Scalar Functions*

- CASE - Case operator

```
SELECT surname,
    CASE sex WHEN 'F' THEN 'Female'
             WHEN 'M' THEN 'Male'
    ELSE 'Unknown' END as gender
    FROM STAFF;
```

Surname	gender
VELLA	Male
MARZELLA	Male
HILTON	Female
JAMIESON	Female
SANDERS	Male
SUMMERS	Male
KNOL	Fema
PORTELLI	Male
KHOR	Male
SCANLON	Male

- IF() - If/else construct

```
SELECT planname, connectFee,
    IF(connectFee<2,'cheap','expensive') as comment
    FROM plan;
```

PlanName	ConnectFee	comment
Yes10	1.00	cheap
Yes20	1.20	cheap
Yes30	1.50	cheap
Yes40	1.75	cheap
Weekender	2.50	expensive
FreeStyle	3.95	expensive

Statistical Functions

- There are five basic statistical functions.
- They are also known as "grouping" functions.
- All statistical functions return one value only, no matter how many rows they operate on.
- When they are used, values of individual rows cannot be displayed.
- Can be used with usual where clauses.

Statistical Functions - COUNT

- COUNT - Counts number of rows

```
SELECT COUNT(*)
FROM mobile;
```

count(*)
1151

```
SELECT COUNT(*)
FROM mobile
WHERE brandname = 'Nokia' ;
```

count(*)
253

Statistical Functions - COUNT

- COUNT - Counts number of rows

```
SELECT COUNT(*)
FROM customer;
```

count(*)
1120

```
SELECT COUNT(*), surname, given
FROM customer;
```

Mixing of GROUP columns (MIN(),MAX(),COUNT(),...) with no GROUP columns is illegal if there is no GROUP BY clause
[Error Code: 1140]
[SQL State: 42000]

Statistical Functions - SUM

- SUM(…) - Add up values in a specified column for all selected rows.

```
SELECT sum(rateperhour*40) as weeklyPay
FROM staff
WHERE resigned IS NULL;
```

weeklyPay
1887.2

- Result would be a single value of all the rates in the selected rows added up.
- No other field can be shown.

Statistical Functions AVG, MAX, MIN

- AVG
  - Average of all values in a specific column.
- MAX
  - Highest value found for a specific column in all selected rows.
- MIN
  - Lowest value found for a specific column in all selected rows.

Grouping Data - GROUP BY

- Also known as "break" reports.
- A grouping field is selected to group the rows
- The rows are sorted by the grouping field.
- Rows with the same value for the grouping field are treated as a "group".
- Usually a statistical function is also used and applied to each group (eg: SUM).

Grouping Example

- A list of all plan names with a count of the number of mobile phones on them.
- The Query:

```
SELECT planname, count(*)
FROM mobile
WHERE mobileid < 4510
GROUP BY planname;
```

Grouping Example – Raw Data

```
select *
from mobile
where mobileid < 4510
```

MobileID	PhoneNumber	BrandName	Joined	Cancelled	PlanName	PhoneColour	CustomerID	StaffID
4500	413448970	Samsung	1999-03-12	1999-11-02	Yes10	Blue	20002	10
4501	413941923	Nokia	1998-07-15	1999-07-29	Yes30	Transparent	20002	9
4502	410717359	NEC	1999-06-03	2001-01-24	Yes10	Red	20004	9
4503	412256126	Ericson	1998-04-05	1998-11-24	Yes10	Brown	20006	6
4504	410079801	Ericson	1999-05-02	<NULL>	FreeStyle	Grey	20006	7
4505	410589454	NEC	1998-08-14	<NULL>	Yes20	Rainbow	20008	5
4506	411229676	Nokia	1999-04-23	2000-01-22	Yes30	Pink	20008	8
4507	413980788	Nokia	1999-06-25	<NULL>	Yes10	Transparent	20010	7
4508	410783126	Ericson	1998-09-14	<NULL>	Yes20	Pink	20010	3
4509	410329528	Philips	1998-06-17	<NULL>	Yes20	Yellow	20012	9

Grouping Example – Sorted

```
select *
from mobile
where mobileid < 4510
```

MobileID	PhoneNumber	BrandName	Joined	Cancelled	PlanName	PhoneColour	CustomerID	StaffID
4504	410079801	Ericson	1999-05-02	<NULL>	FreeStyle	Grey	20006	7
4500	413448970	Samsung	1999-03-12	1999-11-02	Yes10	Blue	20002	10
4502	410717359	NEC	1999-06-03	2001-01-24	Yes10	Red	20004	9
4503	412256126	Ericson	1998-04-05	1998-11-24	Yes10	Brown	20006	6
4507	413980788	Nokia	1999-06-25	<NULL>	Yes10	Transparent	20010	7
4505	410589454	NEC	1998-08-14	<NULL>	Yes20	Rainbow	20008	5
4508	410783126	Ericson	1998-09-14	<NULL>	Yes20	Pink	20010	3
4509	410329528	Philips	1998-06-17	<NULL>	Yes20	Yellow	20012	9
4501	413941923	Nokia	1998-07-15	1999-07-29	Yes30	Transparent	20002	9
4506	411229676	Nokia	1999-04-23	2000-01-22	Yes30	Pink	20008	8

Grouping Example – Break points

MobileID	PhoneNumber	BrandName	Joined	Cancelled	PlanName	PhoneColour	CustomerID	StaffID
4504	410079801	Ericson	1999-05-02	<NULL>	FreeStyle	Grey	20006	7
4500	413448970	Samsung	1999-03-12	1999-11-02	Yes10	Blue	20002	10
4502	410717359	NEC	1999-06-03	2001-01-24	Yes10	Red	20004	9
4503	412256126	Ericson	1998-04-05	1998-11-24	Yes10	Brown	20006	6
4507	413980788	Nokia	1999-06-25	<NULL>	Yes10	Transparent	20010	7
4505	410589454	NEC	1998-08-14	<NULL>	Yes20	Rainbow	20008	5
4508	410783126	Ericson	1998-09-14	<NULL>	Yes20	Pink	20010	3
4509	410329528	Philips	1998-06-17	<NULL>	Yes20	Yellow	20012	9
4501	413941923	Nokia	1998-07-15	1999-07-29	Yes30	Transparent	20002	9
4506	411229676	Nokia	1999-04-23	2000-01-22	Yes30	Pink	20008	8

Grouping Example – Result

```
SELECT planname, count(*)
FROM mobile
WHERE mobileid < 4510
GROUP BY planname;
```

PlanName	count(*)
FreeStyle	1
Yes10	4
Yes20	3
Yes30	2

GROUP BY – Extra fields

- When GROUP BY is used, the only fields that can be displayed are the grouping field and statistical functions.
- Values for individual rows cannot be displayed.
- If you want to display other fields, a trick is to include the extra field as a secondary grouping field in the GROUP BY clause.

GROUP BY – multiple grouping

- For example, for each plan, you want a further breakdown of counts for each year.

```
SELECT planname, year(joined), count(*)
    FROM mobile
    WHERE mobileid < 4510
    GROUP BY planname, year(joined);
```

PlanName	year(joined)	count(*)
FreeStyle	1999	1
Yes10	1998	1
Yes10	1999	3
Yes20	1998	3
Yes30	1998	1
Yes30	1999	1

GROUP BY – multiple grouping fields

- What is the difference between these two queries?

```
SELECT surname, given, count(*)
    FROM customer
    GROUP BY surname, given;
```

Compared with:

```
SELECT customerID, surname, given, count(*)
    FROM customer
    GROUP BY customerID, surname, given;
```

GROUP BY - MAX

- Find the maximum weekly pay for active staff.

```
SELECT max(rateperhour * 40) as max_weeklypay
FROM staff  WHERE resigned is NULL
```

max_weeklypay
753.2

```
SELECT sex, max(rateperhour * 40) as max_weeklypay
FROM staff  WHERE resigned is NULL
GROUP BY sex;
```

Sex	max_weekly...
F	753.2
M	675.6

Grouping Data – HAVING

- The HAVING clause operates like a WHERE clause, but is applied to the grouping value.
- WHERE is applied to each row before the grouping operation is done.
- HAVING is applied after the grouping is performed and operates on the calculated grouping value (before it is displayed).

GROUP BY – HAVING

- For example, you want to show only the plans which have a count of mobiles which is less than 3.

```
SELECT planname, count(*)
    FROM mobile
    WHERE mobileid < 4510
    GROUP BY planname
    HAVING count(*) < 3;
```

PlanName	count(*)
FreeStyle	1
Yes30	2

Count(*) – Simple

MobileID	PhoneNumber	BrandName	Joined	Cancelled	PlanName	PhoneColour	CustomerID	StaffID
4500	413448970	Samsung	1999-03-12	1999-11-02	Yes10	Blue	20002	10
4501	413941923	Nokia	1998-07-15	1999-07-29	Yes30	Transparent	20002	9
4502	410717359	NEC	1999-06-03	2001-01-24	Yes10	Red	20004	9
4503	412256126	Ericson	1998-04-05	1998-11-24	Yes10	Brown	20006	6
4504	410079801	Ericson	1999-05-02	<NULL>	FreeStyle	Grey	20006	7
4505	410589454	NEC	1998-08-14	<NULL>	Yes20	Rainbow	20008	5
4506	411229676	Nokia	1999-04-23	2000-01-22	Yes30	Pink	20008	8
4507	413980788	Nokia	1999-06-25	<NULL>	Yes10	Transparent	20010	7
4508	410783126	Ericson	1998-09-14	<NULL>	Yes20	Pink	20010	3
4509	410329528	Philips	1998-06-17	<NULL>	Yes20	Yellow	20012	9

```
SELECT count(*)
  FROM mobile
  WHERE mobileid < 4510;
```

count(*)
10

```
SELECT count(*)
  FROM mobile
  WHERE mobileid < 4510
  AND planname = 'Yes10';
```

count(*)
4

COUNT Column – NULL & DISTINCT

- COUNT(column)
  - function returns the number of rows without a NULL value in the specified column.

- COUNT(DISTINCT column)
  - the keyword COUNT and DISTINCT can be used together to count the number of distinct results.

Count(column) – NULL

MobileID	PhoneNumber	BrandName	Joined	Cancelled	PlanName	PhoneColour	CustomerID	StaffID
4500	413448970	Samsung	1999-03-12	1999-11-02	Yes10	Blue	20002	10
4501	413941923	Nokia	1998-07-15	1999-07-29	Yes30	Transparent	20002	9
4502	410717359	NEC	1999-06-03	2001-01-24	Yes10	Red	20004	9
4503	412256126	Ericson	1998-04-05	1998-11-24	Yes10	Brown	20006	6
4504	410079801	Ericson	1999-05-02	<NULL>	FreeStyle	Grey	20006	7
4505	410589454	NEC	1998-08-14	<NULL>	Yes20	Rainbow	20008	5
4506	411229676	Nokia	1999-04-23	2000-01-22	Yes30	Pink	20008	8
4507	413980788	Nokia	1999-06-25	<NULL>	Yes10	Transparent	20010	7
4508	410783126	Ericson	1998-09-14	<NULL>	Yes20	Pink	20010	3
4509	410329528	Philips	1998-06-17	<NULL>	Yes20	Yellow	20012	9

```
SELECT count(cancelled)
    FROM mobile
    WHERE mobileid < 4510;
```

count(cancelled)
5

Count(Distinct column)

MobileID	PhoneNumber	BrandName	Joined	Cancelled	PlanName	PhoneColour	CustomerID	StaffID
4500	413448970	Samsung	1999-03-12	1999-11-02	Yes10	Blue	20002	10
4501	413941923	Nokia	1998-07-15	1999-07-29	Yes30	Transparent	20002	9
4502	410717359	NEC	1999-06-03	2001-01-24	Yes10	Red	20004	9
4503	412256126	Ericson	1998-04-05	1998-11-24	Yes10	Brown	20006	6
4504	410079801	Ericson	1999-05-02	<NULL>	FreeStyle	Grey	20006	7
4505	410589454	NEC	1998-08-14	<NULL>	Yes20	Rainbow	20008	5
4506	411229676	Nokia	1999-04-23	2000-01-22	Yes30	Pink	20008	8
4507	413980788	Nokia	1999-06-25	<NULL>	Yes10	Transparent	20010	7
4508	410783126	Ericson	1998-09-14	<NULL>	Yes20	Pink	20010	3
4509	410329528	Philips	1998-06-17	<NULL>	Yes20	Yellow	20012	9

```
SELECT count(DISTINCT phonecolour)
    FROM mobile
    WHERE mobileid < 4510;
```

count(distinct phonecolour)
8

Summary

- Query Structure

SELECT … FROM … WHERE … ORDER BY …
GROUP BY field1, field2, ….
HAVING group_condition1 AND/OR group_condition2 … ;

- What can be displayed in the following GROUP BY query ?

SELECT _____, _____, _____, _____, _____, _____
FROM customer
GROUP BY suburb;