

SQL Programming

Groping Our Way through Grouping

In the module on SET FUNCTIONS we learned that set functions operate on a group of rows and return a single row for that group.

But, if you ask me, the grouping structure was somewhat weak. There were only two groups, or grouping mechanisms that we could use:

- 1.The whole table, or
- 2.Those rows of the table that were included as part of a WHERE predicate expression.

And, as you recall, with either of these options, the result table included a single row.

In this module we'll learn how to deal with multiple groups at the same time.

Module 13: Grouping

Page B-1 GROUP BY - Example

Take a moment to examine this SQL program.

The first thing you probably noticed in this program was the new clause: GROUP BY.

The GROUP BY clause specifies the column(s) that will be used to group the data.

In this example, the grouping column is the home_country column, and SQL will group the data based on each of the distinct values that occurs in the home_country column.

In the talent table, the home_country column contains 5 distinct values, and as we can see from the result table, summary information (COUNT) has been provided for each of these groups.

The screenshot shows the iSQL*Plus web interface in a Microsoft Internet Explorer browser window. The browser title is "iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High...". The interface includes a menu bar (File, Edit, View, Favorites, Tools, Help) and a toolbar with icons for Password, Log Out, and Help. Below the toolbar, there is a "Script Location:" field with a "Browse..." button and a "Load Script" button. The "Enter statements:" section contains the following SQL query:

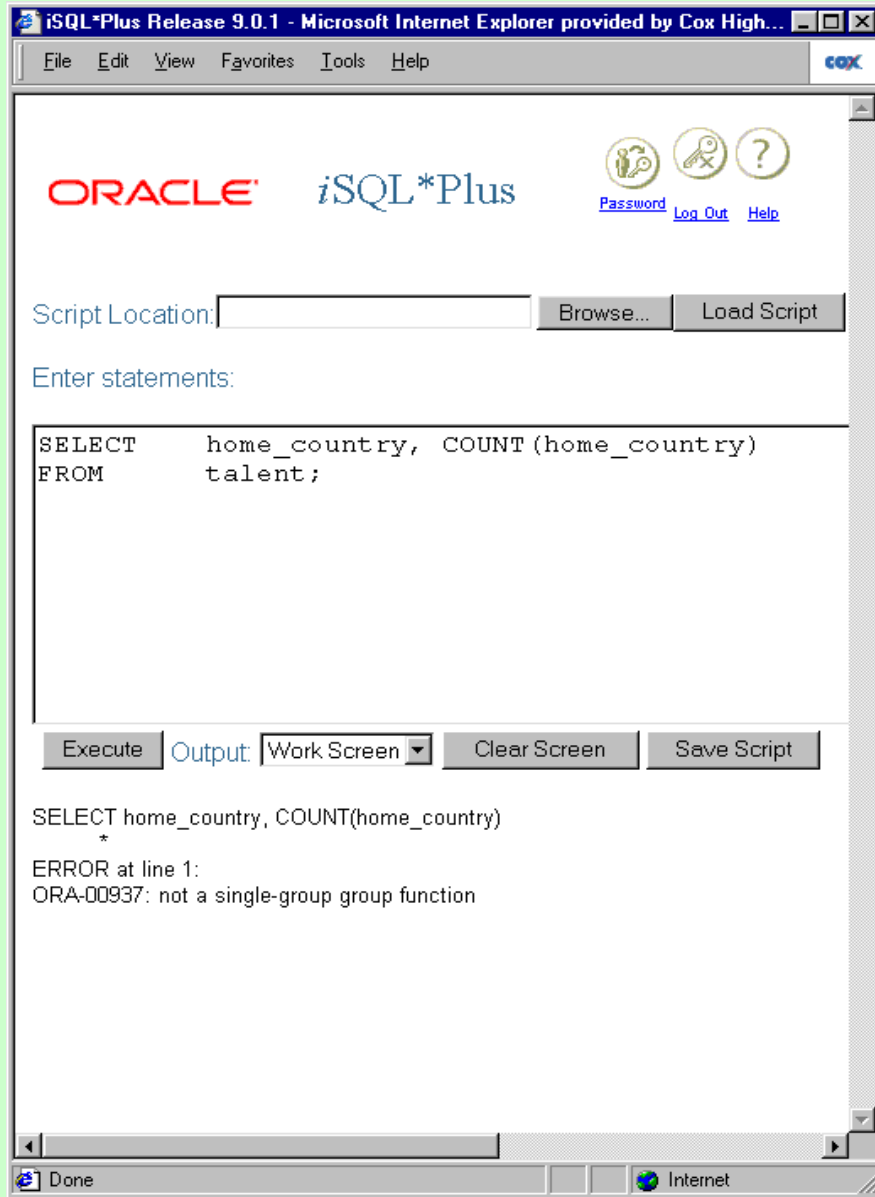
```
SELECT    home_country, COUNT(home_country)
FROM      talent
GROUP BY  home_country;
```

Below the query, there are buttons for "Execute", "Output:" (with a dropdown menu set to "Work Screen"), "Clear Screen", and "Save Script". The results are displayed in a table with two columns: "HOME_COUNTRY" and "COUNT(HOME_COUNTRY)".

HOME_COUNTRY	COUNT(HOME_COUNTRY)
Austria	1
Germany	1
Ireland	1
UK	2
USA	20

The browser status bar at the bottom shows "Done" and "Internet".

Module 13: Grouping



Page B-2 GROUP BY - usage

Grouping isn't an overly complicated concept, but the syntax is rather precise, so as you work on your SQL programs, expect that you'll be generating a LOT more syntax errors with these problem sets.

The SQL syntax requires that each column expression that is listed in the SELECT clause must also be named in the GROUP BY clause, or that column must be a group function.

The error message on this slide cryptically warns you that you're mixing detail with summary information. For obviously `home_country` is not a single-group group function 😊

If we eliminate the double-speak in the Oracle error message, the warning becomes intelligible (almost)

not a group function!

Module 13: Grouping

The screenshot shows the iSQL*Plus web interface in a Microsoft Internet Explorer browser window. The browser title is "iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High...". The interface includes a menu bar (File, Edit, View, Favorites, Tools, Help) and a toolbar with icons for Password, Log Out, and Help. The main content area has a "Script Location:" field with a "Browse..." button and a "Load Script" button. Below this is a text area for "Enter statements:" containing the following SQL query:

```
SELECT    home_country, COUNT(home_country)
FROM      talent
GROUP BY  home_country;
```

Below the query area are buttons for "Execute", "Output:" (with a dropdown menu set to "Work Screen"), "Clear Screen", and "Save Script". The results are displayed in a table with two columns: "HOME_COUNTRY" and "COUNT(HOME_COUNTRY)".

HOME_COUNTRY	COUNT(HOME_COUNTRY)
Austria	1
Germany	1
Ireland	1
UK	2
USA	20

The browser status bar at the bottom shows "Done" and "Internet".

Page B-3 Conceptually

Conceptually you can think of SQL as creating a number of intermediate tables, one for each data value in the group.

These tables will be processed individually in order to calculate the statistics you've requested, and then the single result row from each of these tables showing that statistic, will be gathered together and presented in a single result table.

In this example, consider the FROM clause as building the 1st working table with all the rows from the base table (talent). If there were a WHERE clause in this program, it would trigger immediately after the FROM clause and create yet another intermediate working table with just the 'keeper' rows.

From this intermediate working table, think of SQL as building 5 more working tables, one for each of the values in the home_country column: Austria, Germany, Ireland, UK, and USA. Each of these tables will be processed, the single row statistic will be generated, and then these single row results will be combined in the final result table that is presented on screen.

Module 13: Grouping

Page B-4 Related to DISTINCT

GROUP BY generates groups in pretty much the same fashion as DISTINCT. And as we saw with DISTINCT, many columns can be selected.

In this example we want a breakdown of talent, by home_state, within home_country.

The screenshot shows the iSQL*Plus interface in a Microsoft Internet Explorer window. The title bar reads "iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High...". The menu bar includes File, Edit, View, Favorites, Tools, and Help. The main text area contains the following SQL query:

```
SELECT home_country, home_state, COUNT(*)
FROM talent
GROUP BY home_country, home_state;
```

Below the query area are buttons for "Execute", "Output: Work Screen", "Clear Screen", and "Save Script". The results are displayed in a table with three columns: HOME_COUNTRY, HOME_STATE, and COUNT(*). There are 18 rows of data. At the bottom left, it says "18 rows selected." The status bar at the bottom shows "Done" and "Internet".

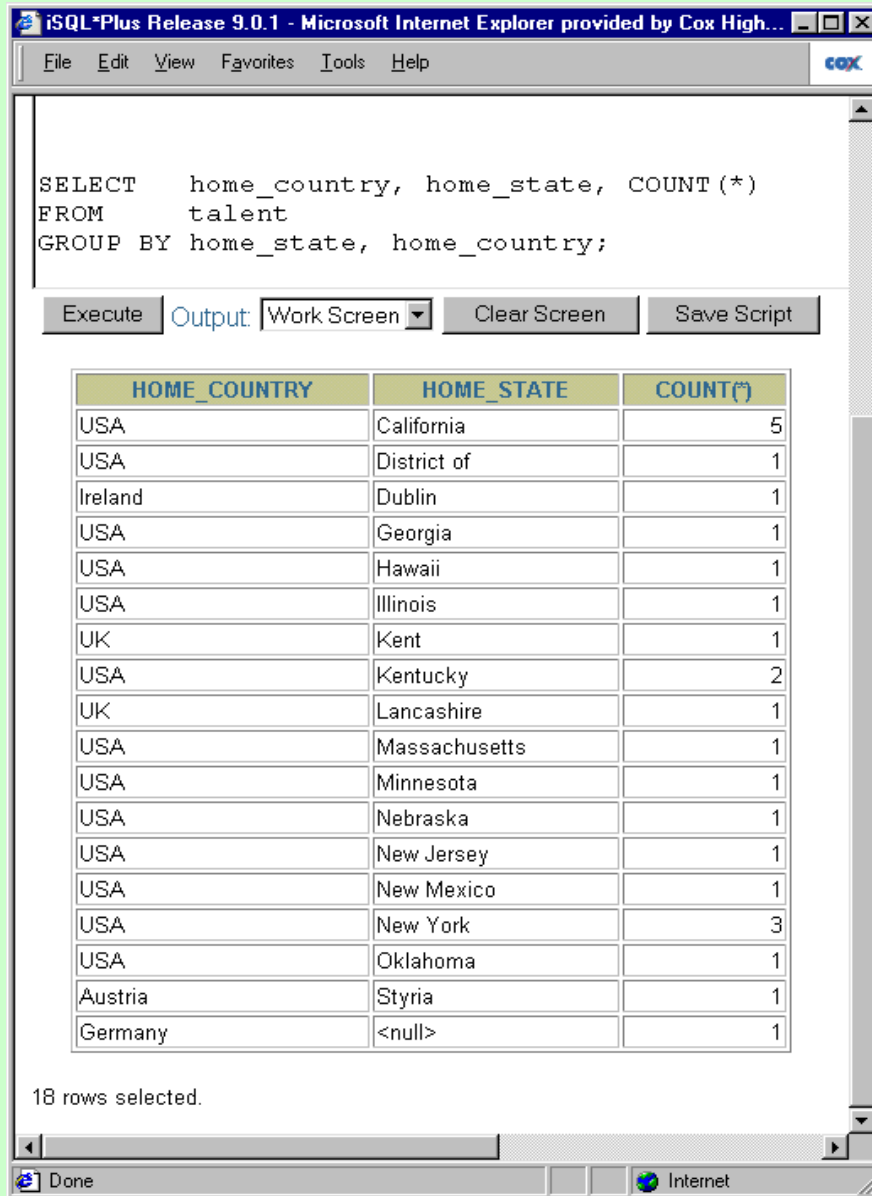
HOME_COUNTRY	HOME_STATE	COUNT(*)
Austria	Styria	1
Germany	<null>	1
Ireland	Dublin	1
UK	Kent	1
UK	Lancashire	1
USA	California	5
USA	District of	1
USA	Georgia	1
USA	Hawaii	1
USA	Illinois	1
USA	Kentucky	2
USA	Massachusetts	1
USA	Minnesota	1
USA	Nebraska	1
USA	New Jersey	1
USA	New Mexico	1
USA	New York	3
USA	Oklahoma	1

Module 13: Grouping

Page B-5 GROUP BY Sequence

Most vendor implementations of SQL will display grouped results in sorted (ascending) order based on the list of column names in the GROUP BY clause.

But do note, that this is not a feature of the standard, and once again, if this is something that is critical to your application, you should include an ORDER BY clause.



The screenshot shows the iSQL*Plus interface in a Microsoft Internet Explorer window. The title bar reads "iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High...". The menu bar includes File, Edit, View, Favorites, Tools, and Help. The main area contains the following SQL query:

```
SELECT home_country, home_state, COUNT(*)
FROM talent
GROUP BY home_state, home_country;
```

Below the query are buttons for "Execute", "Output: Work Screen", "Clear Screen", and "Save Script". The results are displayed in a table with three columns: HOME_COUNTRY, HOME_STATE, and COUNT(*). There are 18 rows of data. At the bottom, it says "18 rows selected." and the status bar shows "Done" and "Internet".

HOME_COUNTRY	HOME_STATE	COUNT(*)
USA	California	5
USA	District of	1
Ireland	Dublin	1
USA	Georgia	1
USA	Hawaii	1
USA	Illinois	1
UK	Kent	1
USA	Kentucky	2
UK	Lancashire	1
USA	Massachusetts	1
USA	Minnesota	1
USA	Nebraska	1
USA	New Jersey	1
USA	New Mexico	1
USA	New York	3
USA	Oklahoma	1
Austria	Styria	1
Germany	<null>	1

Module 13: Grouping

Page B-6 ORDER BY

This example demonstrates the use of the ORDER BY clause in conjunction with GROUPING.

iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High...

File Edit View Favorites Tools Help

```
SELECT home_country, home_state, COUNT(*)
FROM talent
GROUP BY home_state, home_country
ORDER BY home_state, home_country;
```

Execute Output: Work Screen Clear Screen Save Script

HOME_COUNTRY	HOME_STATE	COUNT(*)
USA	California	5
USA	District of	1
Ireland	Dublin	1
USA	Georgia	1
USA	Hawaii	1
USA	Illinois	1
UK	Kent	1
USA	Kentucky	2
UK	Lancashire	1
USA	Massachusetts	1
USA	Minnesota	1
USA	Nebraska	1
USA	New Jersey	1
USA	New Mexico	1
USA	New York	3
USA	Oklahoma	1
Austria	Styria	1
Germany	<null>	1

18 rows selected.

Done Internet

Module 13: Grouping

Page B-7 ORDER BY (cont)

As you might expect, the sort order does not have to jibe with the GROUP BY specification.

The previous slide demonstrated this SQL code:

```
SELECT    home_country, home_state, COUNT(*)
FROM      talent
GROUP BY  home_state, home_country
ORDER BY  home_state, home_country;
```

The screenshot shows the iSQL*Plus interface in a Microsoft Internet Explorer window. The SQL query is executed, and the results are displayed in a table with 18 rows. The table has three columns: HOME_COUNTRY, HOME_STATE, and COUNT(*). The data is sorted by HOME_STATE and then HOME_COUNTRY. The status bar at the bottom indicates '18 rows selected.' and 'Done'.

```
SELECT    home_country, home_state, COUNT(*)
FROM      talent
GROUP BY  home_state, home_country
ORDER BY  home_state, home_country;
```

HOME_COUNTRY	HOME_STATE	COUNT(*)
Austria	Styria	1
Germany	<null>	1
Ireland	Dublin	1
UK	Kent	1
UK	Lancashire	1
USA	California	5
USA	District of	1
USA	Georgia	1
USA	Hawaii	1
USA	Illinois	1
USA	Kentucky	2
USA	Massachusetts	1
USA	Minnesota	1
USA	Nebraska	1
USA	New Jersey	1
USA	New Mexico	1
USA	New York	3
USA	Oklahoma	1

18 rows selected.

Module 13: Grouping

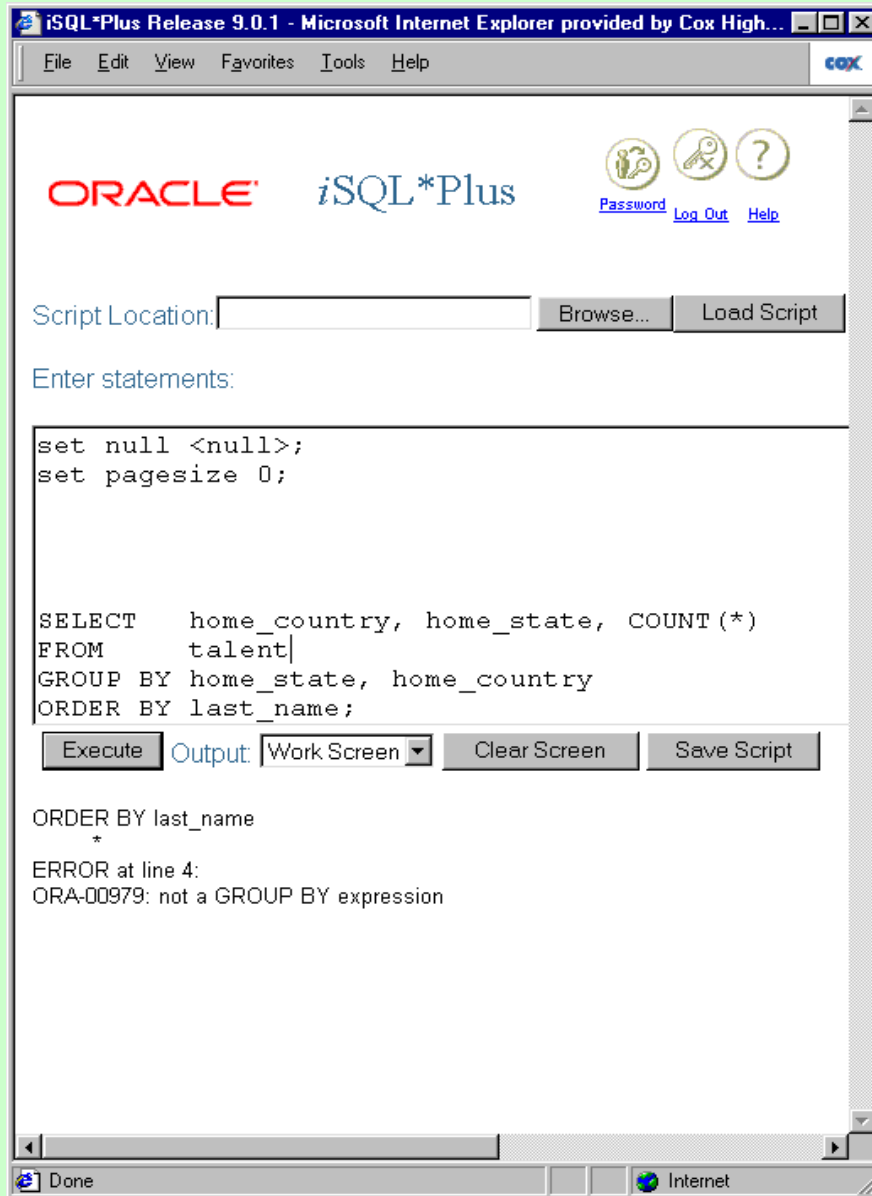
Page B-8 ORDER BY (cont)

But notice what's happening in this example.

We observed earlier that all of the columns of the base table were available to the ORDER BY clause, but that doesn't seem to be the case any more.

And indeed, it is not. Once SQL makes the break to working tables based on group values, the detail columns disappear. Only group level items cascade from this point forward.

But as the next pair of slides demonstrate, all of the group level items remain available to the ORDER BY clause.



Module 13: Grouping

iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High...

File Edit View Favorites Tools Help

```
SELECT home_country, home_state, COUNT(*)
FROM talent
GROUP BY home_state, home_country
ORDER BY COUNT(*);
```

Execute Output: Work Screen Clear Screen Save Script

HOME_COUNTRY	HOME_STATE	COUNT(*)
USA	District of	1
Ireland	Dublin	1
USA	Georgia	1
USA	Illinois	1
UK	Kent	1
USA	Massachusetts	1
USA	Nebraska	1
USA	New Mexico	1
USA	Oklahoma	1
Germany	<null>	1
Austria	Styria	1
USA	New Jersey	1
USA	Minnesota	1
UK	Lancashire	1
USA	Hawaii	1
USA	Kentucky	2
USA	New York	3
USA	California	5

18 rows selected.

Done Internet

Page B-9 ORDER BY (cont)

iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High...

File Edit View Favorites Tools Help

```
SELECT home_country, home_state
FROM talent
GROUP BY home_state, home_country
ORDER BY COUNT(*);
```

Execute Output: Work Screen Clear Screen Save Script

HOME_COUNTRY	HOME_STATE
USA	District of
Ireland	Dublin
USA	Georgia
USA	Illinois
UK	Kent
USA	Massachusetts
USA	Nebraska
USA	New Mexico
USA	Oklahoma
Germany	<null>
Austria	Styria
USA	New Jersey
USA	Minnesota
UK	Lancashire
USA	Hawaii
USA	Kentucky
USA	New York
USA	California

18 rows selected.

Done Internet

Module 13: Grouping

iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High...

File Edit View Favorites Tools Help

```
SELECT home_country, home_state, COUNT(*)
FROM talent
GROUP BY home_state, home_country
ORDER BY COUNT(*) ;
```

Execute Output: Work Screen Clear Screen Save Script

HOME_COUNTRY	HOME_STATE	COUNT(*)
USA	District of	1
Ireland	Dublin	1
USA	Georgia	1
USA	Illinois	1
UK	Kent	1
USA	Massachusetts	1
USA	Nebraska	1
USA	New Mexico	1
USA	Oklahoma	1
Germany	<null>	1
Austria	Styria	1
USA	New Jersey	1
USA	Minnesota	1
UK	Lancashire	1
USA	Hawaii	1
USA	Kentucky	2
USA	New York	3
USA	California	5

18 rows selected.

Done Internet

Page C-1 Restricting rows?

We've seen that the WHERE clause identifies 'keeper' rows, that is, rows that are carried forward from the base table for additional processing by SQL.

There ought to be a similar convention for carrying rows forward from the GROUPed intermediate working tables.

For example, we might be interested in seeing only those rows where the COUNT is greater than 1.

Module 13: Grouping

Page C-2 HAVING

The HAVING clause operates similarly to the WHERE clause.

The WHERE clause identifies which rows carry forward from the base table thru the intermediate working tables.

The HAVING clause identifies which result rows, from GROUP-intermediate working tables are carried forward to the final result table.

The screenshot shows the iSQL*Plus web interface in a Microsoft Internet Explorer browser window. The browser title is "iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High...". The interface includes a menu bar (File, Edit, View, Favorites, Tools, Help) and a toolbar with icons for Password, Log Out, and Help. Below the toolbar, there is a "Script Location:" field with a "Browse..." button and a "Load Script" button. A text area labeled "Enter statements:" contains the following SQL query:

```
SELECT  home_country, home_state, COUNT(*)
FROM    talent
GROUP BY home_state, home_country
HAVING  COUNT(*) > 1;
```

Below the text area, there are buttons for "Execute", "Output:" (with a dropdown menu set to "Work Screen"), "Clear Screen", and "Save Script". The results of the query are displayed in a table with three columns: HOME_COUNTRY, HOME_STATE, and COUNT(*).

HOME_COUNTRY	HOME_STATE	COUNT(*)
USA	California	5
USA	Kentucky	2
USA	New York	3

The browser's status bar at the bottom shows "Done" and "Internet".

Module 13: Grouping

Page C-3 ORDER BY (again)

The ORDER BY clause allows us to arrange the rows in the result table in whatever fashion suits us best.

The screenshot shows the iSQL*Plus web interface in a Microsoft Internet Explorer browser window. The browser title is "iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High...". The interface includes a menu bar (File, Edit, View, Favorites, Tools, Help) and a toolbar with icons for Password, Log Out, and Help. Below the toolbar, there is a "Script Location:" field with a "Browse..." button and a "Load Script" button. The "Enter statements:" section contains the following SQL query:

```
SELECT home_country, home_state, COUNT(*)
FROM talent
GROUP BY home_state, home_country
HAVING COUNT(*) > 1
ORDER BY COUNT(*) ASC;
```

Below the query, there are buttons for "Execute", "Output:" (with a dropdown menu set to "Work Screen"), "Clear Screen", and "Save Script". The results are displayed in a table with three columns: HOME_COUNTRY, HOME_STATE, and COUNT(*).

HOME_COUNTRY	HOME_STATE	COUNT(*)
USA	Kentucky	2
USA	New York	3
USA	California	5

The browser status bar at the bottom shows "Done" and "Internet".

Module 13: Grouping

Page D-1 HAVING

Here's another question for the discussion forum.

← What's going on with this program?

The screenshot shows the iSQL*Plus web interface in a Microsoft Internet Explorer browser window. The browser title is "iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High...". The interface includes a menu bar (File, Edit, View, Favorites, Tools, Help) and a toolbar with icons for Password, Log Out, and Help. The main content area has a "Script Location:" field with a "Browse..." button and a "Load Script" button. Below this is a text area for "Enter statements:" containing the following SQL query:

```
SELECT COUNT(home_country)
FROM talent
HAVING COUNT(home_country) > 1
```

At the bottom of the interface, there are buttons for "Execute", "Output:" (with a dropdown menu set to "Work Screen"), "Clear Screen", and "Save Script". Below these buttons is a table displaying the query results:

COUNT(HOME_COUNTRY)
25

The browser's status bar at the bottom shows "Done" and "Internet".

MySQL provides a very novel, and very useful, group function that can be used to create a 'list'.

Note: This is a group function, and it could have been presented in the previous lecture but I didn't want to show you this until after you'd had a chance to learn about GROUPing.

This function might appear to be at odds with the intuitions you've been developing – but please bear with me.

	Oracle	MySQL
List		GROUP_CONCAT

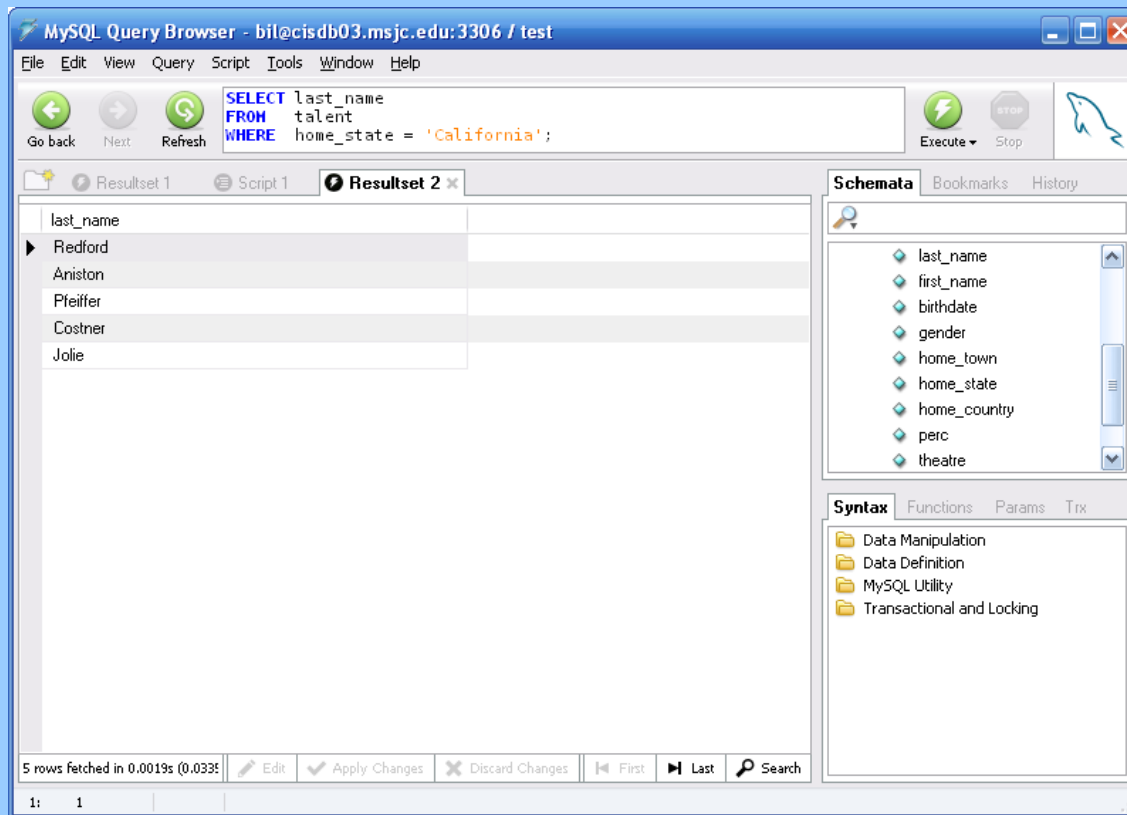
Module 13: Grouping

Page E-2 Vendor Extensions

MySQL allows the programmer to create a list of items based on the values in a column that is being grouped.

Let's say I wanted to know the names of all of the actors in the Talens agency who live in California.

That's pretty simple.



Module 13: Grouping

Page E-3 Vendor Extensions

But now let's say I've been asked to prepare a report that requires the use of GROUPing – say something like, what is the breakdown by state, AND show us who's living there...

I'm thinking of a report like this

Just focus on the report, the result table

Here's a bigger image...

The screenshot shows the MySQL Query Browser interface. The query executed is:

```
SELECT count(*), home_state, group_concat(last_name)
FROM talent t
GROUP BY home_state;
```

The result set, titled 'Resultset 2', displays the following data:

count(*)	home_state	group_concat(last_name)
1	NULL	Willis
5	California	Costner,Pfeiffer,Jolie,Aniston,Redford
1	District of Columbia	Jackson
1	Dublin	Farrell
1	Georgia	Roberts
1	Hawaii	Kidman
1	Illinois	Ford
1	Kent	Bloom
2	Kentucky	Depp,Clooney
1	Lancashire	McKellen
1	Massachusetts	Wahlberg
1	Minnesota	Ryder
1	Nebraska	Brando
1	New Jersey	Harris
1	New Mexico	Moore
3	New York	Sarandon,Pacino,Cruise
1	Oklahoma	Pitt
1	Styria	Schwarzenegger

The interface also shows a 'Schemata' panel on the right with a list of columns: last_name, first_name, birthdate, gender, home_town, home_state, home_country, perc, and theatre. The 'Syntax' panel at the bottom lists categories: Data Manipulation, Data Definition, MySQL Utility, and Transactional and Locking.

MySQL Query Browser - bil@cisdb03.msjc.edu:3306 / test

File Edit View Query Script Tools Window Help

Go back Next Refresh

```
SELECT count(*), home_state, group_concat(last_name)
FROM talent t
GROUP BY home_state;
```

Execute Stop

Resultset 1 Script 1 Resultset 2 x

count(*)	home_state	group_concat(last_name)
1	NULL	Willis
5	California	Costner,Pfeiffer,Jolie,Aniston,Redford
1	District of Columbia	Jackson
1	Dublin	Farrell
1	Georgia	Roberts
1	Hawaii	Kidman
1	Illinois	Ford
1	Kent	Bloom
2	Kentucky	Depp,Clooney
1	Lancashire	McKellen
1	Massachusetts	Wahlberg
1	Minnesota	Ryder
1	Nebraska	Brando
1	New Jersey	Harris
1	New Mexico	Moore
3	New York	Sarandon,Pacino,Cruise
1	Oklahoma	Pitt
1	Styria	Schwarzenegger

18 rows fetched in 0.0052s (0.03)

Edit Apply Changes Discard Changes First Last Search

Schemata Bookmarks History

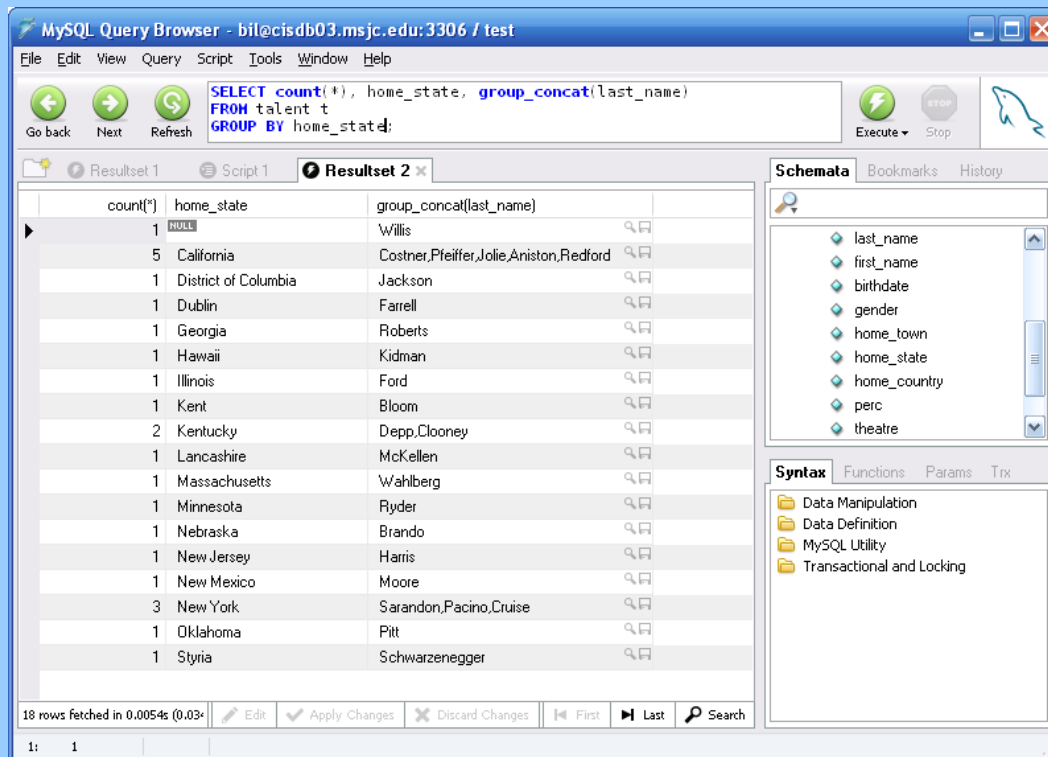
- last_name
- first_name
- birthdate
- gender
- home_town
- home_state
- home_country
- perc
- theatre

Syntax Functions Params Trx

- Data Manipulation
- Data Definition
- MySQL Utility
- Transactional and Locking

We created this report using the GROUP_CONCAT function.

The function creates a list. The list itself is a single item (it's displayed in a single column), and this list is composed as the concatenation of each of the named elements in that group.



The screenshot shows the MySQL Query Browser interface. The query executed is:

```
SELECT count(*), home_state, group_concat(last_name)
FROM talent t
GROUP BY home_state;
```

The result set, labeled 'Resultset 2', displays 18 rows. The columns are 'count(*)', 'home_state', and 'group_concat(last_name)'. The data shows the count of actors for each home state, with their names concatenated into a single string.

count(*)	home_state	group_concat(last_name)
1	NULL	Willis
5	California	Costner,Pfeiffer,Jolie,Aniston,Redford
1	District of Columbia	Jackson
1	Dublin	Farrell
1	Georgia	Roberts
1	Hawaii	Kidman
1	Illinois	Ford
1	Kent	Bloom
2	Kentucky	Depp,Clooney
1	Lancashire	McKellen
1	Massachusetts	Wahlberg
1	Minnesota	Ryder
1	Nebraska	Brando
1	New Jersey	Harris
1	New Mexico	Moore
3	New York	Sarandon,Pacino,Cruise
1	Oklahoma	Pitt
1	Styria	Schwarzenegger

The interface also shows a 'Schemata' panel on the right with a list of tables: last_name, first_name, birthdate, gender, home_town, home_state, home_country, perc, and theatre. The 'Syntax' panel at the bottom right lists categories: Data Manipulation, Data Definition, MySQL Utility, and Transactional and Locking.

This collection of items might appear to be at odds with what we learned earlier. That is, GROUPing doesn't let you look at detail values.

That's still the case. When GROUPing, the detail columns essentially disappear and information is no longer accessible at the single row level. The only data values we can access are:

- 1) the ones that are named in the GROUP BY clause, and
- 2) The aggregate functions

The screenshot shows the MySQL Query Browser interface. The query executed is:

```
SELECT count(*), home_state, group_concat(last_name)
FROM talent t
GROUP BY home_state;
```

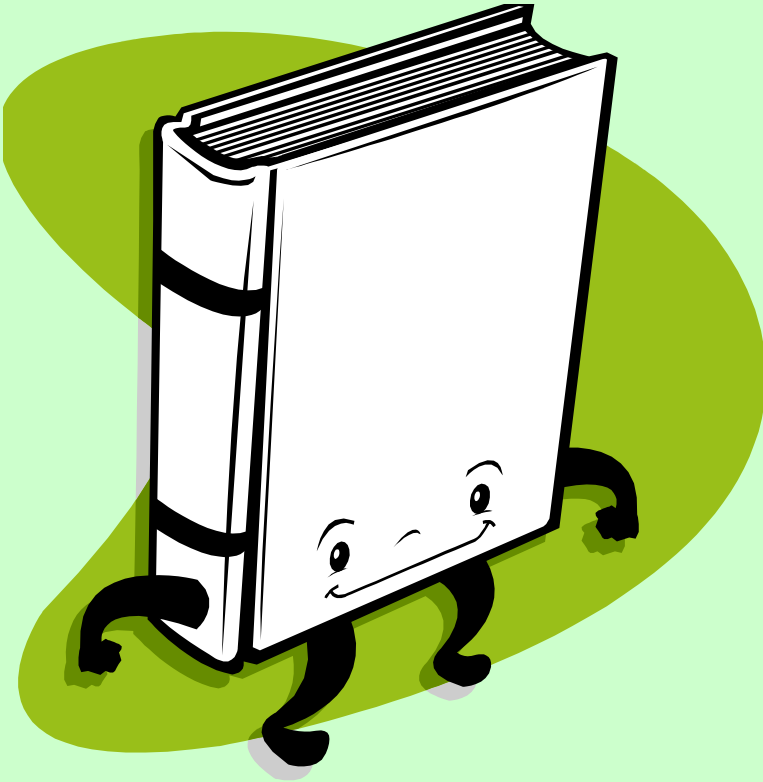
The result set, labeled 'Resultset 2', displays the following data:

count(*)	home_state	group_concat(last_name)
1	NULL	Willis
5	California	Costner,Pfeiffer,Jolie,Aniston,Redford
1	District of Columbia	Jackson
1	Dublin	Farrell
1	Georgia	Roberts
1	Hawaii	Kidman
1	Illinois	Ford
1	Kent	Bloom
2	Kentucky	Depp,Clooney
1	Lancashire	McKellen
1	Massachusetts	Wahlberg
1	Minnesota	Ryder
1	Nebraska	Brando
1	New Jersey	Harris
1	New Mexico	Moore
3	New York	Sarandon,Pacino,Cruise
1	Oklahoma	Pitt
1	Styria	Schwarzenegger

The status bar at the bottom indicates '18 rows fetched in 0.0054s (0.03s)'.

GROUP BY
HAVING

List
GROUP_CONCAT





Please drop me an email if you noticed any errors in this module. I'd also appreciate reading your comments, criticisms, and or suggestions as to how this module could be improved.

Thanks,

bil

That's All