

SQL Programming

Subqueries

It seems that as we progress thru the SQL material, things just keep getting cooler and cooler! (And yes, that is my propeller hat ;-)

In this module we look at one of the truly distinctive features of SQL – the fact that it is a closed system.

But before I attempt to explain the concept of closed systems, we need to cover some background material.



SQL provides us with yet another handy tool to simplify compound expressions.

Rather than writing:

```
WHERE    major_code = 05011
OR       major_code = 08011
OR       major_code = 09351
OR       major_code = 15011
```

we can use the IN operator, and consolidate our code:

```
WHERE major_code
      IN (05011, 08011, 09351, 15011)
```

This comparison operator uses a list structure which is unlike any of the operators that we've seen thus far.

column_item IN (list of *column_items*)

On a very cursory level we might be tempted to say that this operator merely provides a short cut for the programmer who is crafting a compound condition.

However, the IN operator is very much more than merely a short cut technique.

IN allows the programmer to specify a list, or more precisely, a *set* of values to test against.

The magic word in this description is the word: SET, and in a few moments we'll see the tremendous power and flexibility that this simple operator provides.

But first, a few examples. 😊

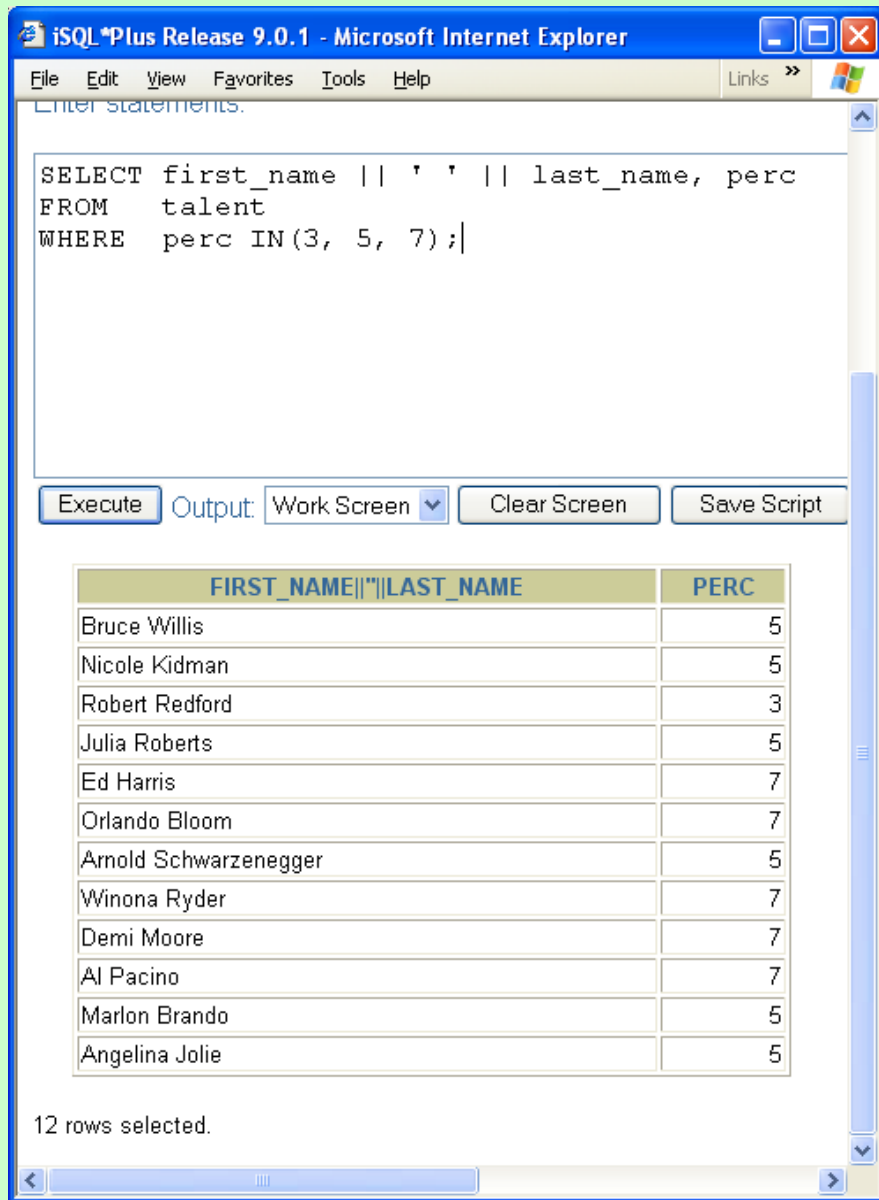
Module 15: Subqueries

Page B-4 IN Syntax

When building an IN predicate expression, parentheses are required to enclose the set of values, commas are used to separate items from one another.

For example:

WHERE perc IN (3, 5, 7)



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

```
SELECT first_name || ' ' || last_name, perc
FROM   talent
WHERE  perc IN(3, 5, 7);
```

Below the query area are buttons for "Execute", "Output: Work Screen" (with a dropdown arrow), "Clear Screen", and "Save Script".

The results are displayed in a table with two columns: "FIRST_NAME||' '||LAST_NAME" and "PERC". There are 12 rows of data.

FIRST_NAME ' ' LAST_NAME	PERC
Bruce Willis	5
Nicole Kidman	5
Robert Redford	3
Julia Roberts	5
Ed Harris	7
Orlando Bloom	7
Arnold Schwarzenegger	5
Winona Ryder	7
Demi Moore	7
Al Pacino	7
Marlon Brando	5
Angelina Jolie	5

At the bottom left, it says "12 rows selected." and there is a scrollbar at the bottom.

Module 15: Subqueries

Page B-5 IN w/String values

IN can also be used to compare a character value against a set of string values.

The screenshot shows the iSQL*Plus web interface in a Microsoft Internet Explorer browser window. The title bar reads "iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer". The browser's menu bar includes File, Edit, View, Favorites, Tools, and Help. The page header features the ORACLE logo, the iSQL*Plus text, and three circular icons with links for Password, Log Out, and Help. Below the header, there is a "Script Location:" text box with a "Browse..." button and a "Load Script" button. The main area is labeled "Enter statements:" and contains a text box with the following SQL query:

```
SELECT first_name || ' ' || last_name, home_country
FROM   talent
WHERE  home_country IN ('Ireland', 'UK');
```

Below the query text box, there are four buttons: "Execute", "Output:" (with a dropdown menu currently set to "Work Screen"), "Clear Screen", and "Save Script". At the bottom, the results of the query are displayed in a table:

FIRST_NAME ' ' LAST_NAME	HOME_COUNTRY
Ian McKellen	UK
Orlando Bloom	UK
Colin Farrell	Ireland

Module 15: Subqueries

Page B-6 IN w/Functions

And as you would expect, IN can also be used in combination with functions,

The screenshot shows the iSQL*Plus web interface in a Microsoft Internet Explorer browser window. The title bar reads "iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer". The menu bar includes File, Edit, View, Favorites, Tools, Help, and Links. The page header features the ORACLE logo, the iSQL*Plus text, and three circular icons with links for Password, Log Out, and Help. Below the header, there is a "Script Location:" text box with a "Browse..." button and a "Load Script" button. The main area is labeled "Enter statements:" and contains a text box with the following SQL query:

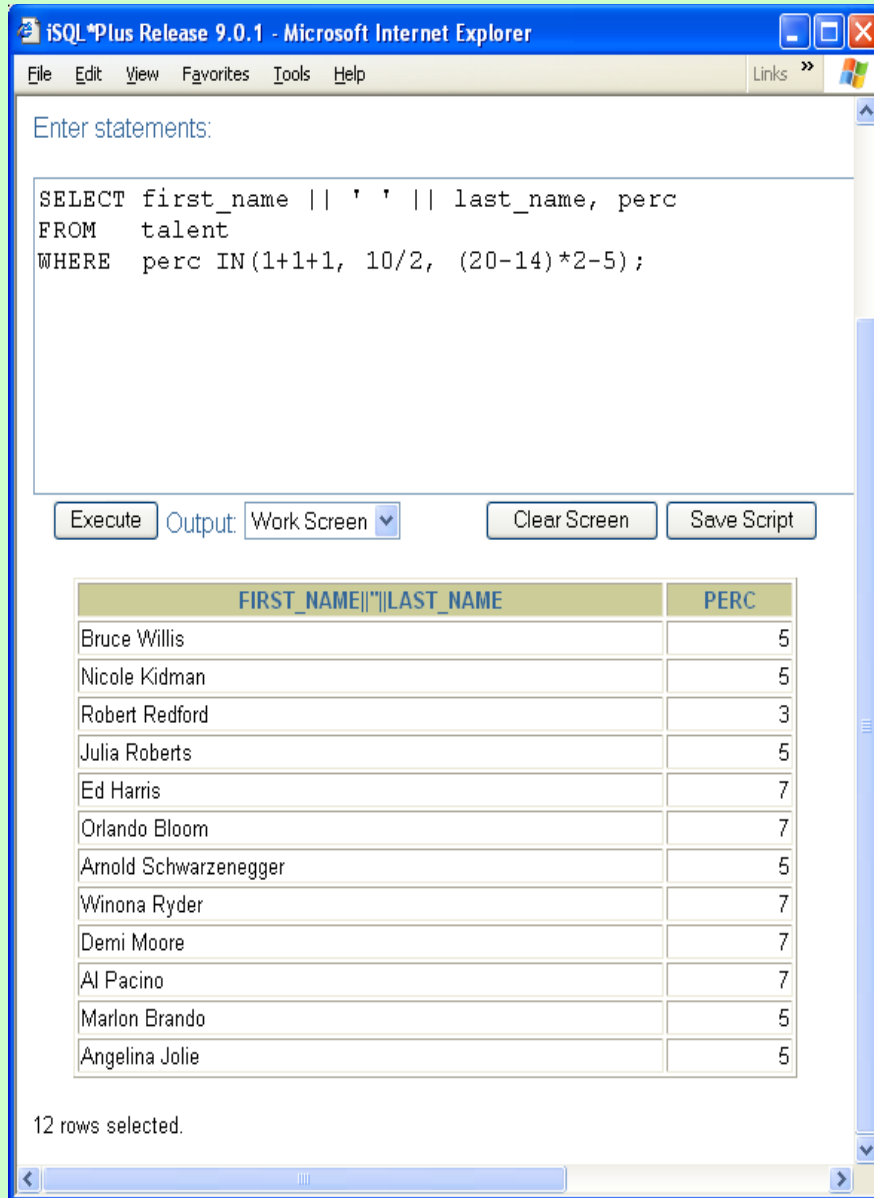
```
SELECT first_name || ' ' || last_name, home_country
FROM talent
WHERE UPPER(home_country) IN (UPPER('Ireland'), 'UK');
```

Below the text box are four buttons: "Execute", "Output:" (with a dropdown menu set to "Work Screen"), "Clear Screen", and "Save Script". At the bottom, a table displays the results of the query:

FIRST_NAME ' ' LAST_NAME	HOME_COUNTRY
Ian McKellen	UK
Orlando Bloom	UK
Colin Farrell	Ireland

Module 15: Subqueries

Page B-7 IN w/Expressions



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

```
Enter statements:  
  
SELECT first_name || ' ' || last_name, perc  
FROM   talent  
WHERE  perc IN (1+1+1, 10/2, (20-14)*2-5);
```

Below the query area are four buttons: "Execute", "Output: Work Screen" (with a dropdown arrow), "Clear Screen", and "Save Script".

The results are displayed in a table with two columns: "FIRST_NAME||' '||LAST_NAME" and "PERC". There are 12 rows of data.

FIRST_NAME ' ' LAST_NAME	PERC
Bruce Willis	5
Nicole Kidman	5
Robert Redford	3
Julia Roberts	5
Ed Harris	7
Orlando Bloom	7
Arnold Schwarzenegger	5
Winona Ryder	7
Demi Moore	7
Al Pacino	7
Marlon Brando	5
Angelina Jolie	5

At the bottom left, it says "12 rows selected." The status bar at the very bottom shows a scrollbar and navigation arrows.

...

Or pretty much anything else we've seen in a predicate expression, including the full set of operations (mathematical, alphanumeric, and datetime).

In my opinion, the IN operator is best used when a single predicate expression cannot isolate all of the rows that should qualify.

For example, consider a company with sales offices across the United States. They may need to query the database for information about their pacific rim offices.

If mailing address information is the only geographic information carried in the table, it would be pretty hard to come up with a simple predicate expression to identify the sales offices that qualify.

In this case it would be more practical to simply list all of the qualifying states. For example:

```
WHERE state_code IN ('WA', 'OR', 'CA')  
WHERE state_code IN ('WA', 'OR', 'CA', 'AK')
```

Let's move on just a bit now.

We've just seen how the IN operator can be used to compare a value against a set of values.

In the examples just past, the sets of values that we used were all literal values. They were hard-coded into our programs.

I wonder if there might be some way to dynamically generate the list of values, rather than having to hard-code it into our program?

Say, something like:

WHERE column IN (generated list)

Then we could write queries that answer questions such as: In the talent database, which clients are charged the highest percentage rate.

```
SELECT name
FROM   talent
WHERE  perc
      IN (find the highest percentage rate)
```

Do you have any thoughts as to how we might generate that list of value(s)?

Module 15: Subqueries

Page C-3 Subqueries (cont.)

If you're thinking that a SELECT statement would do the trick, you're right on track!

To answer the previous question, we would embed a SELECT query inside another SELECT query, as follows:

```
SELECT last_name, first_name, perc
FROM   talent
WHERE  perc IN
        (SELECT MAX(perc)
         FROM   talent);
```

This embedded query (SELECT MAX ...) is referred to as a subquery, or a nested query.

The screenshot shows the iSQL*Plus web interface in a Microsoft Internet Explorer browser window. The title bar reads "iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer". The menu bar includes File, Edit, View, Favorites, Tools, and Help. The main content area features the Oracle iSQL*Plus logo and navigation links for Password, Log Out, and Help. Below the logo, there is a "Script Location:" field with a "Browse..." button and a "Load Script" button. The "Enter statements:" section contains a text area with the following SQL query:

```
SELECT last_name, first_name, perc
FROM   talent
WHERE  perc IN
        (SELECT MAX(perc)
         FROM   talent);
```

Below the text area, there are buttons for "Execute", "Output:", a dropdown menu set to "Work Screen", "Clear Screen", and "Save Script". The "Execute" button has been clicked, and the results are displayed in a table below:

LAST_NAME	FIRST_NAME	PERC
Aniston	Jennifer	10
Wahlberg	Mark	10
Depp	Johnny	10

Module 15: Subqueries

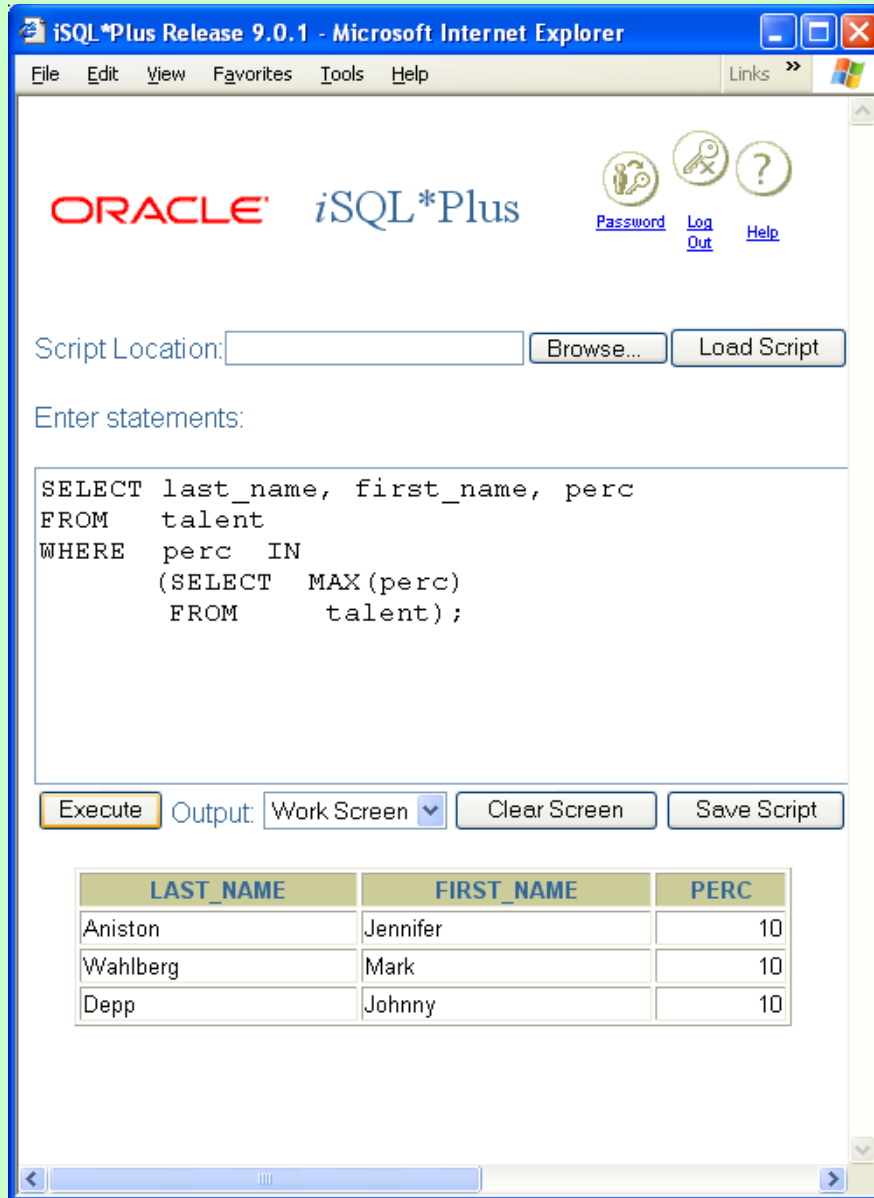
Page C-4 Usage Rules

Let's examine the structure of this SQL program.

1. The nested query is enclosed within a pair of parentheses, and SQL will evaluate this *inside* query, before it does the *outside* query.

This should appeal to your intuitions, especially in light of how we also use parentheses to change the precedence of operations in mathematical expressions.

2. The set of values that are returned by the subquery must be drawn from the same domain as the column that is being tested. That is, the column being tested in this example, perc, is a numeric column. The set of values that are generated by the nested subquery must all be numeric values.



ORACLE iSQL*Plus

Script Location:

Enter statements:

```
SELECT last_name, first_name, perc
FROM   talent
WHERE  perc IN
       (SELECT MAX(perc)
        FROM   talent);
```

Output:

LAST_NAME	FIRST_NAME	PERC
Aniston	Jennifer	10
Wahlberg	Mark	10
Depp	Johnny	10

Module 15: Subqueries

ORACLE iSQL*Plus

Script Location:

Enter statements:

```
SELECT last_name, first_name, perc
FROM   talent
WHERE  perc IN
       (SELECT MAX(perc)
        FROM   talent);
```

Output:

LAST_NAME	FIRST_NAME	PERC
Aniston	Jennifer	10
Wahlberg	Mark	10
Depp	Johnny	10

Page C-5 Usage Rules (cont)

3. The number of columns generated in the subquery must match the number of columns that are being compared. In this example, the subquery returns one column and the number of columns being tested (ie. perc) is one.
4. Depending on the comparison operator, the subquery may be limited to returning only a single row, otherwise the predicate expression would be ill-formed, and the program would 'crash'.
5. SQL is a closed system. What it takes as input is what it generates as output. (Base) tables are input, (result) tables are generated. Sets of data are input, sets of data are output. This means that the programmer can string together an arbitrary number of SQL statements, with the output of one being the input to another.

The talent agency would like a report showing all of our clients whose percentage rate is more than the average percentage rate.

One solution would be to write two SQL programs.

The first program would calculate the average percentage rate.

The second program would select the clients based on a WHERE condition that uses (ie. hard-codes) the value from the first program.

This solution appears on the following slide.

Module 15: Subqueries

iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer

File Edit View Favorites Tools Help Links »

ORACLE iSQL*Plus [Password](#) [Log Out](#) [Help](#)

Script Location: Browse... Load Script

Enter statements:

```
SELECT  AVG(perc)
FROM    talent;
```

Execute Output: Work Screen ▼ Clear Screen Save Script

AVG(PERC)
6.48

Page C-7 Problem 16-1 Solution

iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer

File Edit View Favorites Tools Help Links »

Script Location: Browse... Load Script

Enter statements:

```
SELECT  first_name || ' ' || last_name, perc
FROM    talent
WHERE   perc > (6.48);
```

Execute Output: Work Screen ▼ Clear Screen Save Script

FIRST_NAME ' ' LAST_NAME	PERC
Jennifer Aniston	10
Ed Harris	7
George Clooney	8
Ian McKellen	8
Orlando Bloom	7
Mark Wahlberg	10
Harrison Ford	8
Johnny Depp	10
Winona Ryder	7
Demi Moore	7
Al Pacino	7
Kevin Costner	8
Samuel L. Jackson	8

13 rows selected.

This solution works, but it does have its drawbacks.

The worst aspect of this solution, to me, is that it doesn't lend itself to 'automation'. It requires 'human intervention' every time it's used. Someone has to manually plug in the value from the first program into the second program.

Yikes!

I prefer to 'set it and forget it'. I'd like to write a program, test it, and then reuse it when I can. But with this approach, the only program that can be 'canned' is the first one. The second program needs to be rewritten AND tested each time.

So let's try rewriting this program to use the nested query capabilities of SQL.

As the two-part solution demonstrated, there are two logical parts to this program.

1. Calculating the average percentage rate
2. Using that average value to select the clients.

And as we saw in that solution, the program that calculates the average precedes the other.

Now then, how does that translate over to a nested solution?

Does the program that calculates the average belong in the 'inside' query, or the 'outside' query?

The screenshot shows the iSQL*Plus interface with the following SQL query entered in the 'Enter statements:' field:

```
SELECT last_name, perc
FROM talent
WHERE perc > (SELECT AVG(perc)
              FROM talent);
```

Below the query field are buttons for 'Execute', 'Output: Work Screen', 'Clear Screen', and 'Save Script'. The 'Output' section displays a table with 13 rows selected:

LAST_NAME	PERC
Aniston	10
Harris	7
Clooney	8
McKellen	8
Bloom	7
Wahlberg	10
Ford	8
Depp	10
Ryder	7
Moore	7
Pacino	7
Costner	8
Jackson	8

13 rows selected.

The general rule is this, the inside query is used to generate data that'll be used by the outside query.

Inside query:

```
SELECT AVG(perc)
FROM talent;
```

The outside query will use the set of values returned by the inside query:

```
SELECT last_name, perc
FROM talent
WHERE perc > (generated set of values)
```

And upon combining the two parts we get:

```
SELECT last_name, perc
FROM talent
WHERE perc > (SELECT AVG(perc)
              FROM talent);
```

Let's take a moment to recap what we've covered.

The IN operator specifies a list, or more precisely, a *set* of candidate values.

This set of values may be static (ie. Literal values) or dynamic (set of values returned by a SELECT query).

SQL is a closed system. What comes in, is what goes out. A SQL program inputs a table (set of values, or a relation) and outputs a table (set of values, or a relation).

This feature allows the programmer to string together an arbitrary number of SQL statements, with the output of one being the input to another, and so on, and so on, ...

Subqueries can pretty much be used anywhere in a SQL program, just as long as their use makes sense.

Subqueries can be used with any of the comparison operators, in either WHERE or HAVING clauses.

Module 15: Subqueries

Page C-13 Equality

Which of our clients are charged the lowest percentage rate.

```
SELECT first_name || ' ' || last_name, perc
FROM   talent
WHERE  perc =(SELECT  MIN(perc)
                  FROM    talent);
```

The screenshot shows the iSQL*Plus Release 9.0.1 interface in a Microsoft Internet Explorer browser. The page has a blue header with the Oracle logo and iSQL*Plus text. Below the header, there are links for Password, Log Out, and Help. A 'Script Location' field with a 'Browse...' button and a 'Load Script' button is present. The 'Enter statements:' section contains a text area with the following SQL query:

```
SELECT first_name || ' ' || last_name, perc
FROM   talent
WHERE  perc =(SELECT  MIN(perc)
                  FROM    talent);
```

Below the text area are buttons for 'Execute', 'Output: Work Screen' (with a dropdown arrow), 'Clear Screen', and 'Save Script'. At the bottom, a table displays the results of the query:

FIRST_NAME ' ' LAST_NAME	PERC
Robert Redford	3

Module 15: Subqueries

Page C-13 Inequality

Which of our clients are NOT charged the lowest percentage rate.

```
SET PAGESIZE 0;  
SELECT first_name || ' ' || last_name, perc  
FROM talent  
WHERE perc <> (SELECT MIN(perc)  
               FROM talent);
```

```
SELECT first_name || ' ' || last_name, perc  
FROM talent  
WHERE perc <> (SELECT MIN(perc)  
               FROM talent);
```

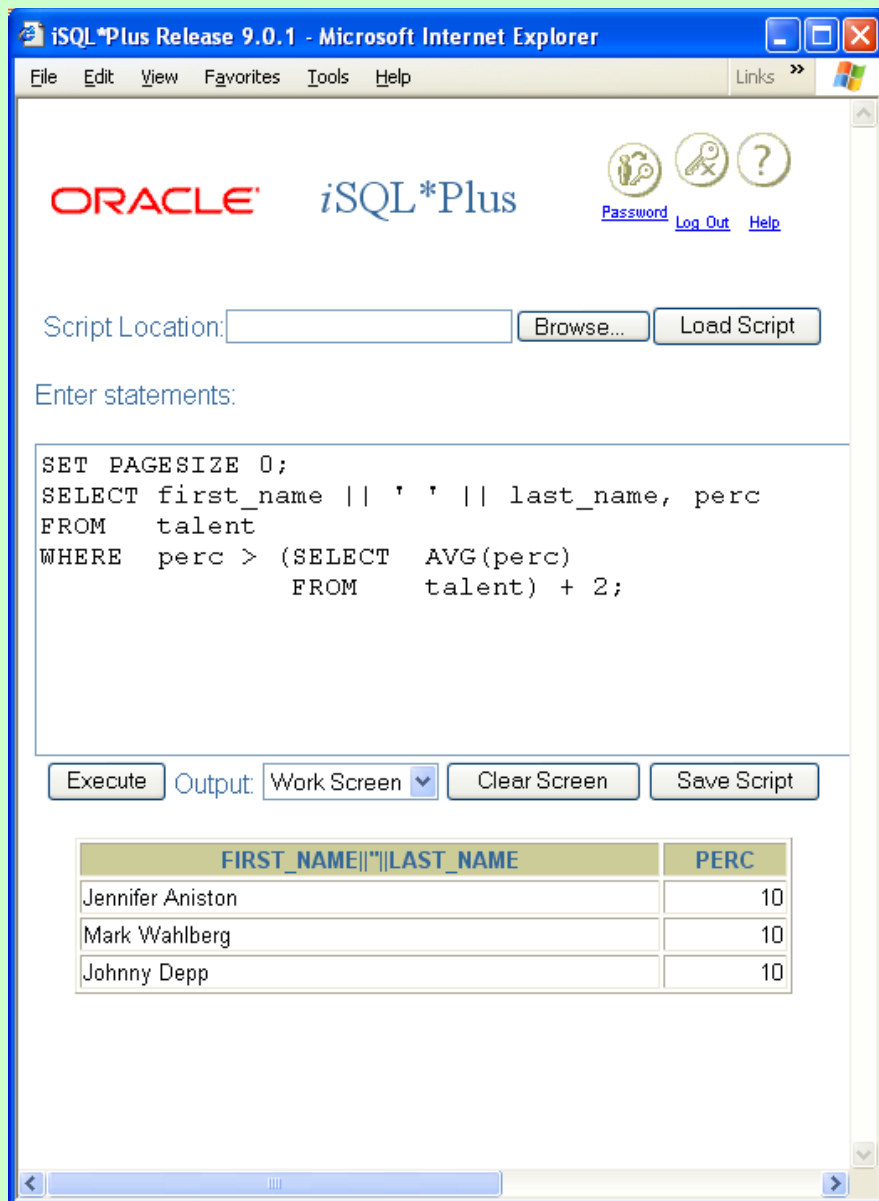
FIRST_NAME ' ' LAST_NAME	PERC
Bruce Willis	5
Tom Cruise	4
Nicole Kidman	5
Brad Pitt	6
Jennifer Aniston	10
Susan Sarandon	4
Julia Roberts	5
Ed Harris	7
George Clooney	8
Ian McKellen	8
Orlando Bloom	7
Mark Wahlberg	10
Harrison Ford	8
Johnny Depp	10
Arnold Schwarzenegger	5
Michelle Pfeiffer	4
Winona Ryder	7

Module 15: Subqueries

Page C-14 Greater than

Which of our clients are charged a percentage rate greater than the average percentage rate + 2?

```
SELECT first_name || ' ' || last_name, perc
FROM   talent
WHERE  perc > (SELECT  AVG(perc)
               FROM    talent) + 2;
```



ORACLE iSQL*Plus

Script Location:

Enter statements:

```
SET PAGESIZE 0;
SELECT first_name || ' ' || last_name, perc
FROM   talent
WHERE  perc > (SELECT  AVG(perc)
               FROM    talent) + 2;
```

Output:

FIRST_NAME '' LAST_NAME	PERC
Jennifer Aniston	10
Mark Wahlberg	10
Johnny Depp	10

Module 15: Subqueries

ORACLE iSQL*Plus

Script Location:

Enter statements:

```
SELECT first_name || ' ' || last_name, perc
FROM talent
WHERE perc NOT
      BETWEEN (SELECT AVG(perc) - STDDEV(perc) FROM talent)
      AND (SELECT AVG(perc) + STDDEV(perc) FROM talent)
```

Output:

FIRST_NAME '' LAST_NAME	PERC
Tom Cruise	4
Robert Redford	3
Jennifer Aniston	10
Susan Sarandon	4
Mark Wahlberg	10
Johnny Depp	10
Michelle Pfeiffer	4

7 rows selected.

Page C-15 Between

Which of our employees are charged a percentage rate that falls outside 1 standard deviation from the mean average percentage rate.

To be 'outside 1 standard deviation from the mean' can be rephrased as NOT within 1 standard deviation of the mean. Or,
NOT BETWEEN
(the average – 1 std deviation) and
(the average + 1 std deviation)

At this stage of the game, I don't expect you to understand what a standard deviation is, that'll come after you've taken a stats class.

But we have seen its use before (in the section on column functions), so I would expect that you'd be able to code this solution. As to testing the solution, if this were the real world, it would be perfectly acceptable for you to code the solution and then request help in testing the outcomes.

Module 15: Subqueries

Oracle iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer

File Edit View Favorites Tools Help Links »

ORACLE iSQL*Plus Password Log Out Help

Script Location: Browse... Load Script

Enter statements:

```
SELECT first_name || ' ' || last_name, perc
FROM talent
WHERE perc NOT
      BETWEEN (SELECT AVG(perc) - STDDEV(perc) FROM talent)
      AND (SELECT AVG(perc) + STDDEV(perc) FROM talent)
```

Execute Output: Work Screen Clear Screen Save Script

FIRST_NAME ' ' LAST_NAME	PERC
Tom Cruise	4
Robert Redford	3
Jennifer Aniston	10
Susan Sarandon	4
Mark Wahlberg	10
Johnny Depp	10
Michelle Pfeiffer	4

7 rows selected.

Page C-16 Digression: Productivity

Please take a moment here to step back and consider this code.

I hope you can appreciate the power and flexibility that SQL offers the programmer.

And in the 'real world' where you're often evaluated by your output or your productivity, SQL is an excellent tool to have in your skill set.

Module 15: Subqueries

ORACLE iSQL*Plus

Script Location:

Enter statements:

```
SELECT    gender, count(*)
FROM      talent
GROUP BY  gender
```

Output:

G	COUNT(*)
F	8
M	17

Page D-1 Fun and Games

Based on our client database, of which gender do we have the most clients? Male or female?

We could write a SQL program that would give us a breakdown by gender.

```
SELECT    gender, count(*)
FROM      talent
GROUP BY  gender
```

But this doesn't really answer the question does it?

The question was phrased in such a fashion that the expected answer should be either Male or Female (or perhaps M or F).

Module 15: Subqueries

Page D-2 Fun and Games (cont)

How about this?

If we know how many clients of each gender we have, perhaps we could write a query that compares those counts against the total number of clients?

And since there are only two genders, whichever gender is represented by more than half of our client base is the answer.

```
SELECT    gender, count(*)
FROM      talent
GROUP BY  gender
HAVING    COUNT(*) >
           (SELECT COUNT(gender)/2
            FROM   talent);
```

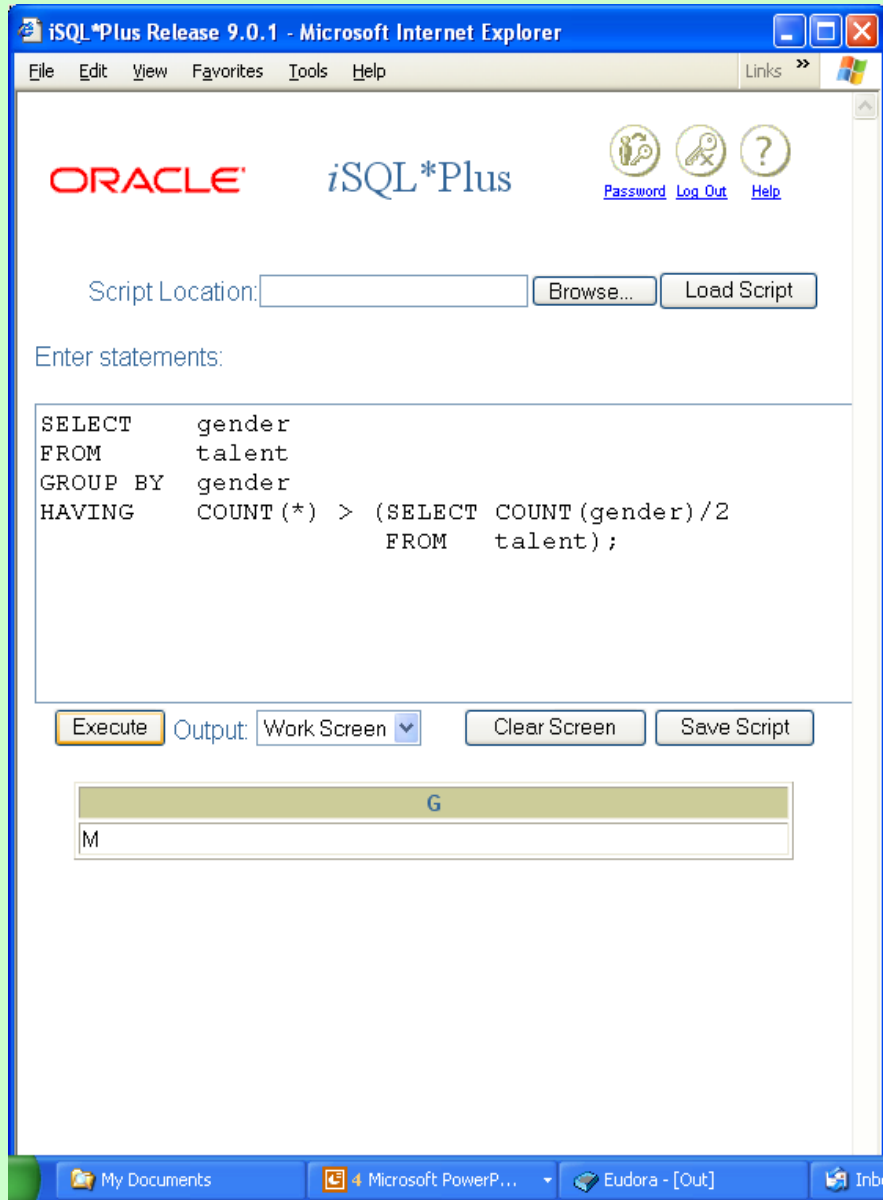
Script Location:

Enter statements:

```
SELECT    gender, count(*)
FROM      talent
GROUP BY  gender
HAVING    COUNT(*) > (SELECT COUNT(gender)/2
                        FROM   talent);
```

G	COUNT(*)
M	17

Module 15: Subqueries



Page D-3 Fun and Games (cont)

And after we're satisfied with the results, we can remove the tally column from the result table,

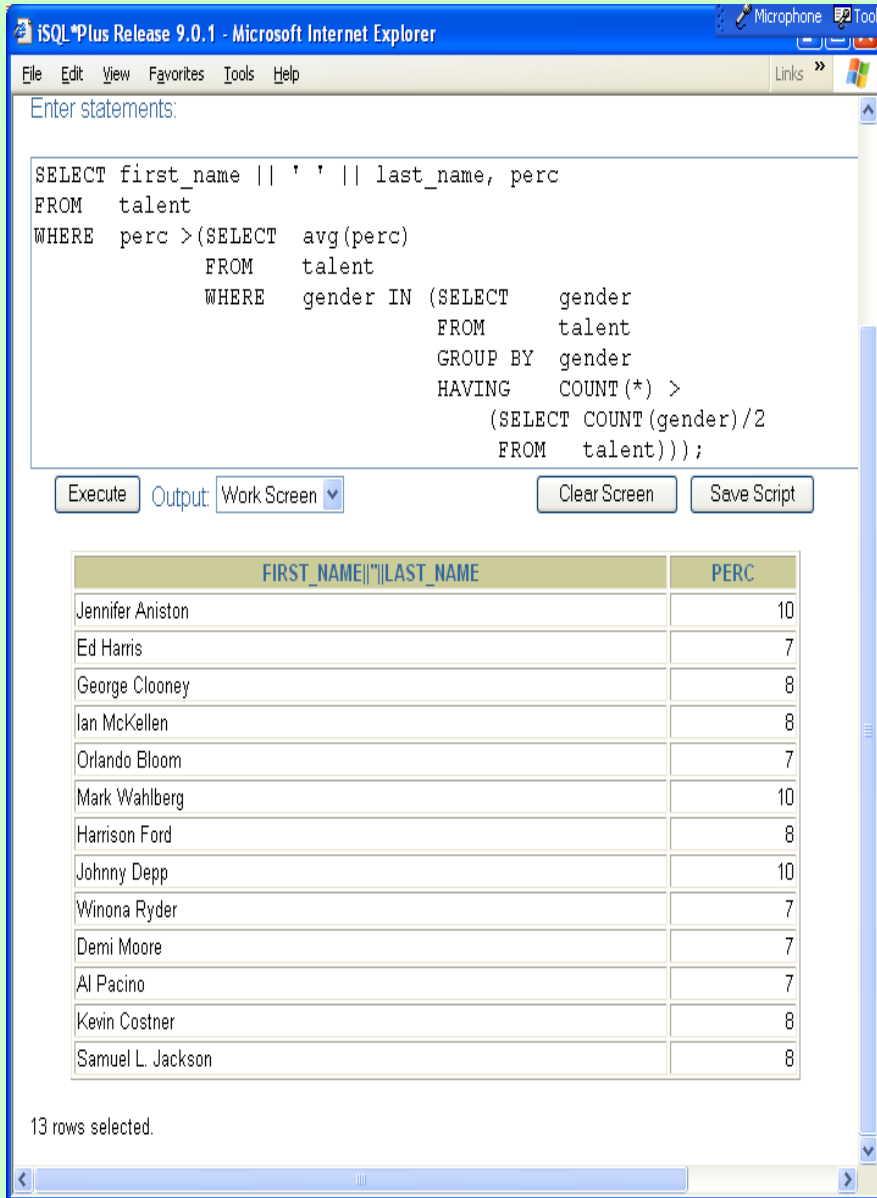
```
SELECT    gender
FROM      talent
GROUP BY  gender
HAVING    COUNT(*) >
          (SELECT COUNT(gender)/2
           FROM    talent);
```

And now we have the 'correct' answer to the question that was posed.

Don't freak out. Yes, we did kind of take a giant leap, but the more you use SQL, the more readily these solutions become apparent.

And, oh by the way, did you notice that we placed the subquery inside the HAVING clause?

Module 15: Subqueries



The screenshot shows the iSQL*Plus interface. The SQL statement entered is:

```
SELECT first_name || ' ' || last_name, perc
FROM talent
WHERE perc > (SELECT avg(perc)
              FROM talent
              WHERE gender IN (SELECT gender
                              FROM talent
                              GROUP BY gender
                              HAVING COUNT(*) >
                                   (SELECT COUNT(gender)/2
                                    FROM talent)));
```

Below the query, there are buttons for "Execute", "Output", "Work Screen", "Clear Screen", and "Save Script". The "Output" button is selected, and the results are displayed in a table:

FIRST_NAME ' ' LAST_NAME	PERC
Jennifer Aniston	10
Ed Harris	7
George Clooney	8
Ian McKellen	8
Orlando Bloom	7
Mark Wahlberg	10
Harrison Ford	8
Johnny Depp	10
Winona Ryder	7
Demi Moore	7
Al Pacino	7
Kevin Costner	8
Samuel L. Jackson	8

At the bottom, it says "13 rows selected."

Page D-4 Problem 16-2

Now I did ask you to not freak out.

So tell me, what does this SQL program do?



I'd start by looking for SELECT statements.

I see four SELECT keywords, so I'll presume that there are three (ie. $4 - 1 = 3$) nested queries.

Start from the inside and work your way out. Along the way you might want to try running 'snippets' of this code in SQL to see what happens.

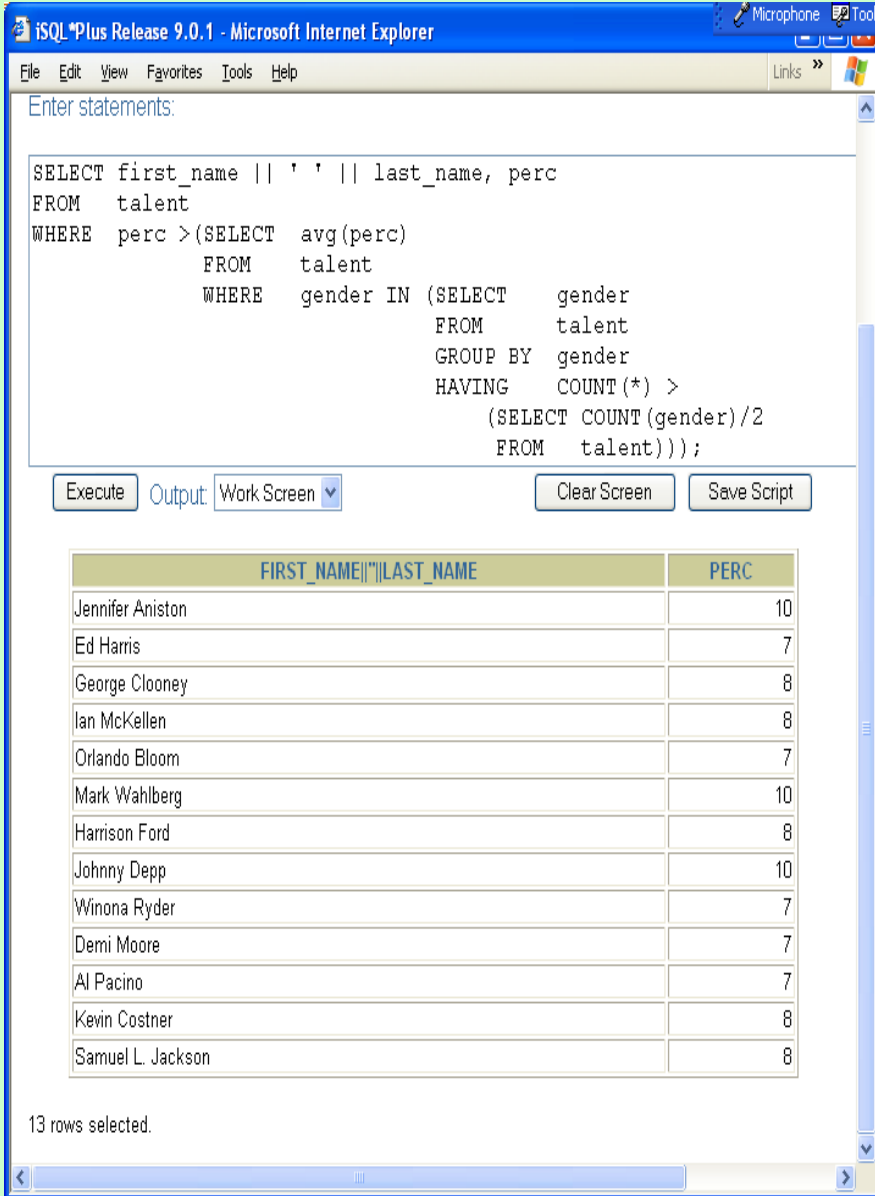
Module 15: Subqueries

Page D-5 Problem 16-2 Stage 1

The innermost nested query simply tells us what half of our client tally is. If we have 100 clients on file, half of that value is 50.

```
SELECT COUNT(gender) / 2
FROM talent;
```

This value will be used in the next SELECT statement to identify which gender is represented by the majority of our clients (which gender is represented by more than HALF of our clients).



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

```
SELECT first_name || ' ' || last_name, perc
FROM talent
WHERE perc > (SELECT avg(perc)
              FROM talent
              WHERE gender IN (SELECT gender
                              FROM talent
                              GROUP BY gender
                              HAVING COUNT(*) >
                                   (SELECT COUNT(gender) / 2
                                    FROM talent)));
```

Below the text 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: "FIRST_NAME||' '||LAST_NAME" and "PERC". There are 13 rows of data:

FIRST_NAME ' ' LAST_NAME	PERC
Jennifer Aniston	10
Ed Harris	7
George Clooney	8
Ian McKellen	8
Orlando Bloom	7
Mark Wahlberg	10
Harrison Ford	8
Johnny Depp	10
Winona Ryder	7
Demi Moore	7
Al Pacino	7
Kevin Costner	8
Samuel L. Jackson	8

At the bottom left, it says "13 rows selected."

Module 15: Subqueries

Page D-6 Problem 16-2 Stage 2

This query returns a single column, ie. the gender column, for the gender that has over half the client base.

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

```
SELECT first_name || ' ' || last_name, perc
FROM talent
WHERE perc > (SELECT avg(perc)
              FROM talent
              WHERE gender IN (SELECT gender
                              FROM talent
                              GROUP BY gender
                              HAVING COUNT(*) >
                                   (SELECT COUNT(gender)/2
                                    FROM talent)));
```

Below the text area are buttons for "Execute", "Output", "Work Screen" (with a dropdown arrow), "Clear Screen", and "Save Script".

The results are displayed in a table with two columns: "FIRST_NAME||' '||LAST_NAME" and "PERC". There are 13 rows of data, representing actors and their percentage of the client base.

FIRST_NAME ' ' LAST_NAME	PERC
Jennifer Aniston	10
Ed Harris	7
George Clooney	8
Ian McKellen	8
Orlando Bloom	7
Mark Wahlberg	10
Harrison Ford	8
Johnny Depp	10
Winona Ryder	7
Demi Moore	7
Al Pacino	7
Kevin Costner	8
Samuel L. Jackson	8

At the bottom left, it says "13 rows selected."

```
SELECT      gender
FROM        talent
GROUP BY    gender
HAVING      COUNT(*) >
              (SELECT COUNT(gender)/2
               FROM    talent);
```

Module 15: Subqueries

Page D-7 Problem 16-2 Stage 3

What is the average percentage rate for the highest represented gender?

iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer

File Edit View Favorites Tools Help Links

Enter statements:

```
SELECT first_name || ' ' || last_name, perc
FROM talent
WHERE perc > (SELECT avg(perc)
              FROM talent
              WHERE gender IN (SELECT gender
                              FROM talent
                              GROUP BY gender
                              HAVING COUNT(*) >
                                   (SELECT COUNT(gender)/2
                                    FROM talent)));
```

Execute Output Work Screen Clear Screen Save Script

FIRST_NAME ' ' LAST_NAME	PERC
Jennifer Aniston	10
Ed Harris	7
George Clooney	8
Ian McKellen	8
Orlando Bloom	7
Mark Wahlberg	10
Harrison Ford	8
Johnny Depp	10
Winona Ryder	7
Demi Moore	7
Al Pacino	7
Kevin Costner	8
Samuel L. Jackson	8

13 rows selected.

```
SELECT avg(perc)
FROM talent
WHERE gender IN
      (SELECT gender
       FROM talent
       GROUP BY gender
       HAVING COUNT(*) >
            (SELECT COUNT(gender)/2
             FROM talent))
```

Module 15: Subqueries

Page D-8 Problem 16-2 Stage 4

Which of our clients pays a higher percentage rate

than the average rate

that is charged to the majority gender?

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

```
SELECT first_name || ' ' || last_name, perc
FROM talent
WHERE perc > (SELECT avg(perc)
              FROM talent
              WHERE gender IN (SELECT gender
                              FROM talent
                              GROUP BY gender
                              HAVING COUNT(*) >
                                   (SELECT COUNT(gender)/2
                                    FROM talent)));
```

Below the text 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: "FIRST_NAME||' '||LAST_NAME" and "PERC". There are 13 rows of data, all of which have a percentage rate greater than the average rate of 7.5.

FIRST_NAME ' ' LAST_NAME	PERC
Jennifer Aniston	10
Ed Harris	7
George Clooney	8
Ian McKellen	8
Orlando Bloom	7
Mark Wahlberg	10
Harrison Ford	8
Johnny Depp	10
Winona Ryder	7
Demi Moore	7
Al Pacino	7
Kevin Costner	8
Samuel L. Jackson	8

At the bottom left, it says "13 rows selected."

In the examples that I used to demonstrate subqueries, I was careful to ensure that the result tables of each of the nested queries returned a single value.

The following question is clear and unambiguous:

IS major_code = 15011

The predicate becomes a little confusing though when the condition has multiple values:

IS major_code = 15011, 08011

What does this mean? Must the major_code be BOTH of these values, to be evaluated as TRUE, or will it evaluate TRUE if the major_code is equal to either value?

Module 15: Subqueries

iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer

File Edit View Favorites Tools Help Links

ORACLE iSQL*Plus Password Log Out Help

Script Location: Browse... Load Script

Enter statements:

```
SELECT first_name || ' ' || last_name, perc
FROM talent
WHERE perc >
      (SELECT perc
       FROM talent
       WHERE perc > 7)
```

Execute Output Work Screen Clear Screen Save Script

(SELECT perc
*
ERROR at line 4:
ORA-01427: single-row subquery returns more than one row

Page E-2 Atomic Values (cont.)

Or consider one of the other non-equality comparison operators.

This is a well-formed, unambiguous predicate:

`perc > 15011`

But what does this mean:

`perc > 8, 9`

Again, should the condition be evaluated TRUE if perc is greater than either of the values, or must perc be greater than BOTH of the values for this predicate to be TRUE?

In point of fact, SQL will not even attempt to answer this question, and in the Oracle environment it return the cryptic error message: *'single-row subquery returns more than one row'*.

Module 15: Subqueries

iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer

File Edit View Favorites Tools Help Links »

ORACLE iSQL*Plus

[Password](#) [Log Out](#) [Help](#)

Script Location:

Enter statements:

```
SELECT first_name || ' ' || last_name, perc
FROM talent
WHERE perc >
      (SELECT perc
       FROM talent
       WHERE perc > 7)
```

(SELECT perc
*
ERROR at line 4:
ORA-01427: single-row subquery returns more than one row

Page E-3 Atomic Values (cont.)

The problem is that SQL can only compare single values with single values.

Or to state this issue more precisely, SQL can only compare *atomic* values with other atomic values.

An atomic value is a value that cannot be further broken down, or that contains any other parts.

In this example, the left-hand side of the predicate, *perc*, is an atomic value. But the right-hand side of the predicate, the subquery, is not an atomic value because it might return many rows.

Module 15: Subqueries

Page F-1 Quantifier Predicates

To resolve the issue, or at least provide a suitable workaround, SQL offers the programmer three quantifier predicates.

ALL
ANY
SOME

The screenshot shows the iSQL*Plus Release 9.0.1 interface in a Microsoft Internet Explorer browser. The page has a blue header with the Oracle logo and iSQL*Plus text. Below the header, there are links for Password, Log Out, and Help. A 'Script Location' field with 'Browse...' and 'Load Script' buttons is present. The main area is labeled 'Enter statements:' and contains a text box with the following SQL query:

```
SELECT first_name || ' ' || last_name, perc
FROM talent
WHERE perc >
      (SELECT perc
       FROM talent
       WHERE perc > 7)
```

Below the text box are buttons for 'Execute', 'Output', 'Work Screen', 'Clear Screen', and 'Save Script'. The 'Execute' button is highlighted. Below the buttons, the output area shows the start of a query result: '(SELECT perc *'. Below this, an error message is displayed: 'ERROR at line 4: ORA-01427: single-row subquery returns more than one row'.

Module 15: Subqueries

Page F-2 Odd numbers

The following query returns rows from the talent table where the percentage rate is odd.

```
SELECT perc
FROM talent
WHERE MOD(perc,2) <> 0;
```

The screenshot shows the iSQL*Plus Release 9.0.1 interface within a Microsoft Internet Explorer browser. The title bar reads "iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer". The menu bar includes "File", "Edit", "View", "Favorites", "Tools", "Help", and "Links". The main text area contains the SQL query:

```
SELECT perc
FROM talent
WHERE MOD(perc,2) <> 0;
```

 Below the query area are buttons for "Execute", "Output", "Work Screen" (with a dropdown arrow), "Clear Screen", and "Save Script". The results are displayed in a table with a single column labeled "PERC". The table contains 12 rows of data. At the bottom left, it states "12 rows selected.".

PERC
5
5
3
5
7
7
5
7
7
7
5
5

Module 15: Subqueries

Page F-3 ANY

The ANY quantifier predicate can be used to qualify rows where the compared value is less than ANY of the values in the set of values.

In this example the subquery returns all of the odd percentage rates in the table (3, 5, 7).

The perc value in each row in the base table is compared against this set of values, and if it's less than any of the values in the set, that row is qualified for inclusion in the result table.

You can think of the ANY operator as being logically equivalent to a series of predicates joined by the OR operator.

The screenshot shows the iSQL*Plus interface in a Microsoft Internet Explorer browser window. The title bar reads "iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer". The menu bar includes "File", "Edit", "View", "Favorites", "Tools", "Help", and "Links". The main area is labeled "Enter statements:" and contains the following SQL query:

```
SELECT first_name || ' ' || last_name, perc
FROM talent
WHERE perc < ANY(SELECT perc
                  FROM talent
                  WHERE MOD(perc,2) <> 0);
```

Below the query area are buttons for "Execute", "Output", "Work Screen" (with a dropdown arrow), "Clear Screen", and "Save Script". The "Output" button is selected, and the results are displayed in a table with two columns: "FIRST_NAME||' '||LAST_NAME" and "PERC".

FIRST_NAME ' ' LAST_NAME	PERC
Bruce Willis	5
Tom Cruise	4
Nicole Kidman	5
Robert Redford	3
Brad Pitt	6
Susan Sarandon	4
Julia Roberts	5
Arnold Schwarzenegger	5
Michelle Pfeiffer	4
Marlon Brando	5
Colin Farrell	6
Angelina Jolie	5

At the bottom left, it says "12 rows selected."

Module 15: Subqueries

Page F-4 ALL

The screenshot shows the iSQL*Plus Release 9.0.1 interface within a Microsoft Internet Explorer browser. The interface includes a menu bar (File, Edit, View, Favorites, Tools, Help), a toolbar with icons for home, back, forward, and search, and a status bar at the bottom. The main content area is divided into several sections:

- Script Location:** A text input field followed by "Browse..." and "Load Script" buttons.
- Enter statements:** A large text area containing the following SQL query:

```
SELECT first_name || ' ' || last_name, perc
FROM talent
WHERE perc > ALL(SELECT perc
                  FROM talent
                  WHERE MOD(perc,2) <> 0);
```
- Execution Controls:** "Execute" button, "Output:" dropdown menu (set to "Work Screen"), "Clear Screen" button, and "Save Script" button.
- Results Table:** A table with two columns: "FIRST_NAME||' '||LAST_NAME" and "PERC". It contains 8 rows of data.
- Status:** "8 rows selected."

FIRST_NAME ' ' LAST_NAME	PERC
Jennifer Aniston	10
George Clooney	8
Ian McKellen	8
Mark Wahlberg	10
Harrison Ford	8
Johnny Depp	10
Kevin Costner	8
Samuel L. Jackson	8

The ALL quantifier predicate can be similarly used to ensure that the compared value is greater than EACH and EVERY one of the values in the set of values.

In this example the subquery returns all of the odd percentage rates in the table (3, 5, 7).

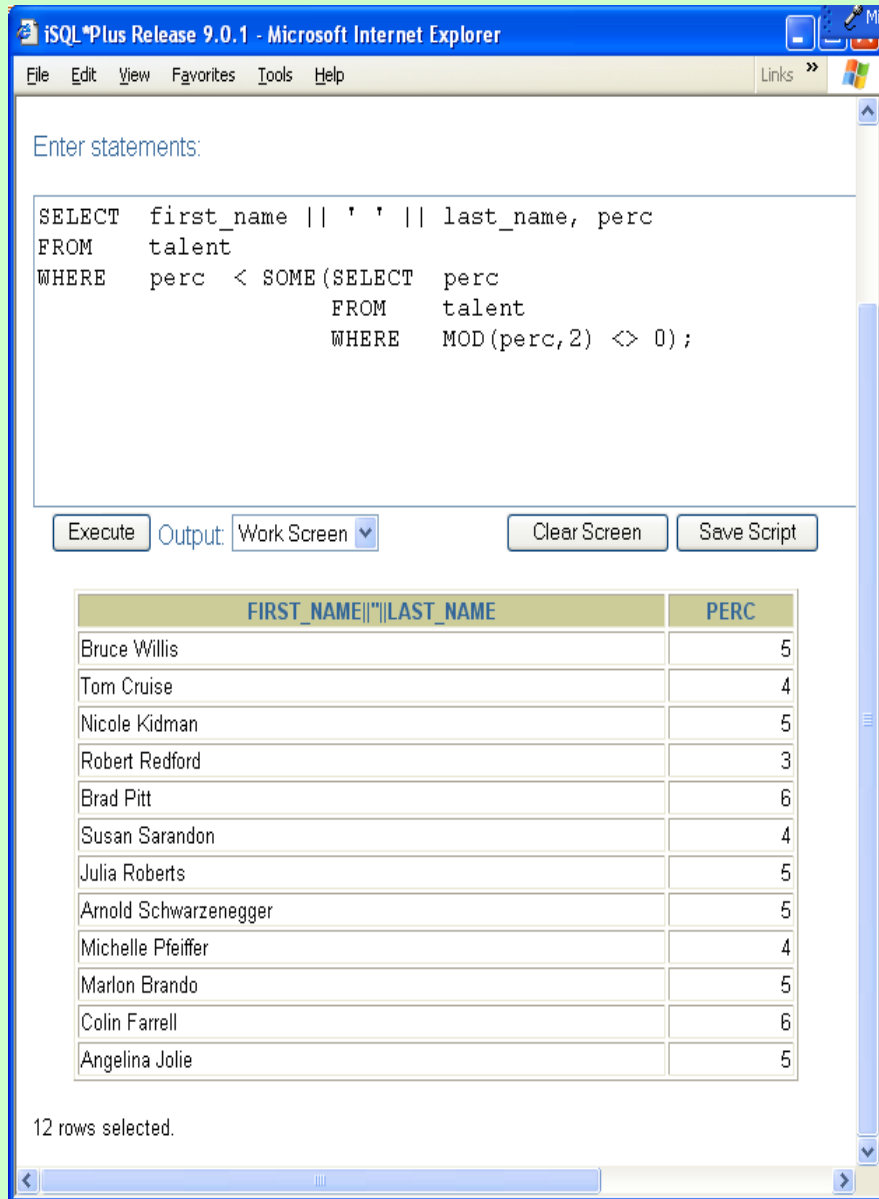
The perc value in each row in the base table is compared against this set of values, and if it's greater than each of the values in the set, that row is qualified for inclusion in the result table.

You can think of the ALL operator as being logically equivalent to a series of predicates joined by the AND operator.

Module 15: Subqueries

Page F-5 SOME

SOME and ANY produce the same results, and which operator you choose depends only on which term 'sounds' better than the other.



The screenshot shows the iSQL*Plus Release 9.0.1 interface within a Microsoft Internet Explorer browser. The main window contains a text area for entering SQL statements. Below the text area are buttons for 'Execute', 'Output', 'Work Screen', 'Clear Screen', and 'Save Script'. The 'Output' button is selected, and the results of the query are displayed in a table below. The table has two columns: 'FIRST_NAME||''||LAST_NAME' and 'PERC'. The results show 12 rows of data, including names like Bruce Willis, Tom Cruise, Nicole Kidman, Robert Redford, Brad Pitt, Susan Sarandon, Julia Roberts, Arnold Schwarzenegger, Michelle Pfeiffer, Marlon Brando, Colin Farrell, and Angelina Jolie, each with a corresponding 'PERC' value.

```
Enter statements:
```

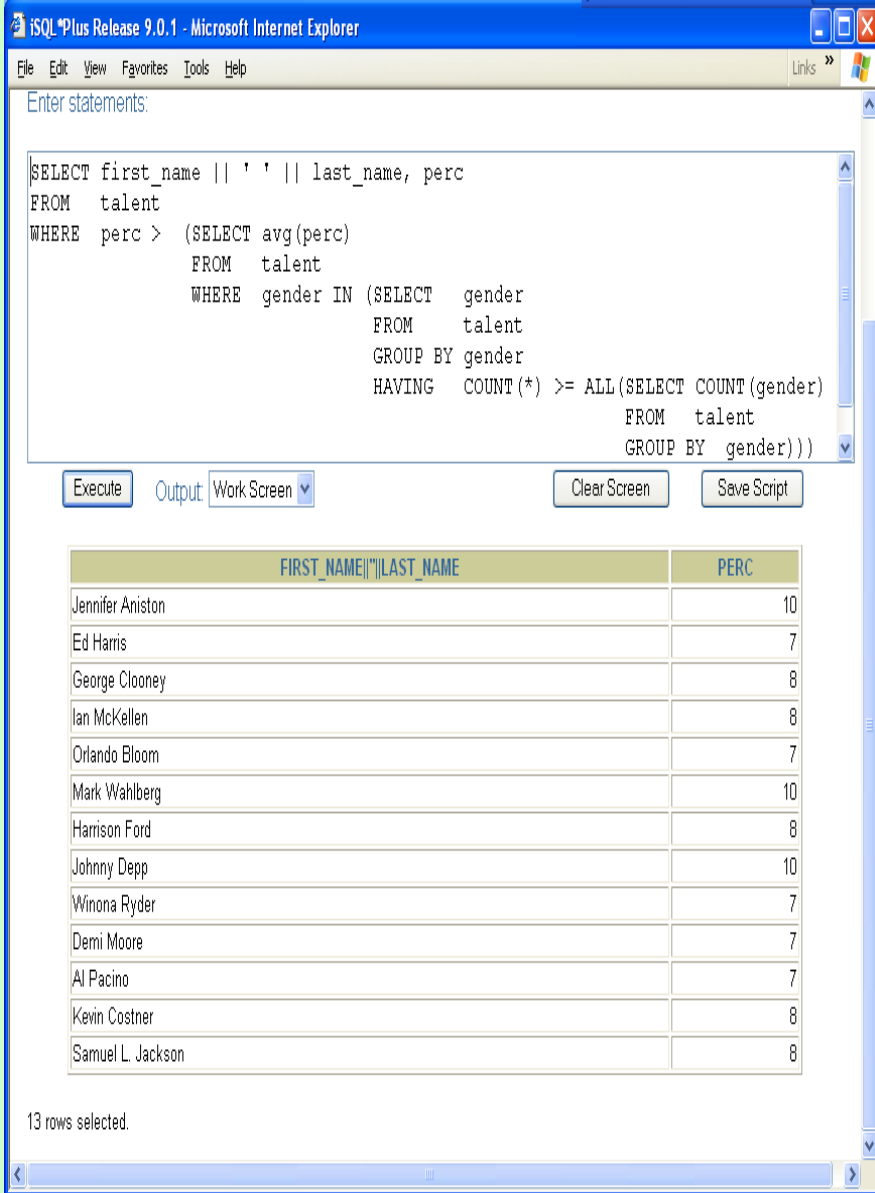
```
SELECT first_name || ' ' || last_name, perc
FROM talent
WHERE perc < SOME (SELECT perc
                    FROM talent
                    WHERE MOD(perc,2) <> 0);
```

Execute Output Work Screen Clear Screen Save Script

FIRST_NAME '' LAST_NAME	PERC
Bruce Willis	5
Tom Cruise	4
Nicole Kidman	5
Robert Redford	3
Brad Pitt	6
Susan Sarandon	4
Julia Roberts	5
Arnold Schwarzenegger	5
Michelle Pfeiffer	4
Marlon Brando	5
Colin Farrell	6
Angelina Jolie	5

12 rows selected.

Module 15: Subqueries



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

```
SELECT first_name || ' ' || last_name, perc
FROM talent
WHERE perc > (SELECT avg(perc)
              FROM talent
              WHERE gender IN (SELECT gender
                              FROM talent
                              GROUP BY gender
                              HAVING COUNT(*) >= ALL(SELECT COUNT(gender)
                                                    FROM talent
                                                    GROUP BY gender)))
```

Below the query area are buttons for "Execute", "Output", "Work Screen", "Clear Screen", and "Save Script". The "Output" button is selected, and the results are displayed in a table with two columns: "FIRST_NAME||' '||LAST_NAME" and "PERC".

FIRST_NAME ' ' LAST_NAME	PERC
Jennifer Aniston	10
Ed Harris	7
George Clooney	8
Ian McKellen	8
Orlando Bloom	7
Mark Wahlberg	10
Harrison Ford	8
Johnny Depp	10
Winona Ryder	7
Demi Moore	7
Al Pacino	7
Kevin Costner	8
Samuel L. Jackson	8

At the bottom left, it says "13 rows selected."

Page F-6 Fun and Games Revisited

Now that we have some mechanism that'll account for not atomic comparisons, we can rewrite the fun-and-games problem, so that it's a little more effective.

This revised code eliminates our reliance on a presumption of only two gender codes, as well as deals effectively with scenarios where the gender breakdown is equally diverse.

Here's a discussion topic for the Bb forum:

What is the impact of a subquery that includes NULL values on an ANY-quantified predicate expression?

What is the impact of a subquery that includes NULL values on an ALL-quantified predicate expression?

IN

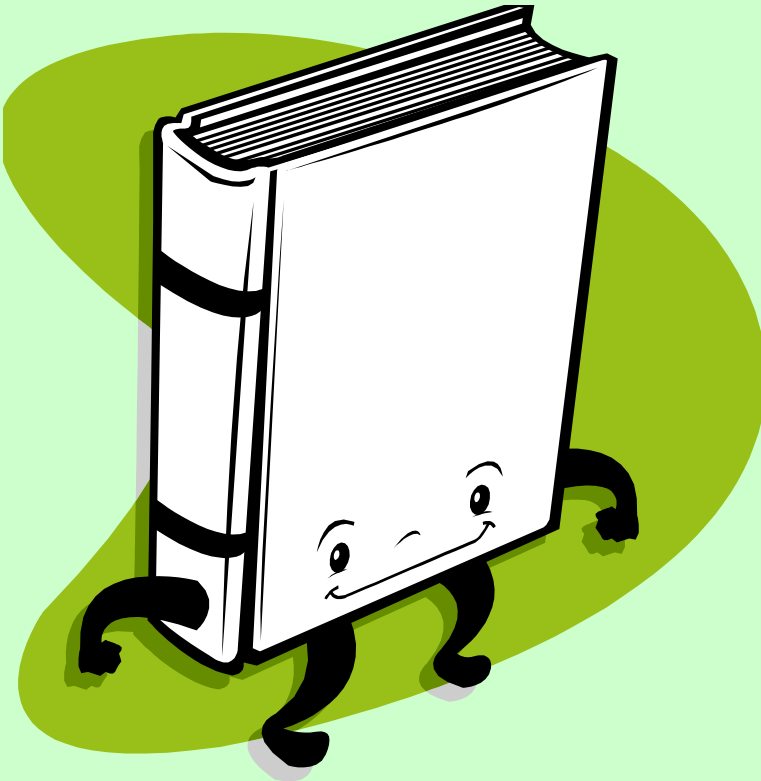
Subquery, nested query, embedded query
Inside query, outside query

Closed system

Usage rules

Atomic values

Quantifier / quantified predicates
ALL, ANY, SOME



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