SQL Programming

Numeric, Character, and Date Operations

Page A-1: Intro

All programming languages provide some capabilities for manipulating numeric and character values.

In the case of numeric values, the four basic mathematical operations of addition, subtraction, multiplication, and division may be applied.

In the case of character values, usually some form of concatenation is provided.

In this unit we'll examine how and where we can perform these rudimentary operations in SQL.

Don't worry about the term *concatenation,* I'll define that in a few slides, until then you'll have to manage your anticipation.

Page B-1 Numeric Operations

The basic numeric operations are the simple arithmetic operations of:

addition,
subtraction,
multiplication, and
division.

Now, take a guess. Which SQL statement do we use for math?

SELECT
FROM
WHERE

Can we include these arithmetic operations in the SELECT clause, or do they belong in the FROM clause, or perhaps the WHERE clause?

Or are you holding our for a new clause entirely?

Page B-2 Numeric Operations (cont)

The answer is: numeric operations can go anywhere (almost) except for the FROM clause.

Well now, if that’s the case, then when would you do math in a SELECT clause, as opposed to, say, in the WHERE clause?

Page B-3 Numeric Operations (cont)

That's right , math happens in the SELECT clause when you want to see the 'answer' in the result table.

Your intuitions are coming along quite nicely. You know that the SELECT clause controls what information is displayed on the screen in the result table. So you are absolutely right in thinking that the SELECT statement is the place to do math, if you want to see the answer on the screen (ie. In the result table)

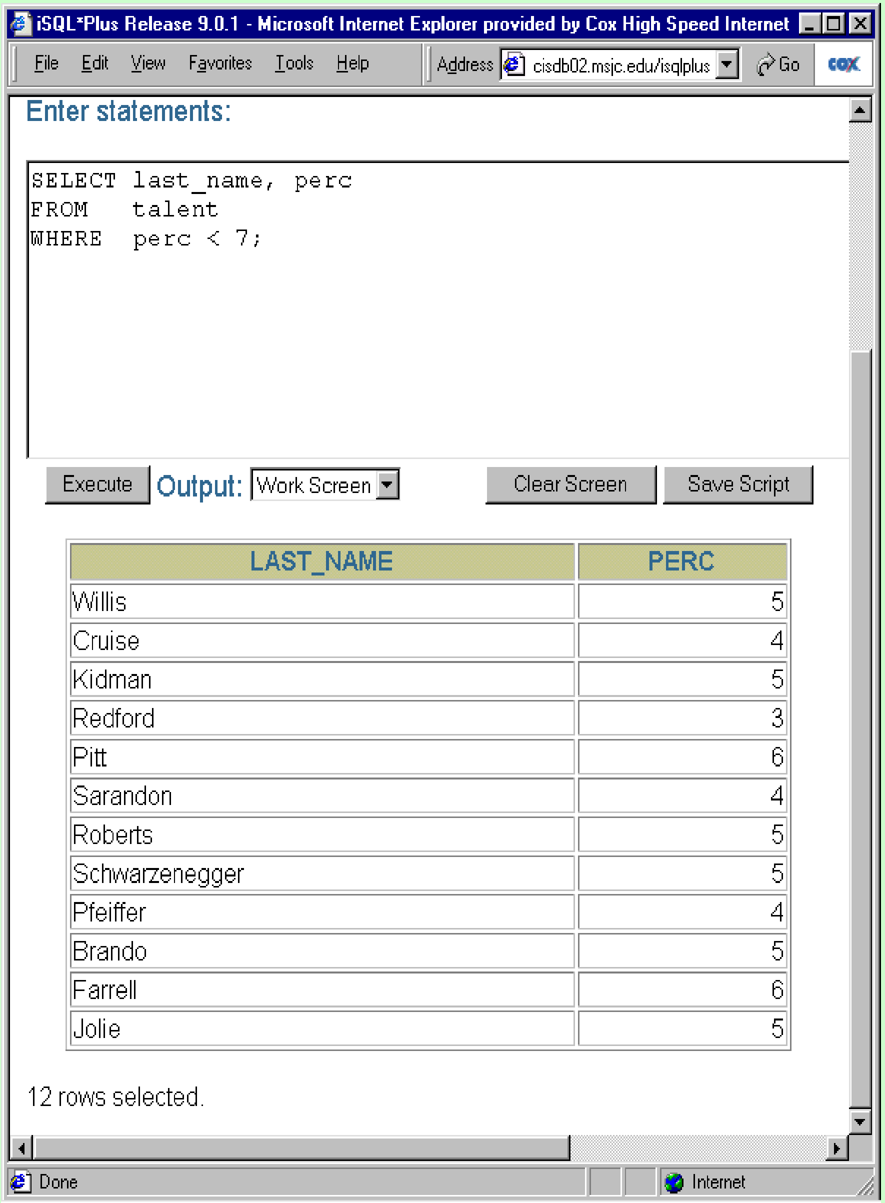
Good job.

If you didn't anticipate that answer, that's okay. My guess is that you were reading along and were in a hurry to get thru the slides. Take your time.

Cogitate.

You're not in a race, you're learning. How do you best learn?

Page B-4 Operations - Example



```
SELECT last_name, perc
FROM   talent
WHERE  perc < 7;
```

LAST_NAME	PERC
Willis	5
Cruise	4
Kidman	5
Redford	3
Pitt	6
Sarandon	4
Roberts	5
Schwarzenegger	5
Pfeiffer	4
Brando	5
Farrell	6
Jolie	5

12 rows selected.

For these math examples, let’s play around with the PERC column in the TALENT table.

We’ll only use a few of the rows (WHERE perc < 7), so that they all fit on the slide show.

Page B-5: Problem 6-1 Statement

Our user community needs a report ...

Assuming that we can get million dollar contracts for each of our clients, how much money would we be earning for each star?

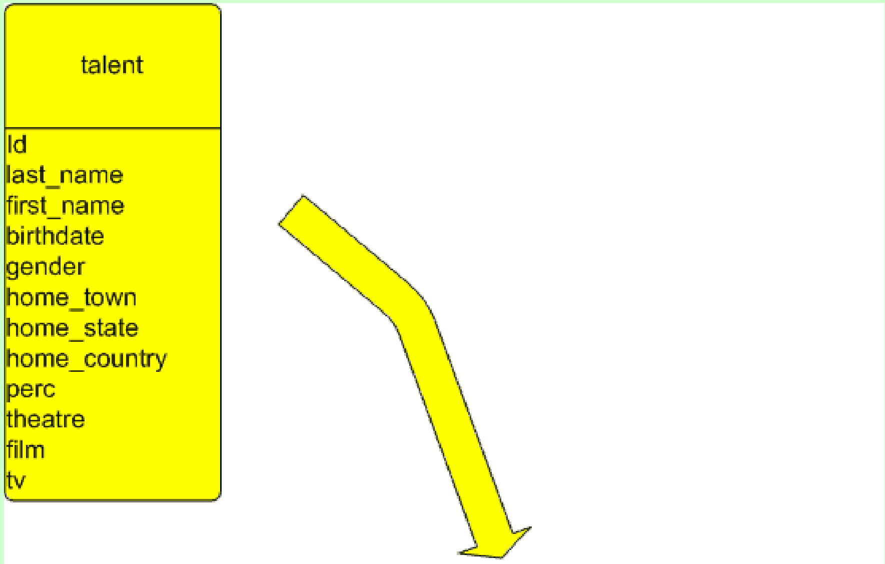
Rephrase: Earnings are calculated as the percentage rate * contract value

Page B-6: Problem 6-1 Design and Code

Step 1: Build the Table Build Chart (TBC)

Step 2: Double check your TBC solution

Step 3: Transform the TBC into code.



talent

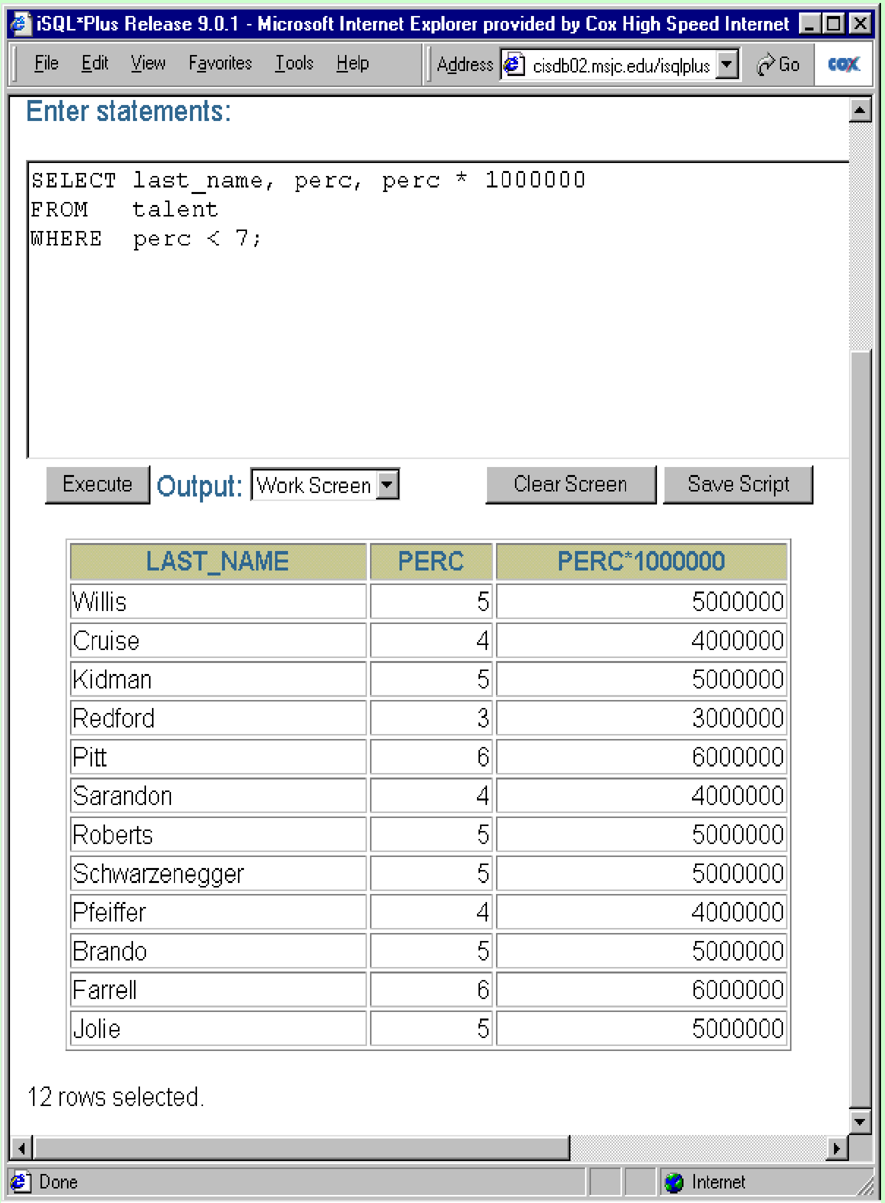
Id
last_name
first_name
birthdate
gender
home_town
home_state
home_country
perc
theatre
film
tv

Column Name/Expression	last_name	perc	Perc * 1000000
Table Name	talent	talent	...
Alias			...
Criteria	...		...
Display			

The mathematical expression is listed in it's own column in the TBC.

```
SELECT  last_name, perc, perc * 1000000
FROM     talent
WHERE   perc < 7;
```

Page B-7: Problem 6-1 Analysis


iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Speed Internet
File Edit View Favorites Tools Help
Address cisdb02.msjc.edu/isqlplus
Go
Enter statements:
Execute
Output: Work Screen
Clear Screen
Save Script
Done
Internet

```
SELECT last_name, perc, perc * 1000000
FROM   talent
WHERE  perc < 7;
```

LAST_NAME	PERC	PERC*1000000
Willis	5	5000000
Cruise	4	4000000
Kidman	5	5000000
Redford	3	3000000
Pitt	6	6000000
Sarandon	4	4000000
Roberts	5	5000000
Schwarzenegger	5	5000000
Pfeiffer	4	4000000
Brando	5	5000000
Farrell	6	6000000
Jolie	5	5000000

12 rows selected.

The math in this problem is simple multiplication.

perc * 1000000

Notice the placement of commas in the SELECT clause. Commas separate columns of data, ie, columns of data in the result table.

Page B-8: Mathematical Operators

Just as we saw that symbols could replace English phrases in comparison operations, special symbols are used to represent the mathematical operations as well.

Operation	Operator
Addition	+
Subtraction	-
Multiplication	*
Division	/

Unlike most other programming languages, the SQL standard does NOT provide a special operator for exponentiation.

Page B-9: Problem 6-2 Statement

Our user community needs a report …

Same as the last problem but is there anything we programmers can do to make the report a little bit prettier? The users don’t much care for that column heading that looks like computer speak:

```
perc * 1000000
```

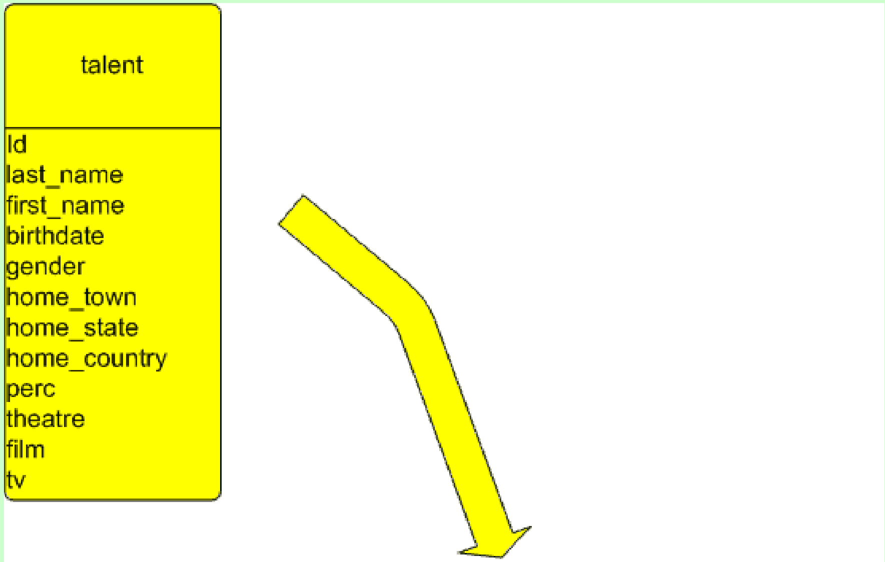
Rephrase: Use an alias to dress up the output

Module 05: Operations Page B-10: Problem 6-2 Design and Code

Step 1: Build the Table Build Chart (TBC)

Step 2: Double check your TBC solution

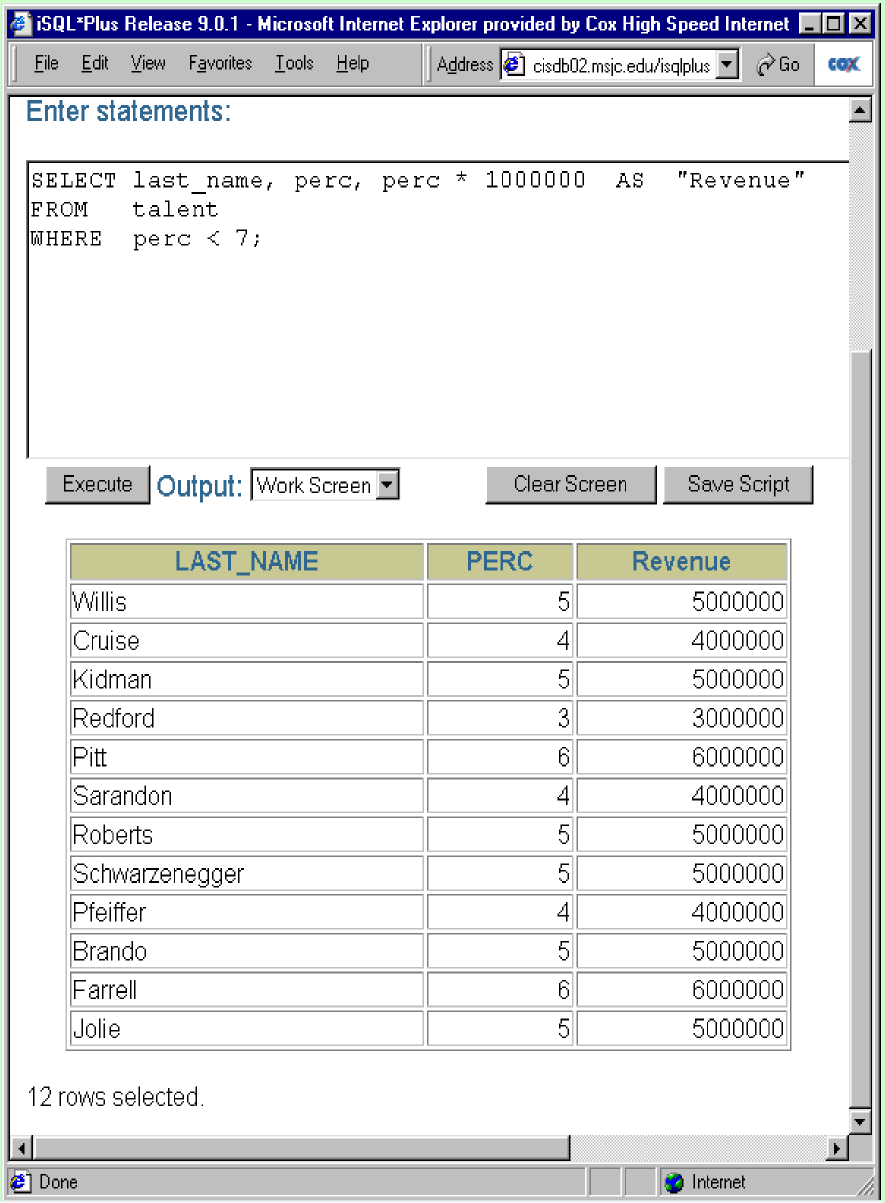
Step 3: Transform the TBC into code.



Column Name/Expression	last_name	perc	Perc * 1000000
Table Name	talent	talent	...
Alias			Revenue
Criteria	...		...
Display			

```
SELECT   last_name, perc, perc * 1000000 AS “Revenue”
FROM     talent
WHERE    perc < 7;
```

Page B-11: Problem 6-2 Analysis


iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Speed Internet
File Edit View Favorites Tools Help
Address cisdb02.msjc.edu/isqlplus
Go
Enter statements:
Execute
Output: Work Screen
Clear Screen
Save Script
Done
Internet

```
SELECT last_name, perc, perc * 1000000  AS  "Revenue"
FROM   talent
WHERE  perc < 7;
```

LAST_NAME	PERC	Revenue
Willis	5	5000000
Cruise	4	4000000
Kidman	5	5000000
Redford	3	3000000
Pitt	6	6000000
Sarandon	4	4000000
Roberts	5	5000000
Schwarzenegger	5	5000000
Pfeiffer	4	4000000
Brando	5	5000000
Farrell	6	6000000
Jolie	5	5000000

12 rows selected.

SQL is pretty consistent, and it makes learning it a bit easier.

In this case, we specify an alias for a derived column in the same manner that we would for a simple column specification.

Now then, is there anything that we can do to dress up the values in the Revenue column? Perhaps we could use a few dollar signs and commas to make the values more readable?

Nope!

You don’t know enough SQL to be able to do that, but you’ll be there in just a few lessons.

Page B-12: Problem 6-3 Statement

Our user community needs a report …

Management is thinking about raising the percentage rate, across the board, for all of our clients, a full half percent (.5)

What will the projected revenues be next year after this change takes effect (again assuming we secure million dollar contracts for all of our clients)

Rephrase:

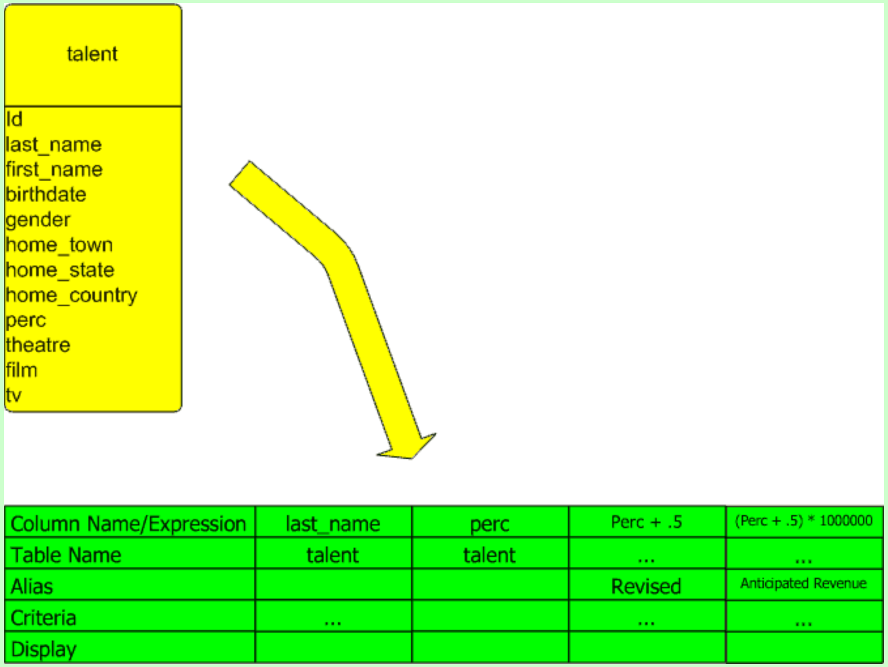
For each client show their last name, current percentage rate, revised percentage rate, anticipated revenues.

Module 05: Operations Page B-13: Problem 6-3 Design and Code

Step 1: Build the Table Build Chart (TBC)

Step 2: Double check your TBC solution

Step 3: Transform the TBC into code.



talent

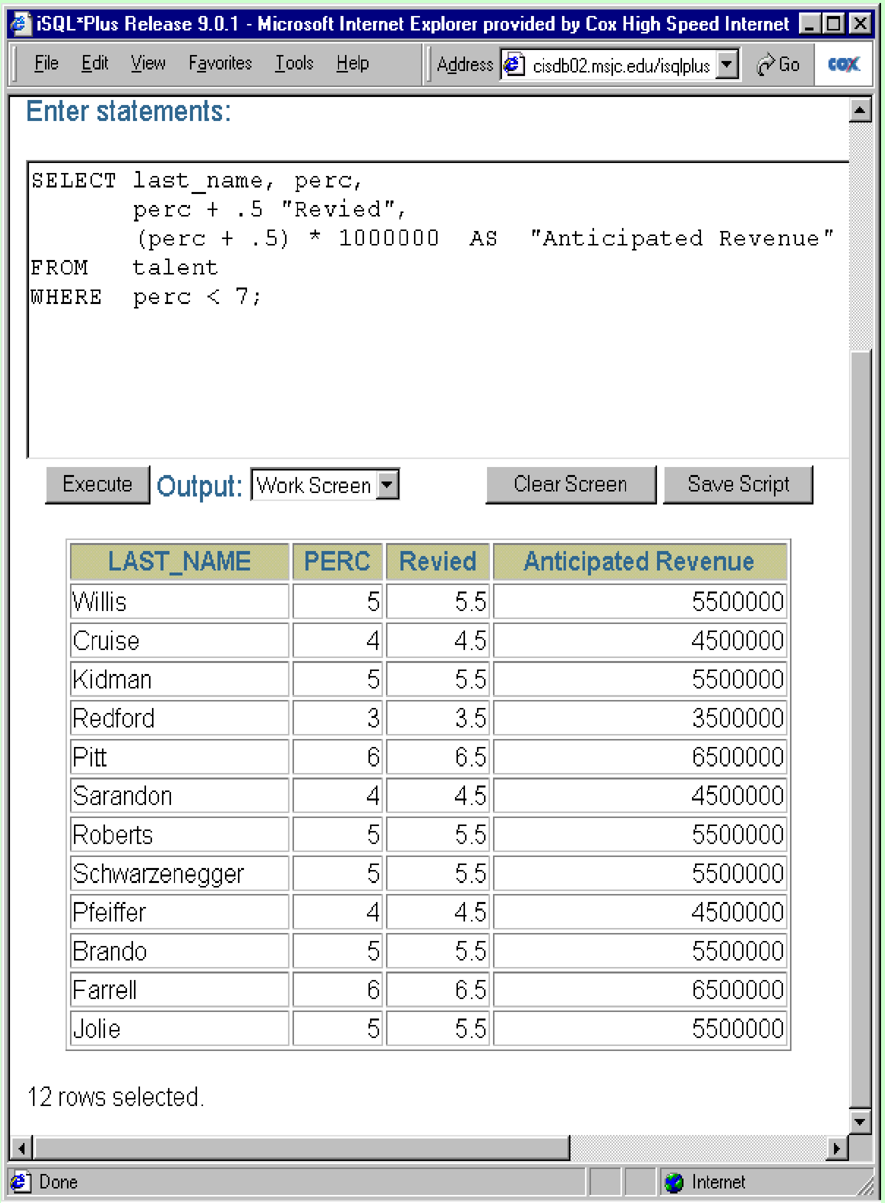
Id
last_name
first_name
birthdate
gender
home_town
home_state
home_country
perc
theatre
film
tv

Column Name/Expression	last_name	perc	Perc + .5	(Perc + .5) * 1000000
Table Name	talent	talent	...	...
Alias			Revised	Anticipated Revenue
Criteria	...		...	...
Display				

```
SELECT last_name, perc,
 perc + .5 “Revised”,
(perc + .5) * 1000000 “Anticipated Revenue”
FROM   talent
WHERE  perc < 7;
```

Regrets: I don’t much care for the way the code wraps on the slide here,but we’re a little cramped for space. Examine the solutions slide for a cleaner presentation of the code.

Page B-14: Problem 6-3 Analysis



iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Speed Internet
File Edit View Favorites Tools Help
Address cisdb02.msjc.edu/isqlplus
Go
Enter statements:

```
SELECT last_name, perc,
       perc + .5 "Revied",
       (perc + .5) * 1000000  AS  "Anticipated Revenue"
FROM   talent
WHERE  perc < 7;
```

Execute
Output: Work Screen
Clear Screen
Save Script

LAST_NAME	PERC	Revied	Anticipated Revenue
Willis	5	5.5	5500000
Cruise	4	4.5	4500000
Kidman	5	5.5	5500000
Redford	3	3.5	3500000
Pitt	6	6.5	6500000
Sarandon	4	4.5	4500000
Roberts	5	5.5	5500000
Schwarzenegger	5	5.5	5500000
Pfeiffer	4	4.5	4500000
Brando	5	5.5	5500000
Farrell	6	6.5	6500000
Jolie	5	5.5	5500000

12 rows selected.

The math in this problem is pretty simple: addition and multiplication, but you should take note of the way the parentheses are used in the calculation of anticipated revenue.

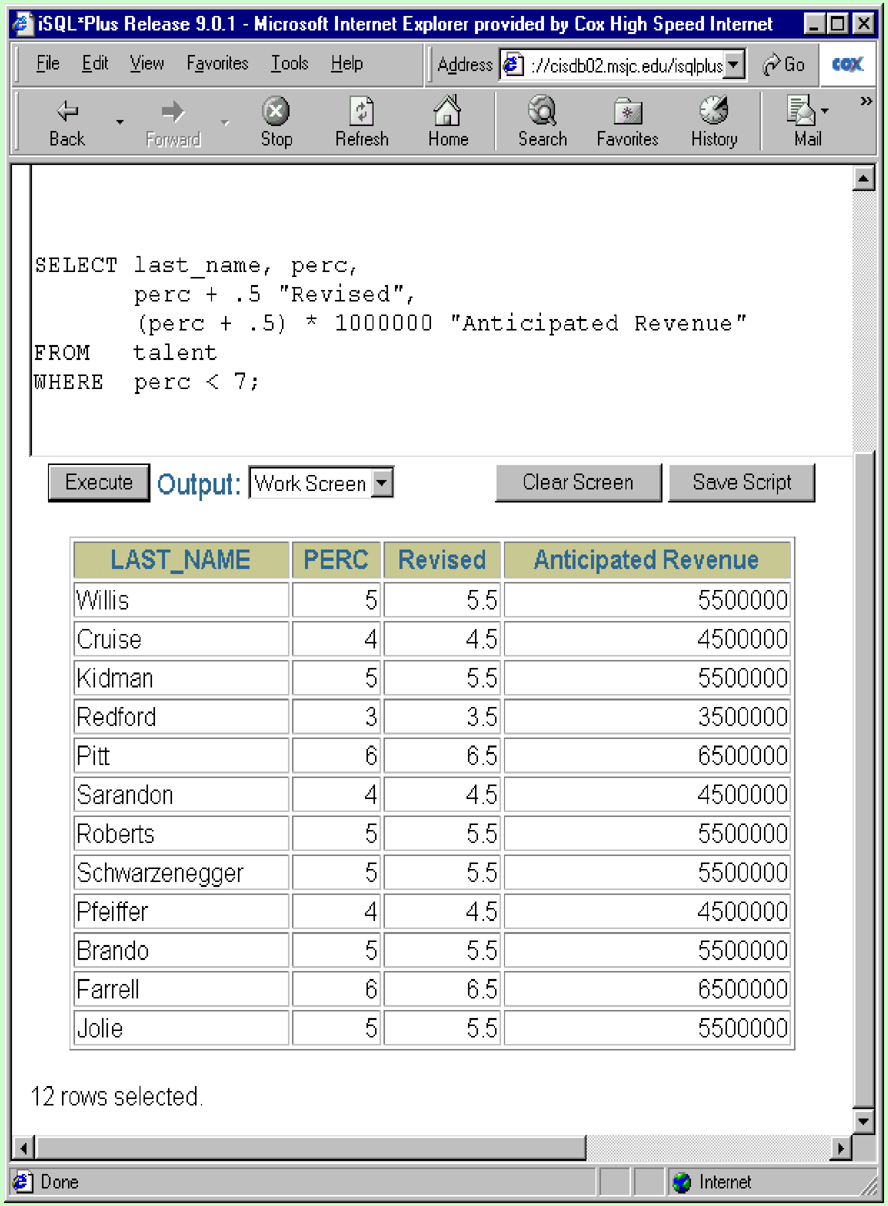
Check out the following slide.

The window on the left side shows the result when using parentheses in the calculation. The window on the right side shows the effect of the calculation without using parentheses.

The window on the left-hand side shows the correct way to use parenthesis for this calculation.

Module 05: Operations

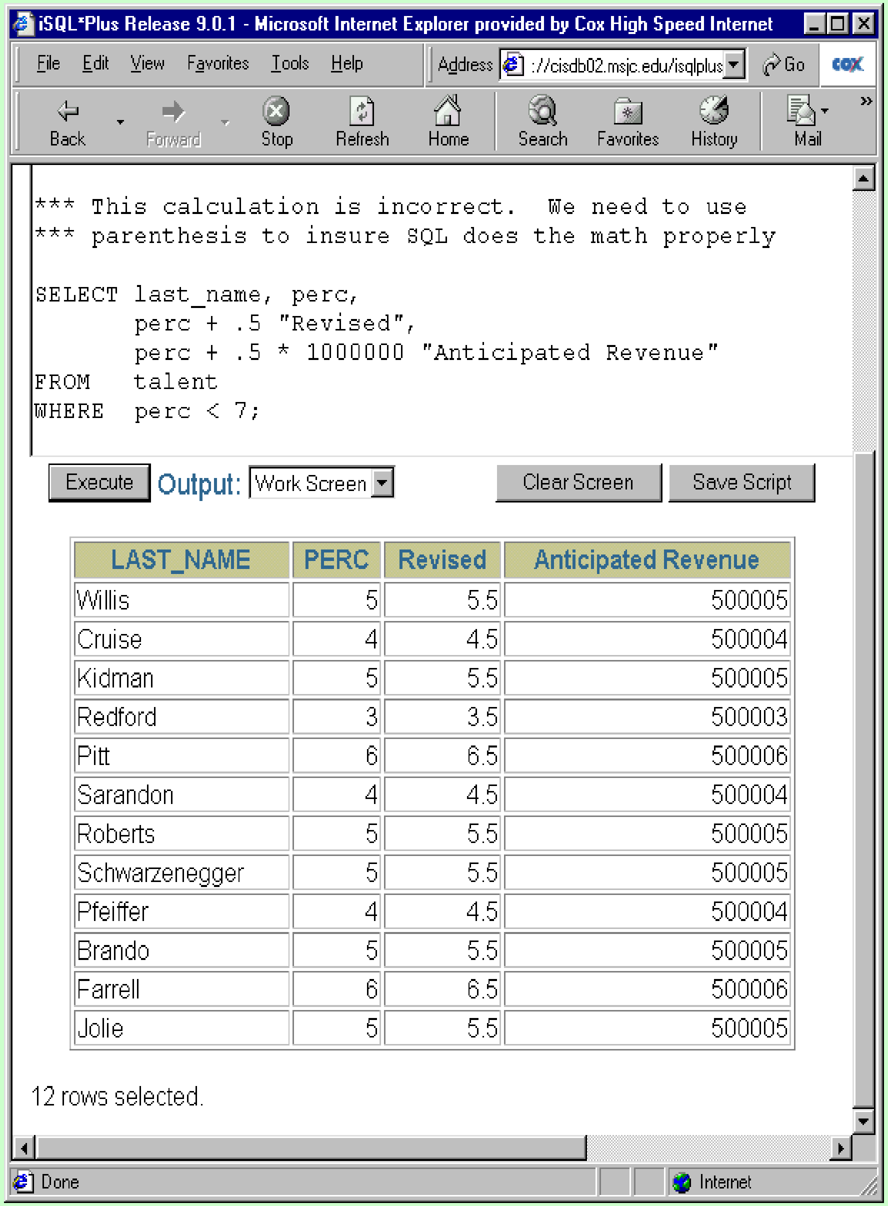
Page B-15: Precedence of Operations


iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Speed Internet
File Edit View Favorites Tools Help
Address ://cisdb02.msjc.edu/isqlplus Go
Back Forward Stop Refresh Home Search Favorites History Mail
Execute
Output: Work Screen
Clear Screen
Save Script
Done
Internet

```
SELECT last_name, perc,
       perc + .5 "Revised",
       (perc + .5) * 1000000 "Anticipated Revenue"
FROM   talent
WHERE  perc < 7;
```

LAST_NAME	PERC	Revised	Anticipated Revenue
Willis	5	5.5	5500000
Cruise	4	4.5	4500000
Kidman	5	5.5	5500000
Redford	3	3.5	3500000
Pitt	6	6.5	6500000
Sarandon	4	4.5	4500000
Roberts	5	5.5	5500000
Schwarzenegger	5	5.5	5500000
Pfeiffer	4	4.5	4500000
Brando	5	5.5	5500000
Farrell	6	6.5	6500000
Jolie	5	5.5	5500000

12 rows selected.


iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Speed Internet
File Edit View Favorites Tools Help
Address ://cisdb02.msjc.edu/isqlplus Go
Back Forward Stop Refresh Home Search Favorites History Mail
Execute
Output: Work Screen
Clear Screen
Save Script
Done
Internet

```
*** This calculation is incorrect.  We need to use
*** parenthesis to insure SQL does the math properly

SELECT last_name, perc,
       perc + .5 "Revised",
       perc + .5 * 1000000 "Anticipated Revenue"
FROM   talent
WHERE  perc < 7;
```

LAST_NAME	PERC	Revised	Anticipated Revenue
Willis	5	5.5	500005
Cruise	4	4.5	500004
Kidman	5	5.5	500005
Redford	3	3.5	500003
Pitt	6	6.5	500006
Sarandon	4	4.5	500004
Roberts	5	5.5	500005
Schwarzenegger	5	5.5	500005
Pfeiffer	4	4.5	500004
Brando	5	5.5	500005
Farrell	6	6.5	500006
Jolie	5	5.5	500005

12 rows selected.

Page B-16: Precedence of Operations

Precedence of operations is another one of those computing terms, but not to worry, it is easily decipherable using your skills in English.

Precedence of operations means:

Which operations take precedence over other operations, or

Which operations precede others?

Precedence only matters if you have more than one operation, so precedence of operations is all about specifying which part of the expression gets evaluated before the other parts of the expression.

In math, the precedence rules state that

Multiplication and division precede addition and subtraction.

But if any part of the expression is wrapped in parentheses, then that part is evaluated first.

Page B-17: Precedence of Operations

Precedence of Arithmetic Operations

1. Expressions in parentheses are evaluated first (left to right)
2. Then multiplication and division (reading left to right)
3. Finally, addition and subtraction, (again reading left to right)

Let’s work thru a few examples

Page B-18: Precedence of Operations

Evaluate each of these expressions before reading the solutions on the next slide.

1. 5 + 3 + 2

2. 5 * 3 + 2

3. 5 * (3 + 2)

4. 5 + 2 * 3

5. 5 + 3 * 2 + 4

6. 2 * 3 + 4 * 2

Module 05: Operations   Page B-19: Precedence of Operations

Evaluate each of these expressions before reading the solutions on the next slide.

1. 5 + 3 + 2

2. 5 * 3 + 2

3. 5 * (3 + 2)

4. 5 + 2 * 3

5. 5 + 3 * 2 + 4

6. 2 * 3 + 4 * 2

1. 5 + 3 comes first
8 + 2 = 10

2. 5 * 3 comes first
15 + 2 = 17

3. (3 + 2) comes first
5 * 5 = 25

4. 2 * 3 comes first
5 + 6 = 11

5. 3 * 2 comes first
then 5 + 6 = 11
then 11 + 4 = 15

6. 2 * 3 comes first, then 4 * 2
then 6 + 8 = 14

Page B-20: Problem 6-4 Statement

Building on that last problem,

Our users now want a report that shows how much more money we'll be bringing in after the change has taken effect.

Rephrase:

Our users now want a report that shows how much more money, **per client**, we'll be bringing in after the change has taken effect.

Rephrase:

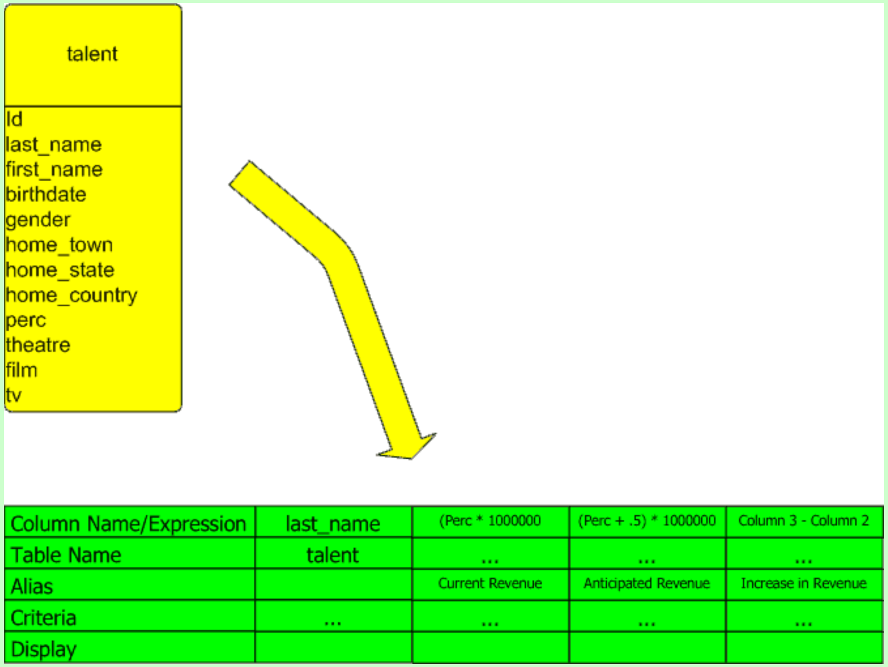
For each client, show their name, their current revenue (assuming million dollar contract), their anticipated revenue (after the ½ percent increase in percentage), and the difference between these two values.

Page B-21: Problem 6-4 Design

Step 1: Build the Table Build Chart (TBC)

Step 2: Double check your TBC solution

Step 3: Transform the TBC into code.



talent

Id
last_name
first_name
birthdate
gender
home_town
home_state
home_country
perc
theatre
film
tv

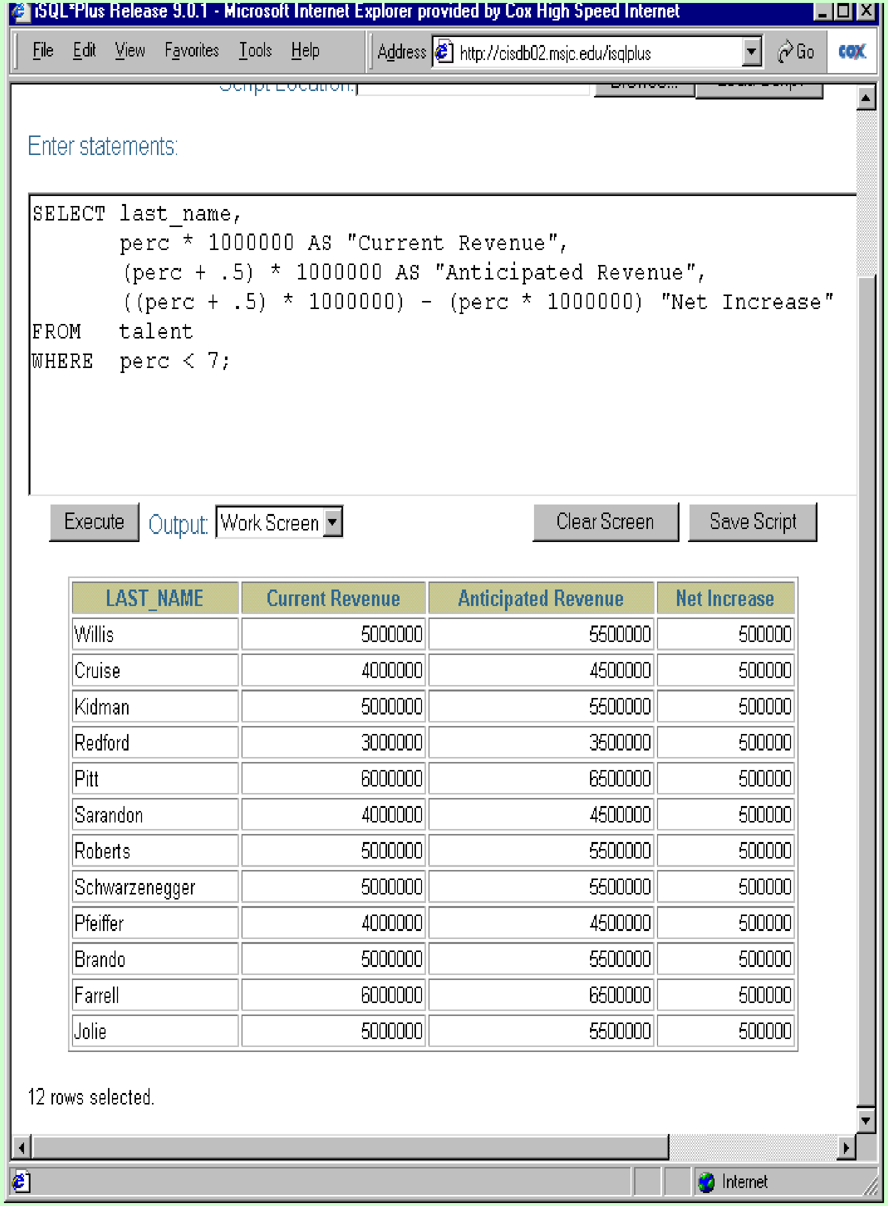
Column Name/Expression	last_name	(Perc * 1000000	(Perc + .5) * 1000000	Column 3 - Column 2
Table Name	talent	...	...	...
Alias		Current Revenue	Anticipated Revenue	Increase in Revenue
Criteria	...	...	...	...
Display				

I used a shortcut in the rightmost column of the TBC by specifying the column names rather than the calculation. I only did that so that you might be able to read the slide.

Check out the code on the following slide.

Module 05: Operations

Page B-22: Problem Correction


iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Speed Internet
File Edit View Favorites Tools Help
Address http://cisdb02.msjc.edu/isqlplus
Go
Enter statements:
Execute
Output: Work Screen
Clear Screen
Save Script
Internet

```
SELECT last_name,
       perc * 1000000 AS "Current Revenue",
       (perc + .5) * 1000000 AS "Anticipated Revenue",
       ((perc + .5) * 1000000) - (perc * 1000000) "Net Increase"
FROM   talent
WHERE  perc < 7;
```

LAST_NAME	Current Revenue	Anticipated Revenue	Net Increase
Willis	5000000	5500000	500000
Cruise	4000000	4500000	500000
Kidman	5000000	5500000	500000
Redford	3000000	3500000	500000
Pitt	6000000	6500000	500000
Sarandon	4000000	4500000	500000
Roberts	5000000	5500000	500000
Schwarzenegger	5000000	5500000	500000
Pfeiffer	4000000	4500000	500000
Brando	5000000	5500000	500000
Farrell	6000000	6500000	500000
Jolie	5000000	5500000	500000

```
12 rows selected.
```

Examine the output carefully.

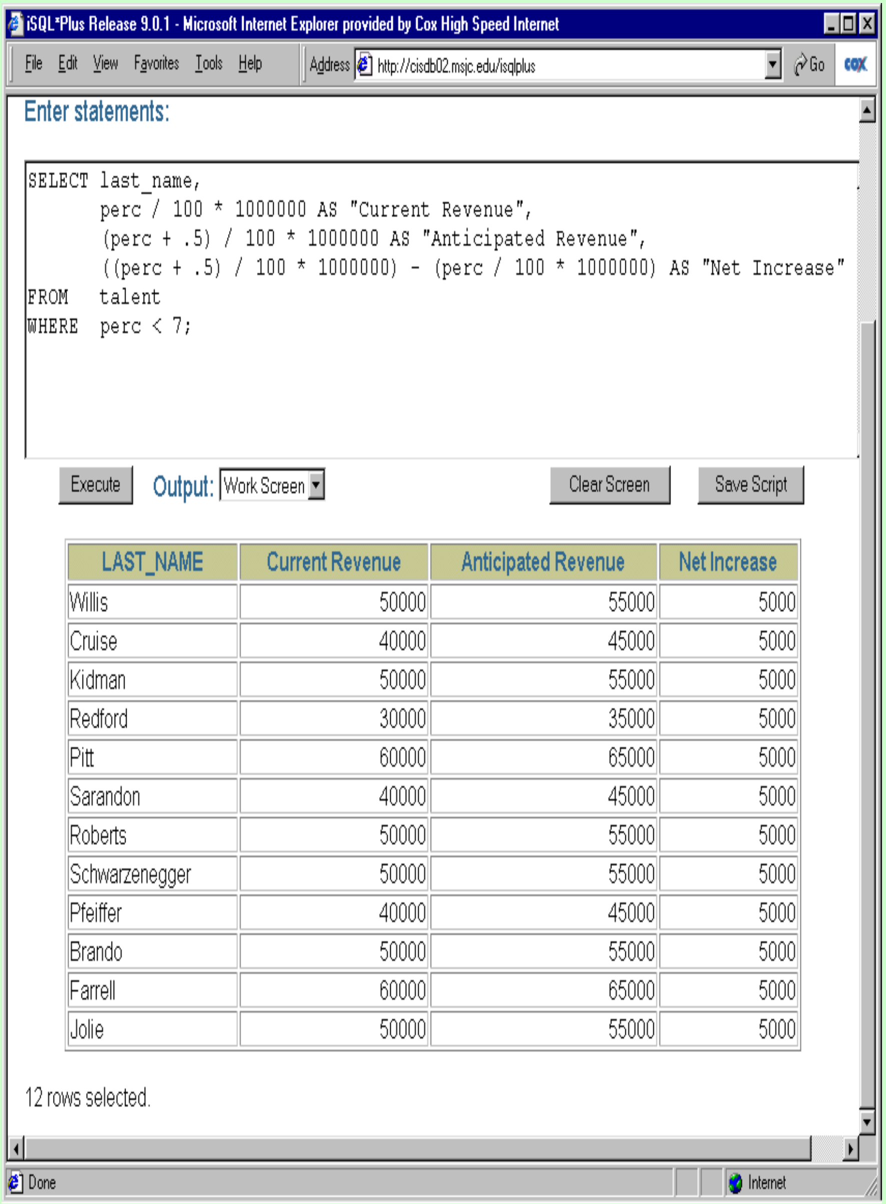
Hey, wait a second! This isn't right. Everybody is being charged an extra ½ percent, but the net increase column shows an increase per client of ½ million.

Something's wrong.

Oh man! All of these calculations are off! I misinterpreted the data!!!

The perc column in our database system shows the percentage value as a decimal. So an 8% piece of the action is stored in the table as 8. If I want to use that value as a percentage, I need to divide it by 100.

I can say good bye to the Christmas bonus this year – if I still have a job.


iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Speed Internet
File Edit View Favorites Tools Help
Address http://cisdb02.msjc.edu/isqlplus
Go
Enter statements:
Execute
Output: Work Screen
Clear Screen
Save Script
Done
Internet

```
SELECT last_name,
       perc / 100 * 1000000 AS "Current Revenue",
       (perc + .5) / 100 * 1000000 AS "Anticipated Revenue",
       ((perc + .5) / 100 * 1000000) - (perc / 100 * 1000000) AS "Net Increase"
FROM   talent
WHERE  perc < 7;
```

LAST_NAME	Current Revenue	Anticipated Revenue	Net Increase
Willis	50000	55000	5000
Cruise	40000	45000	5000
Kidman	50000	55000	5000
Redford	30000	35000	5000
Pitt	60000	65000	5000
Sarandon	40000	45000	5000
Roberts	50000	55000	5000
Schwarzenegger	50000	55000	5000
Pfeiffer	40000	45000	5000
Brando	50000	55000	5000
Farrell	60000	65000	5000
Jolie	50000	55000	5000

12 rows selected.

Okay, so the mistake was contrived.

Still, you need to keep your wits about you. This isn't rocket science, it's simple math. You ought to be able to desk check your work and take your test values and compare them to your results.

Step 2 in our design methodology requires you to test your TBC. That means work it out. By hand if you have to. Use data that you're familiar with. A few moments of your time can save you a lot of grief – especially in a production environment.

(← Here's the correction)

Consider the cost.

How much time did you spend developing that SQL code, typing it, correcting syntax errors, and polishing it up – only to realize it was all a waste. You did all of that work for an incorrect solution.

Page B-24: Programming Strategies

Now then, how about the cost to your career?

‘Nuff said.

I need to go off and write this mistake down in my programming journal.

Page C-1: Numeric Ops - WHERE Clause

Numeric operations are included in the SELECT clause when we want to see the result.

You also know that numeric operations can be included in the WHERE clause as well.

So the million dollar question is this:

Why would you perform any numeric operations in a WHERE clause?

Page C-2: WHERE Clause

Numeric operations are included in the predicate expression of a WHERE clause when we want SQL to use the result of that mathematical operation as one of the criteria for including rows of data from the base table in the result table.

Page C-3: Problem 6-5 Statement

Our user community needs a report …

Management is still mulling over the proposal to raise the percentage rate for all of our clients. They'd like to see a report listing the talent whose percentage rate would be above the industry average, if the rate increase was 1.5%

Rephrase

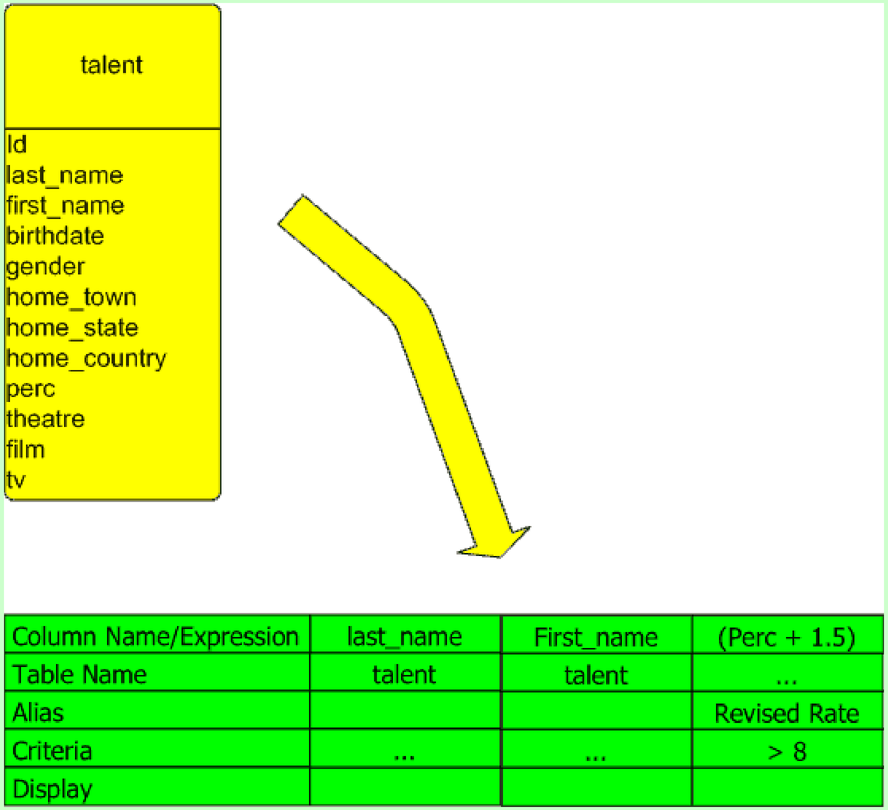
Prepare a report that shows the name and projected percentage rate of our clients whose new percentage rate would be over 8%.

Page C-4: Problem 6-5 Design and Code

Step 1: Build the Table Build Chart (TBC)

Step 2: Double check your TBC solution

Step 3: Transform the TBC into code.

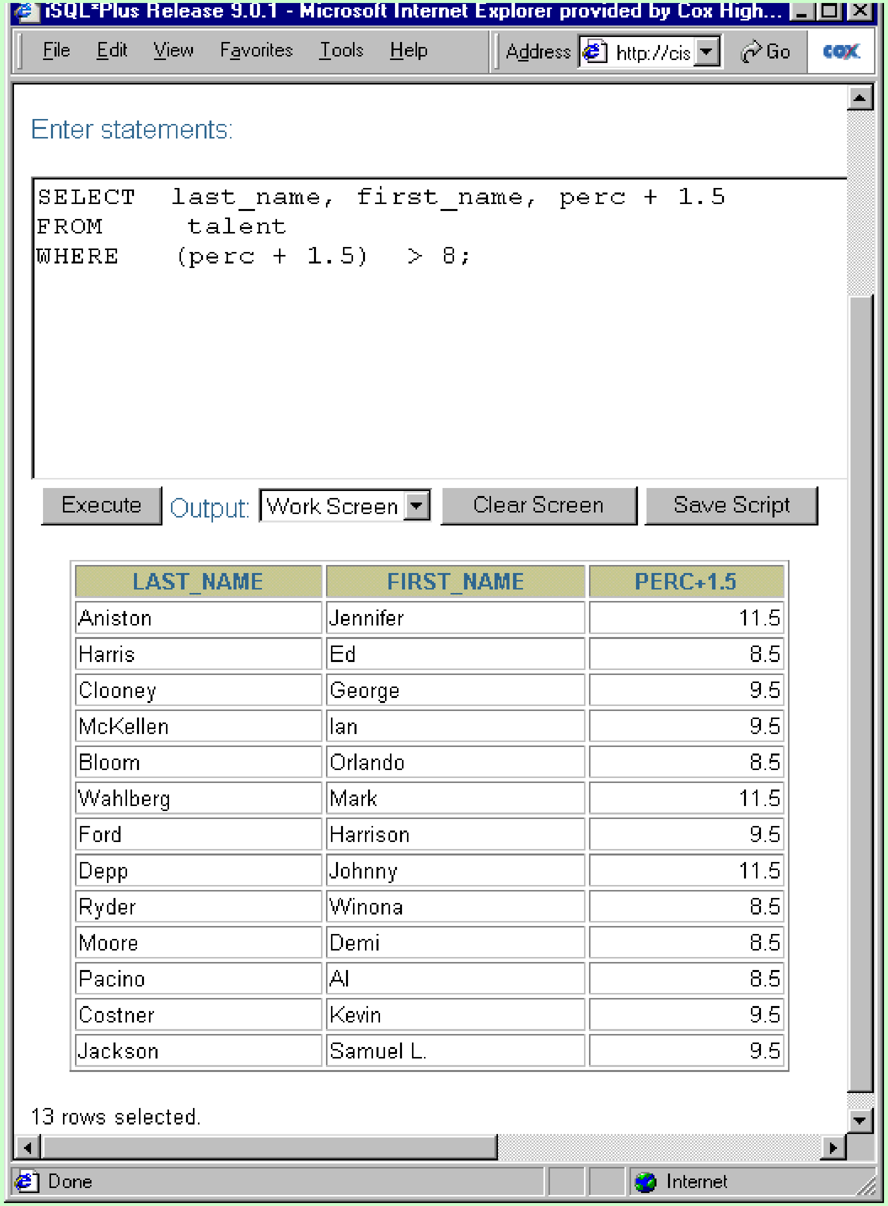


Column Name/Expression	last_name	First_name	(Perc + 1.5)
Table Name	talent	talent	...
Alias			Revised Rate
Criteria	...	...	> 8
Display			

The mathematical expression is listed in it's own column in the TBC. And that column is used as part of the WHERE criteria

```
SELECT  last_name, first_name, perc + 1.5
FROM    talent
WHERE   (perc + 1.5)  > 8;
```

Page C-5: Problem 6-5 Analysis



iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High...
File Edit View Favorites Tools Help
Address http://cis
Go
Enter statements:
Execute
Output: Work Screen
Clear Screen
Save Script
Done
Internet

```
SELECT  last_name, first_name, perc + 1.5
FROM     talent
WHERE   (perc + 1.5)  > 8;
```

LAST_NAME	FIRST_NAME	PERC+1.5
Aniston	Jennifer	11.5
Harris	Ed	8.5
Clooney	George	9.5
McKellen	Ian	9.5
Bloom	Orlando	8.5
Wahlberg	Mark	11.5
Ford	Harrison	9.5
Depp	Johnny	11.5
Ryder	Winona	8.5
Moore	Demi	8.5
Pacino	Al	8.5
Costner	Kevin	9.5
Jackson	Samuel L.	9.5

13 rows selected.

The parentheses in this example are not required - I included them to improve the program’s readability.

Any questions?

Page C-6: Generalities

The numeric operations that we've looked at so far have involved a single column from a table combined in some operation with numeric literals.

```
Perc + .5
(perc + .5) / 100
```

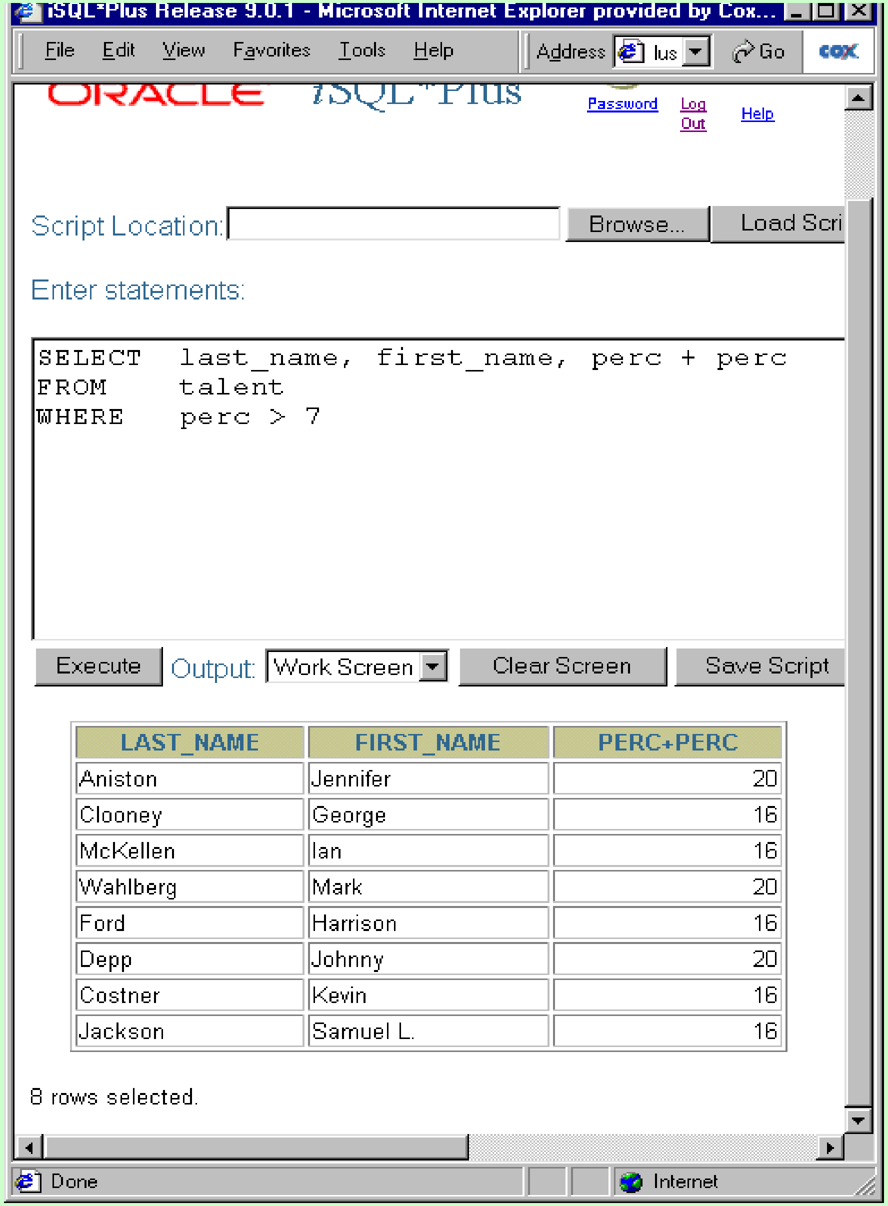
Please don't be misled into thinking that this is the general form for math operations.

Numeric operations can apply to ANY numeric value. This means that we can have combinations of numeric columns and numeric literals.

For example:

Numeric column * numeric literal
Numeric column * numeric column
Numeric literal * numeric literal

Page C-7: Numeric columns



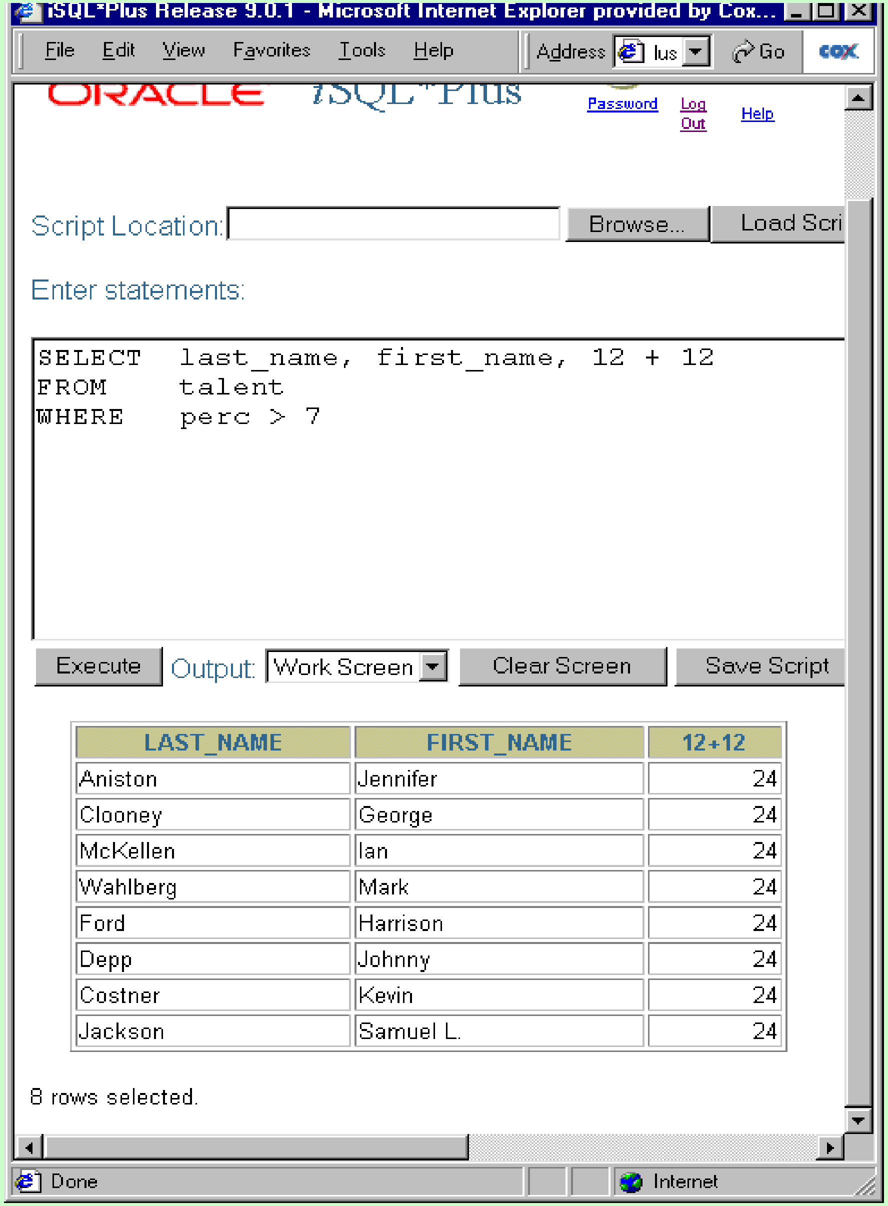
```
SELECT   last_name, first_name, perc + perc
FROM     talent
WHERE    perc > 7
```

LAST_NAME	FIRST_NAME	PERC+PERC
Aniston	Jennifer	20
Clooney	George	16
McKellen	Ian	16
Wahlberg	Mark	20
Ford	Harrison	16
Depp	Johnny	20
Costner	Kevin	16
Jackson	Samuel L.	16

8 rows selected.

Module 05: Operations

Page C-8: Numeric literals



```
SELECT  last_name, first_name, 12 + 12
FROM    talent
WHERE   perc > 7
```

LAST_NAME	FIRST_NAME	12+12
Aniston	Jennifer	24
Clooney	George	24
McKellen	Ian	24
Wahlberg	Mark	24
Ford	Harrison	24
Depp	Johnny	24
Costner	Kevin	24
Jackson	Samuel L.	24

8 rows selected.

Page D-1: Calculator

Which leads me to my next point…

Can I use SQL as a calculator? Say I’ve got a math problem, I’m adding up a few numbers and my calculator isn’t handy.

Can I use SQL to do the math for me?

Sure.

Page D-2: Problem 6-6

I’m filling out a purchase requisition and the items cost:

32.95

16.50

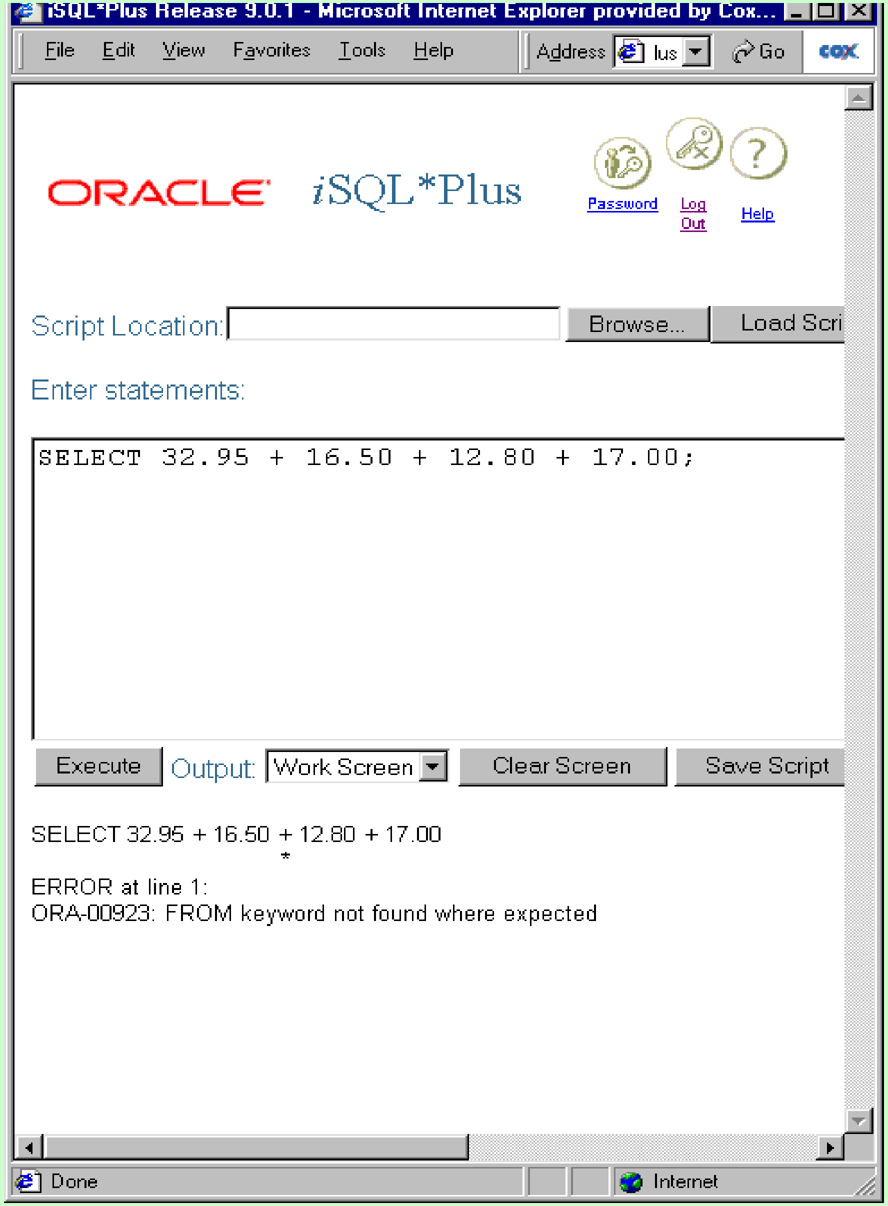
12.80

17.00

Write a SQL program to ‘do the math’ and figure out the total.

```
SELECT 32.95 + 16.50 + 12.80 + 17.00;
```

Page D-3: Error



```
SELECT 32.95 + 16.50 + 12.80 + 17.00;
```

```
SELECT 32.95 + 16.50 + 12.80 + 17.00
                      *
ERROR at line 1:
ORA-00923: FROM keyword not found where expected
```

But this is an invalid SQL program.

Why?

Look at the error message:

FROM keyword not found where expected

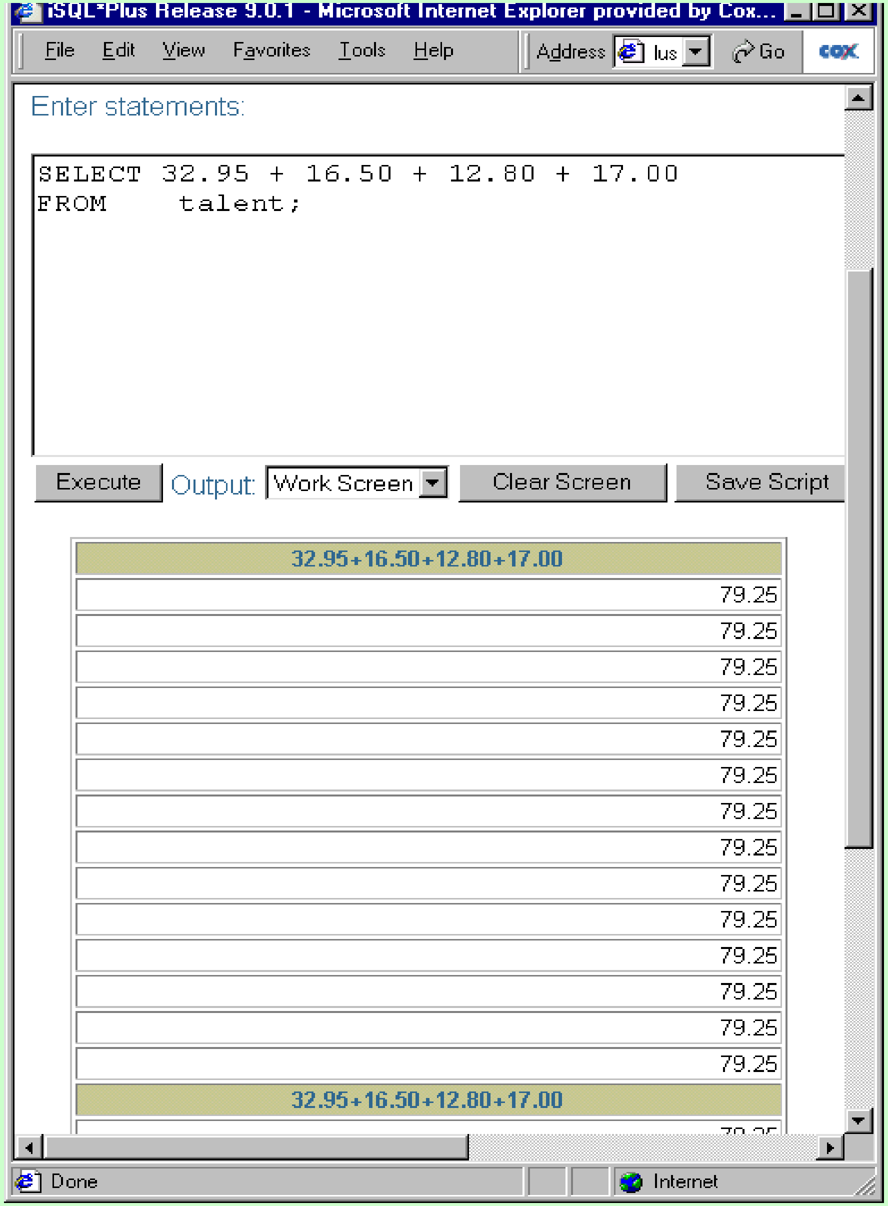
SQL couldn't find the FROM clause. As it turns out, every SQL program requires a SELECT clause and a FROM clause.

No big deal, I've got plenty of tables to choose from!

```
SELECT 32.95 + 16.50 + 12.80 + 17.00
FROM    talent;
```

Module 05: Operations

Page D-4 Correction

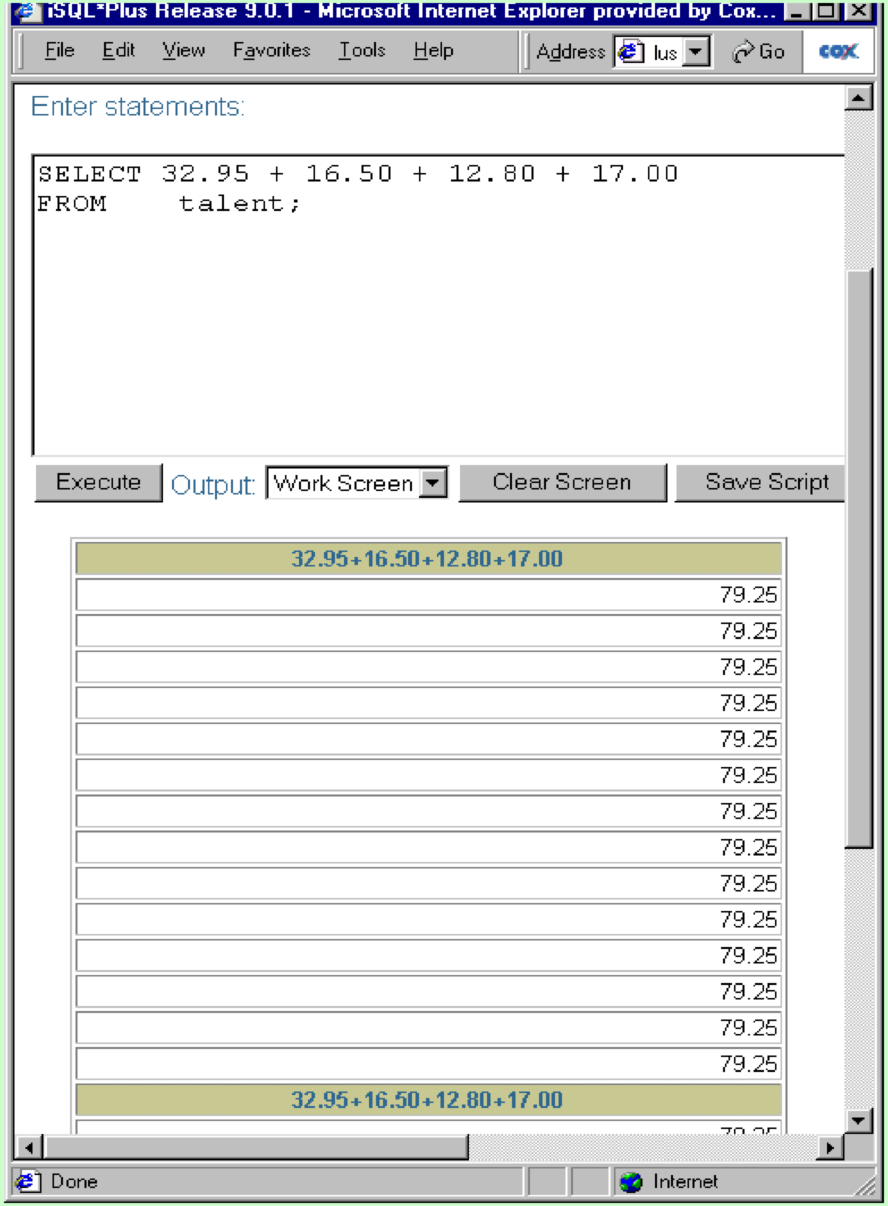

iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox...
File Edit View Favorites Tools Help
Address lus Go
Enter statements:
Execute
Output: Work Screen
Clear Screen
Save Script
Done
Internet

```
SELECT 32.95 + 16.50 + 12.80 + 17.00
FROM     talent;
```

```
32.95+16.50+12.80+17.00
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
32.95+16.50+12.80+17.00
```

Why are there so many lines with the same answer????

Page D-5 WHERE clause



```
SELECT 32.95 + 16.50 + 12.80 + 17.00
FROM     talent;
```

```
32.95+16.50+12.80+17.00
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
                  79.25
32.95+16.50+12.80+17.00
```

Does this program have a WHERE clause?

No, it doesn’t. Hence every row in the base table will be included in the result table. So every row in the talent table is displayed here in the result table.

Which columns from the talent table were selected?

None of the columns – I’m only displaying the result of some math operation. But that operation is applied to each row. So each base table row is carried over to the result table, hence the result table has as many rows as the base table.

This is overkill. I only need one row of output to see the result of my calculation. What would be ideal would be if there were a table somewhere that just had one row of information in it. Then I could do all of my math using that table, and my result table would have just one row!

Page D-6 Single? Nope, dual??

In my ideal world, that table would be named SINGLE, and it would contain only a single row with a single column.

But this isn't the ideal world, and that table doesn't come with the database. ☹

--

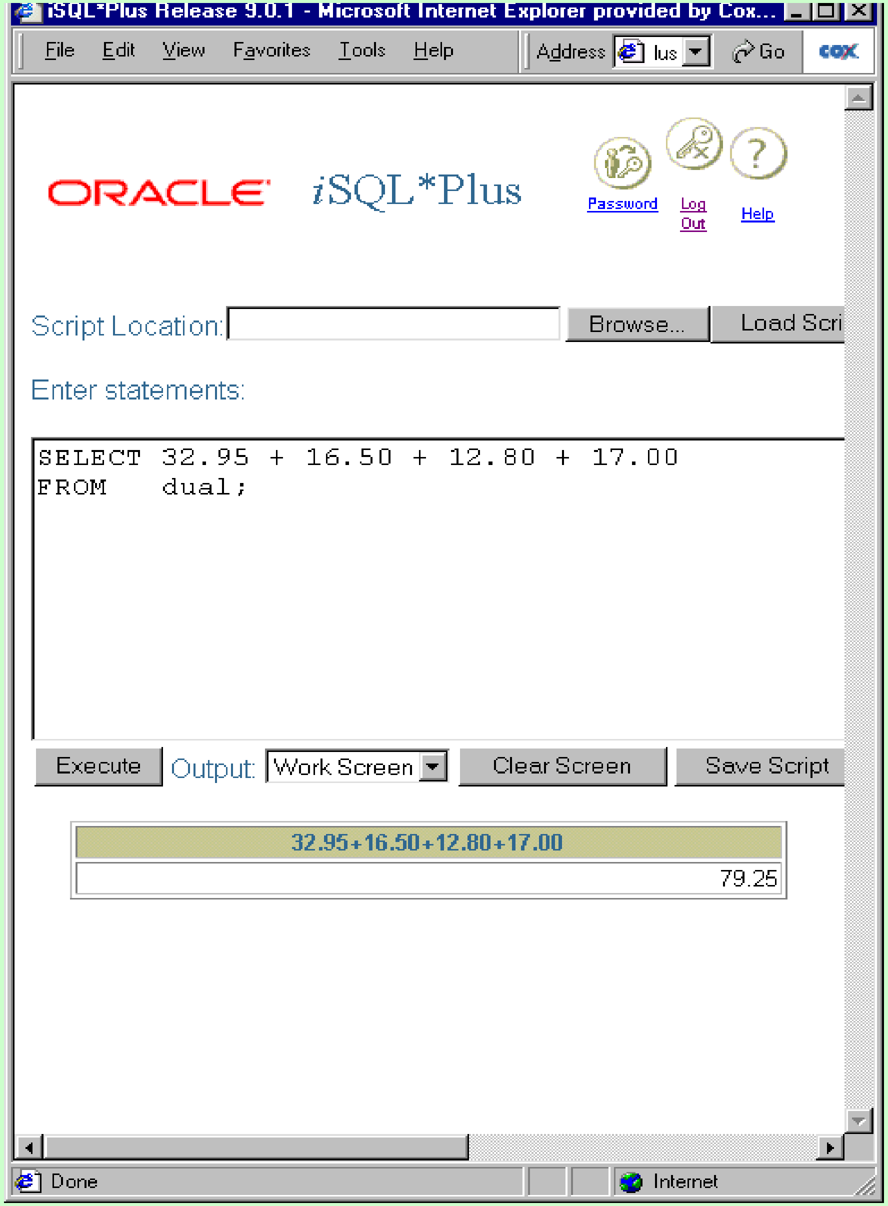
Curiously though, Oracle does have a table named DUAL, that contains a single column and a single row. You can use DUAL on every Oracle database that you encounter.

--

If you don't like using DUAL, feel free to create your own SINGLE table and use that.

But if you're going to be working in an Oracle environment you'll be expected to know about DUAL, and at least understand it if you encounter it in someone else's SQL program.

Page D-7 Correction



```
SELECT 32.95 + 16.50 + 12.80 + 17.00
FROM   dual;
```

32.95+16.50+12.80+17.00
79.25

There now, that's much better!

Page E-1 Character Operations

Character operations are used in pretty much the same fashion as numeric operations.

They may be used in almost every clause, except the FROM clause.

Character operations may apply against column values or character literals – just as long as each of the values is a character value.

Page E-2 Concatenation

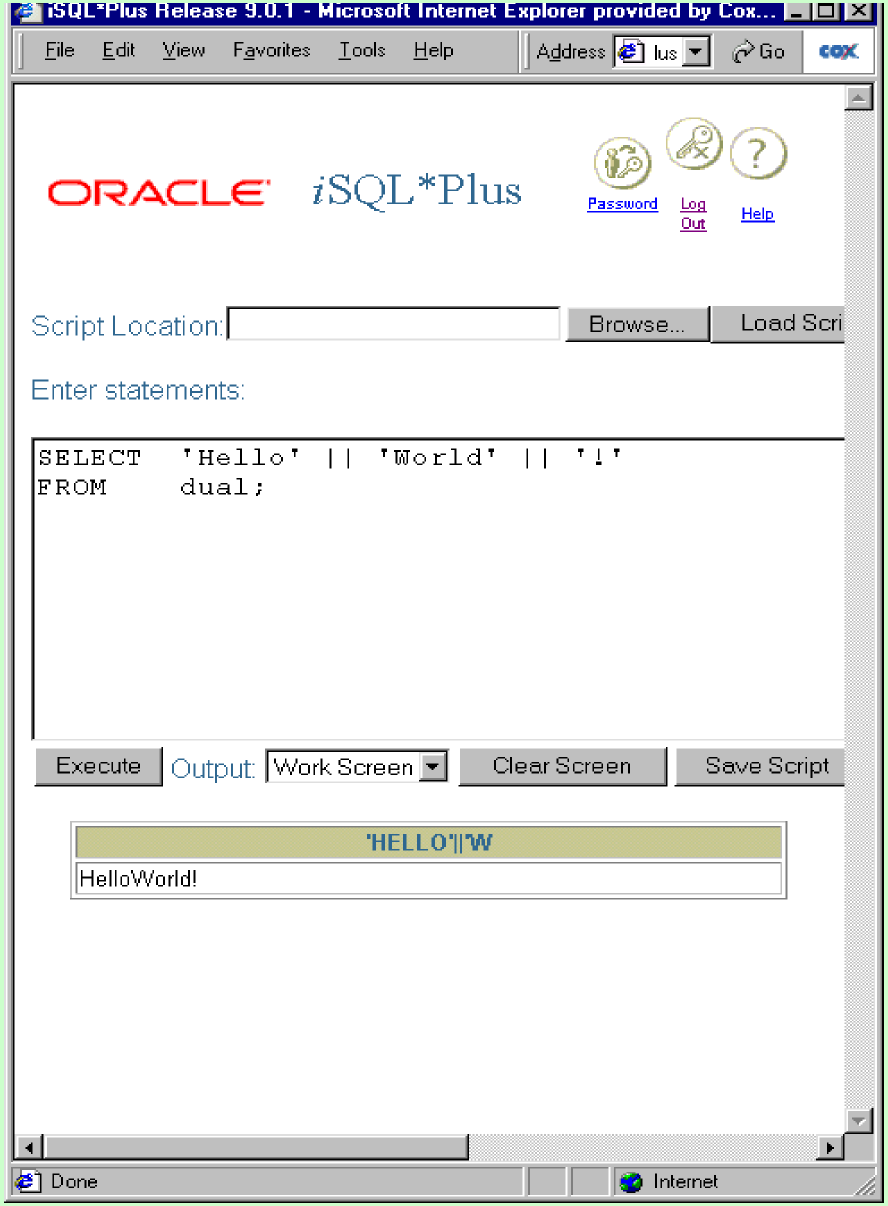
The cool things about character operations from the student's perspective is that there's really only one operation to learn: concatenation.

Concatenation is just a fancy term for *smooshing*. You take two character values and smoosh 'em together.

Webster's has a more precise definition: '*to link together in a series or chain'.*

Concatenation (smooshing) is demonstrated on the following slide.

Page E-3 Example - 1



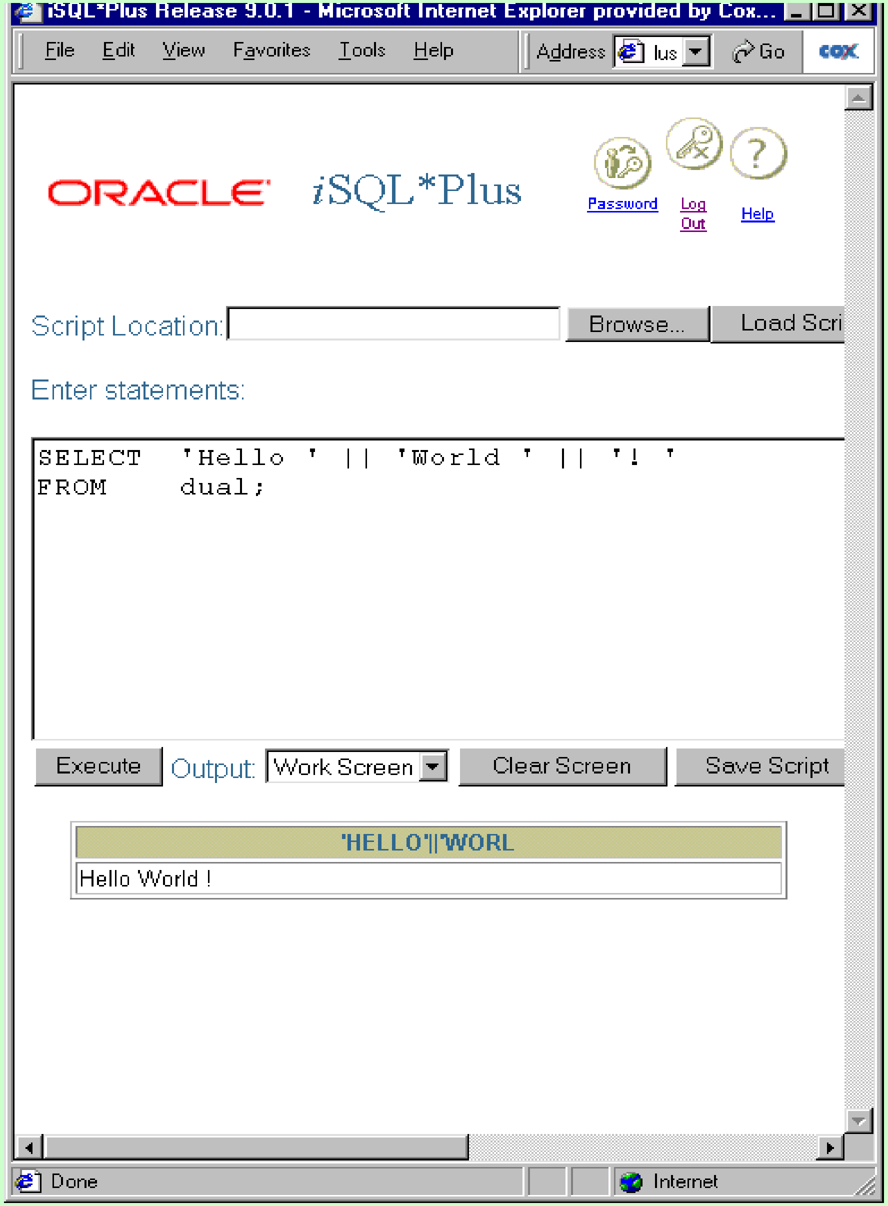
```
SELECT  'Hello' || 'World' || '!'
FROM    dual;
```

'HELLO'||'W
HelloWorld!

The concatenation operator, or symbol, is a pair of vertical bars.

Commas are not present in this SELECT clause because, the output is intended to appear in a single column in the result table.

Page E-4 Example - 2



```
SELECT  'Hello ' || 'World ' || '! '
FROM    dual;
```

'HELLO'||'WORL
Hello World !

The only difference between this SQL program and the program in the preceding example is the embedded space character that's been added on the right-hand side of each of the character literals.

Page E-5 Problem 6-7

Our user community would like a report that lists the name for each of our clients. They’d like the names listed in alphabetical order (by last name) and they want the information presented in first_name first, last_name last, order.

Page E-6 Problem 6-7 Design & Code

My first pass at this problem design would be something like this:

```
SELECT      first_name, last_name
FROM        talent
ORDER BY  last_name;
```

Now I’ll refine this solution so that the output is more aesthetically pleasing.

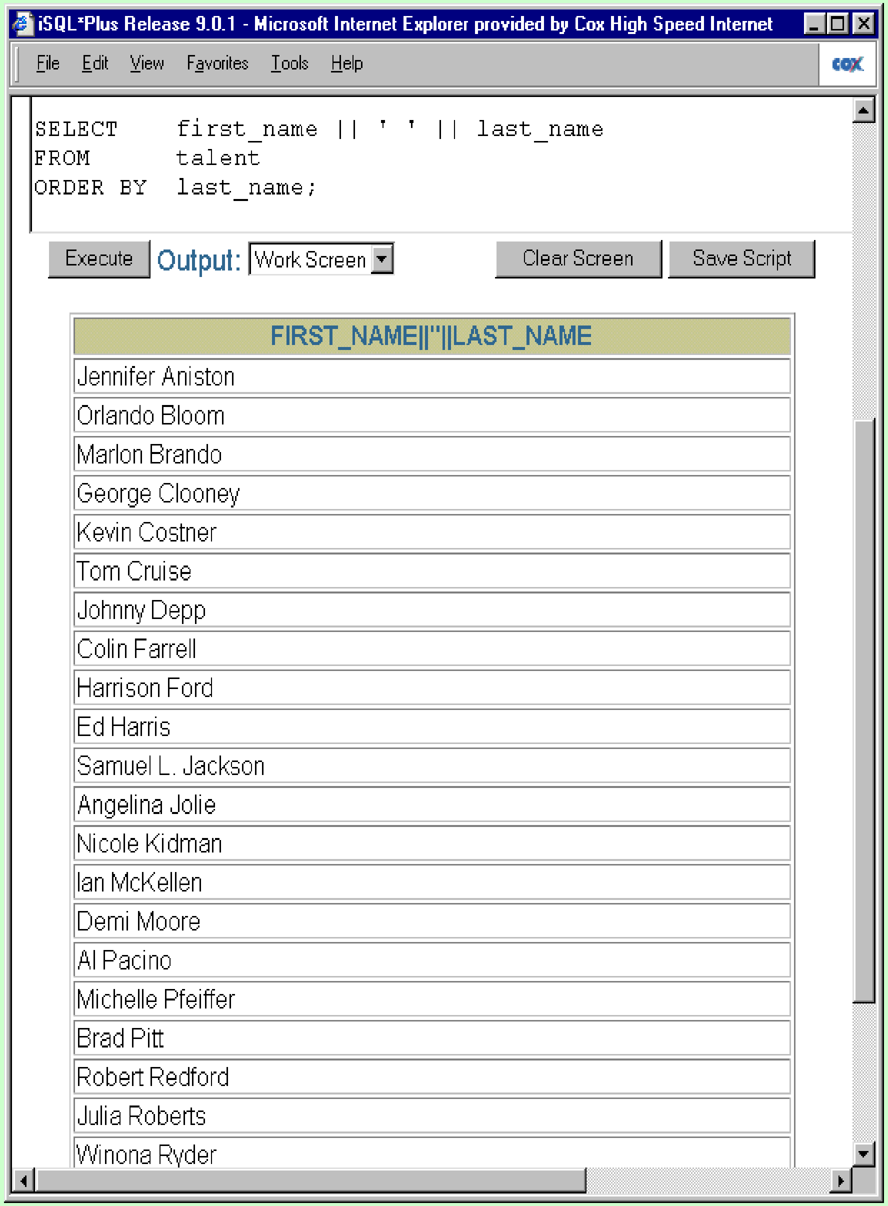
```
SELECT      first_name || last_name
FROM        talent
ORDER BY  last_name;
```

Not quite there.

```
SELECT      first_name || ‘ ‘ || last_name
FROM        talent
ORDER BY  last_name;
```

That’s more better! ☺

Page E-7 Problem 6-7 Result


iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Speed Internet
File
Edit
View
Favorites
Tools
Help
cox
Execute
Output:
Work Screen
Clear Screen
Save Script

```
SELECT    first_name || ' ' || last_name
FROM      talent
ORDER BY  last_name;
```

```
FIRST_NAME||''||LAST_NAME
Jennifer Aniston
Orlando Bloom
Marlon Brando
George Clooney
Kevin Costner
Tom Cruise
Johnny Depp
Colin Farrell
Harrison Ford
Ed Harris
Samuel L. Jackson
Angelina Jolie
Nicole Kidman
Ian McKellen
Demi Moore
Al Pacino
Michelle Pfeiffer
Brad Pitt
Robert Redford
Julia Roberts
Winona Ryder
```

Please work through the examples on the preceding page.

Page F-1 Date Ops

Date operations allow us to manipulate date values.

Where are date operations allowed to appear in a SQL program?

Bueno!

Date operations may be used in almost every clause with the exception of the FROM clause.

Now then, what kinds of operations does a programmer need to perform on date values?

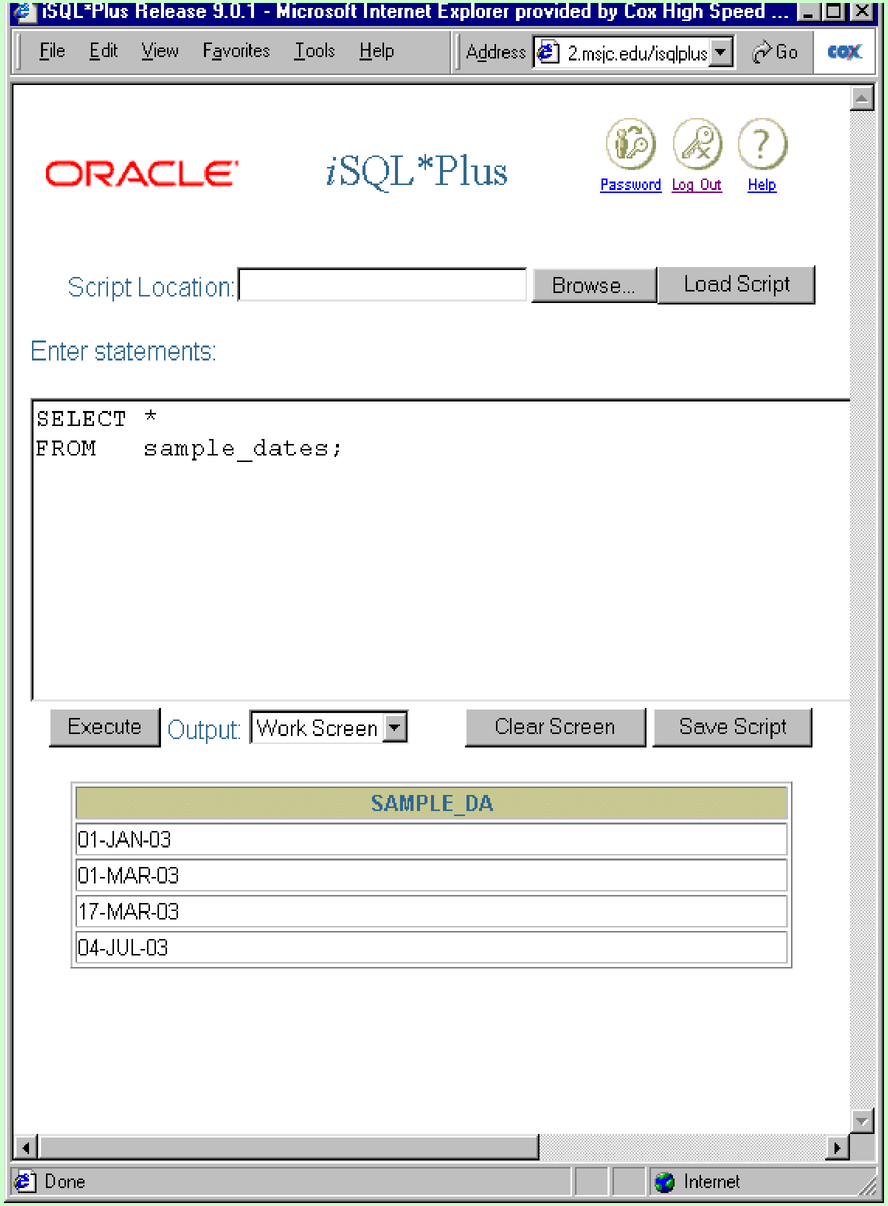
Mostly date math. That is, given a date, what would yesterday's date (or tomorrow's) date be.

Date plus or minus some number of days

Or, given two dates, how many days exist between them?

And these date operations need to be date aware, and should only provide valid values, accounting for the correct number of days per month, as well as leap years.

Page F-3 Date Ops



```
SELECT *
FROM   sample_dates;
```

SAMPLE_DA
01-JAN-03
01-MAR-03
17-MAR-03
04-JUL-03

These date operations are built in to the Oracle database system:

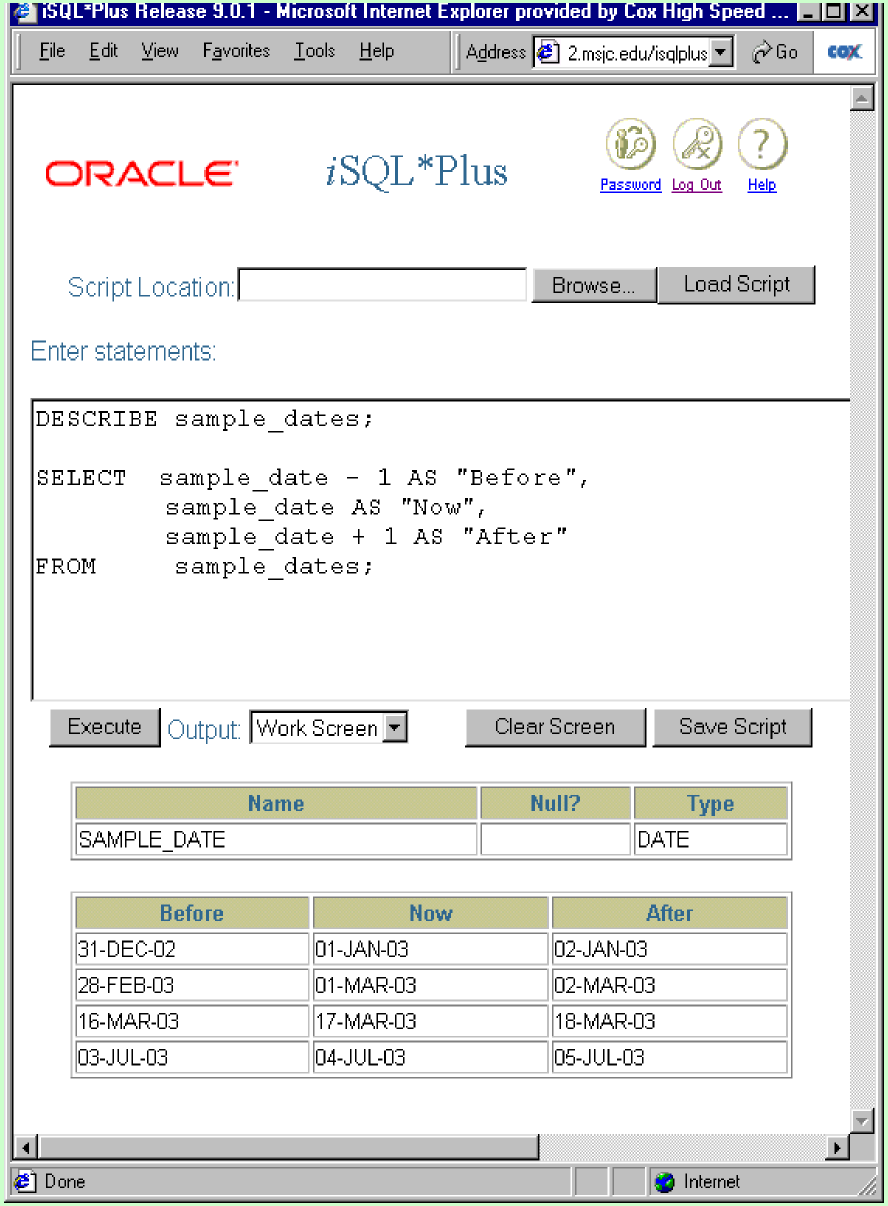
date – number = date

date + number = date

date – date = number of days

Let's give them a try.

(← Here's a table with some sample date values)



```
DESCRIBE sample_dates;

SELECT  sample_date - 1 AS "Before",
        sample_date AS "Now",
        sample_date + 1 AS "After"
FROM     sample_dates;
```

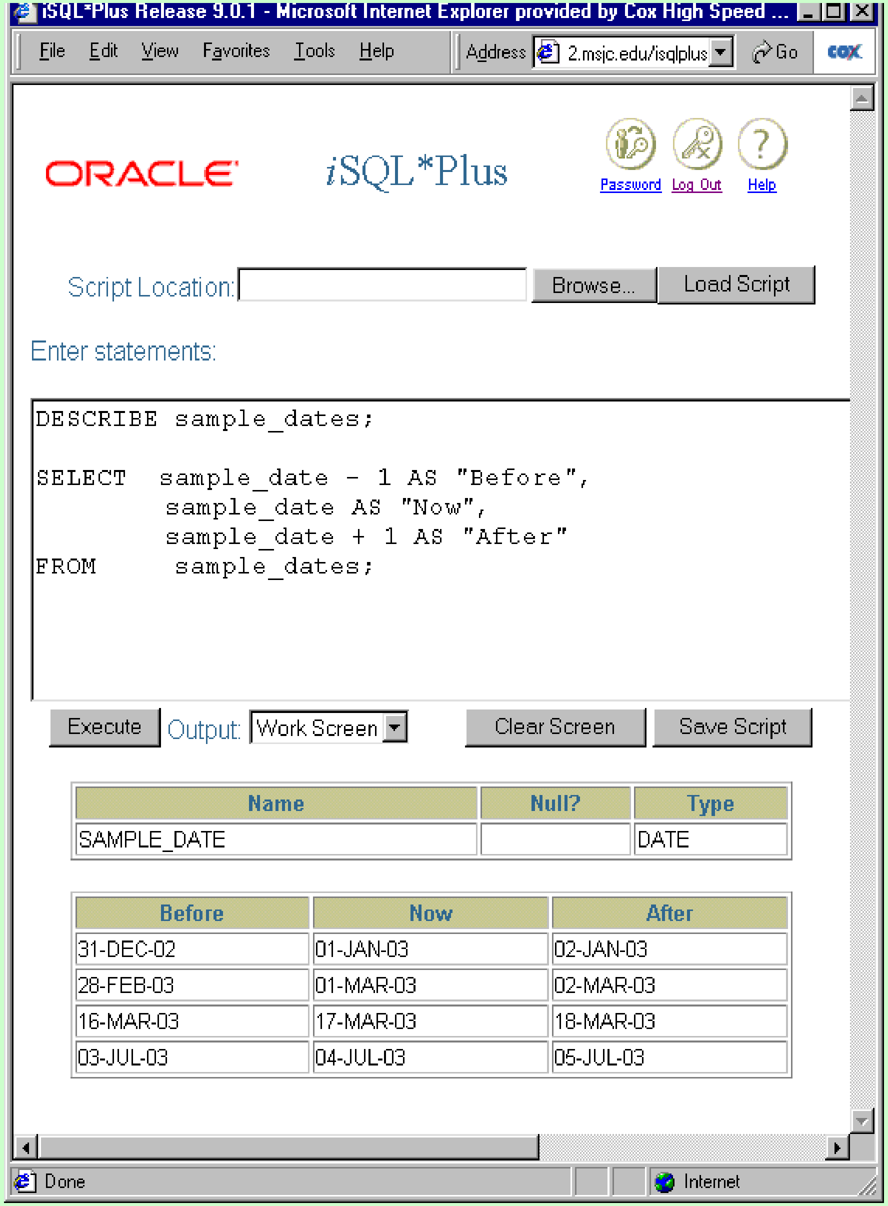
Name	Null?	Type
SAMPLE_DATE		DATE

Before	Now	After
31-DEC-02	01-JAN-03	02-JAN-03
28-FEB-03	01-MAR-03	02-MAR-03
16-MAR-03	17-MAR-03	18-MAR-03
03-JUL-03	04-JUL-03	05-JUL-03

What days precede and follow each of the dates in the table?

```
SELECT   sample_date – 1  AS “Before”,
          sample_date AS “Now”,
          sample_date + 1 AS “After”
FROM     sample_dates;
```

Page F-5 Date Ops Analysis



```
DESCRIBE sample_dates;

SELECT  sample_date - 1 AS "Before",
        sample_date AS "Now",
        sample_date + 1 AS "After"
FROM    sample_dates;
```

Name	Null?	Type
SAMPLE_DATE		DATE

Before	Now	After
31-DEC-02	01-JAN-03	02-JAN-03
28-FEB-03	01-MAR-03	02-MAR-03
16-MAR-03	17-MAR-03	18-MAR-03
03-JUL-03	04-JUL-03	05-JUL-03

These operations are critical features of SQL and are often needed by programmers. It’s a very good thing that Oracle SQL has them.

BUT, ...

One of the things I don’t care for about the Oracle implementation of date operations is the lack of a unit of measurement.

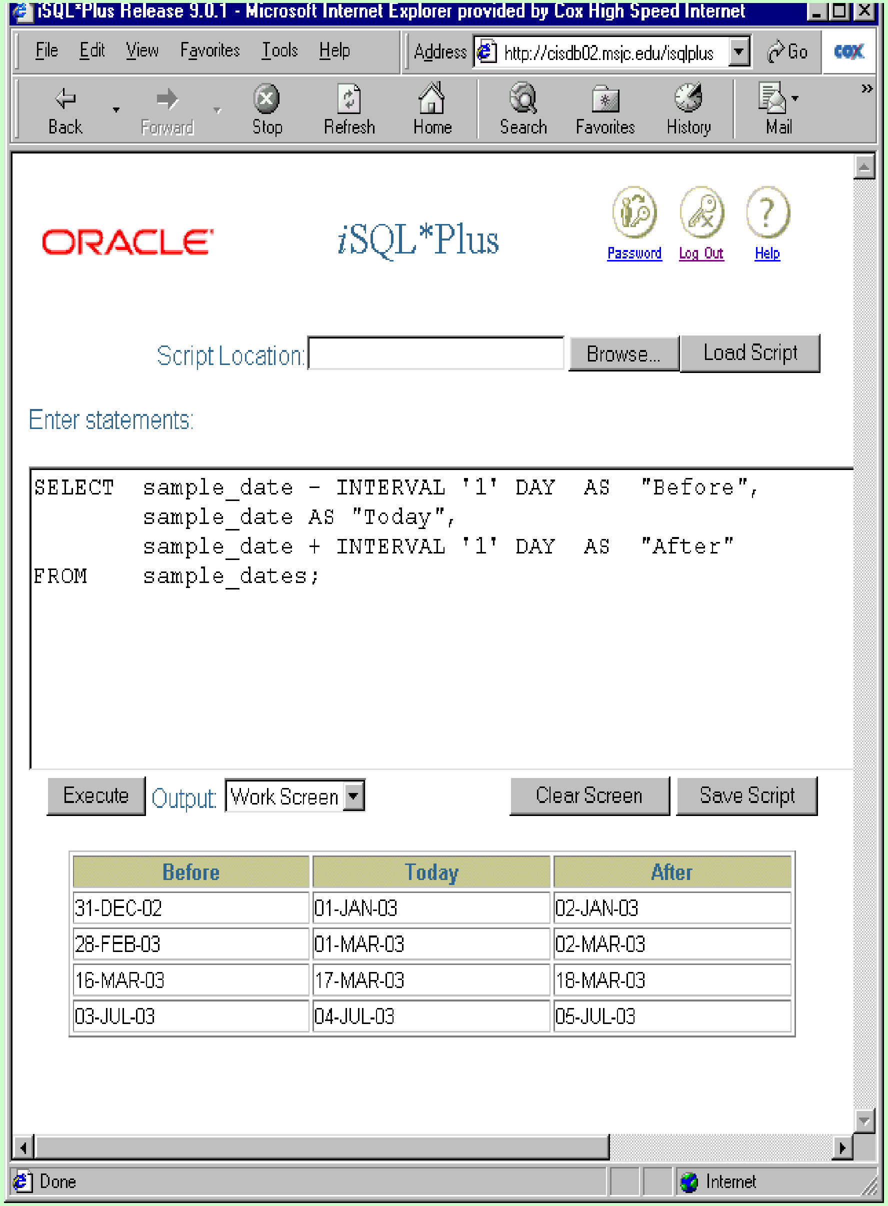
Look at any one of those date values in the result table and you can see three fields:

day
month
year

So, when we add 1 to a date, what really is it that we’re adding? Another day? Another month? Another year?

It seems to me that things would be better all around if we were able to explicitly specify which unit of measurement we’re adding.

Page F-6 Interval - Day


iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Speed Internet
File Edit View Favorites Tools Help
Address http://cisdb02.msjc.edu/isqlplus
Go
Back Forward Stop Refresh Home Search Favorites History Mail
ORACLE
iSQL*Plus
Password Log Out Help
Script Location:
Browse...
Load Script
Enter statements:
Execute
Output: Work Screen
Clear Screen
Save Script
Done
Internet

```
SELECT  sample_date - INTERVAL '1' DAY  AS  "Before",
        sample_date AS "Today",
        sample_date + INTERVAL '1' DAY  AS  "After"
FROM    sample_dates;
```

Before	Today	After
31-DEC-02	01-JAN-03	02-JAN-03
28-FEB-03	01-MAR-03	02-MAR-03
16-MAR-03	17-MAR-03	18-MAR-03
03-JUL-03	04-JUL-03	05-JUL-03

What we really need is the ability to specify the interval of time.

And as database vendors upgrade their products to conform to the newer SQL standards, we're finding that INTERVALS are making their way into the language. Even Oracle ☺

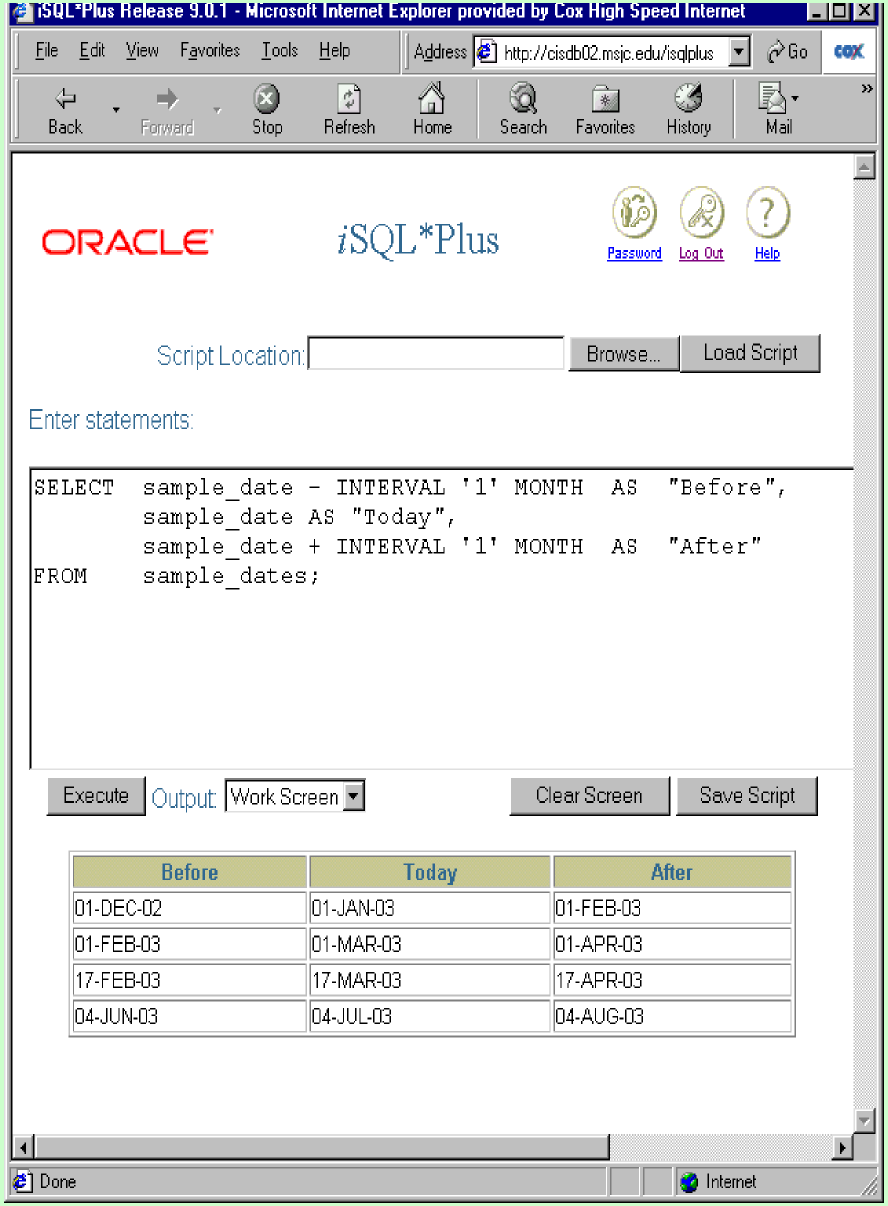
Now don't get me wrong. It really was great of Oracle to provide these capabilities to the programmer, in the sense that something is better than nothing.

But the standard now defines INTERVALS, so you, as a SQL programmer should start writing your programs to take advantage of the features that are common across all SQL implementations, rather than writing code based on proprietary features offered by a single vendor.

The standard now provides for these operations:

Date – date = interval
Date + interval = date
Interval + interval = interval

Page F-7 Interval - Months


iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Speed Internet
File Edit View Favorites Tools Help
Address http://cisdb02.msjc.edu/isqlplus
Go
Back Forward Stop Refresh Home Search Favorites History Mail
ORACLE
iSQL*Plus
Password Log Out Help
Script Location:
Browse...
Load Script
Enter statements:
Execute
Output: Work Screen
Clear Screen
Save Script
Done
Internet

```
SELECT  sample_date - INTERVAL '1' MONTH  AS  "Before",
        sample_date AS "Today",
        sample_date + INTERVAL '1' MONTH  AS  "After"
FROM    sample_dates;
```

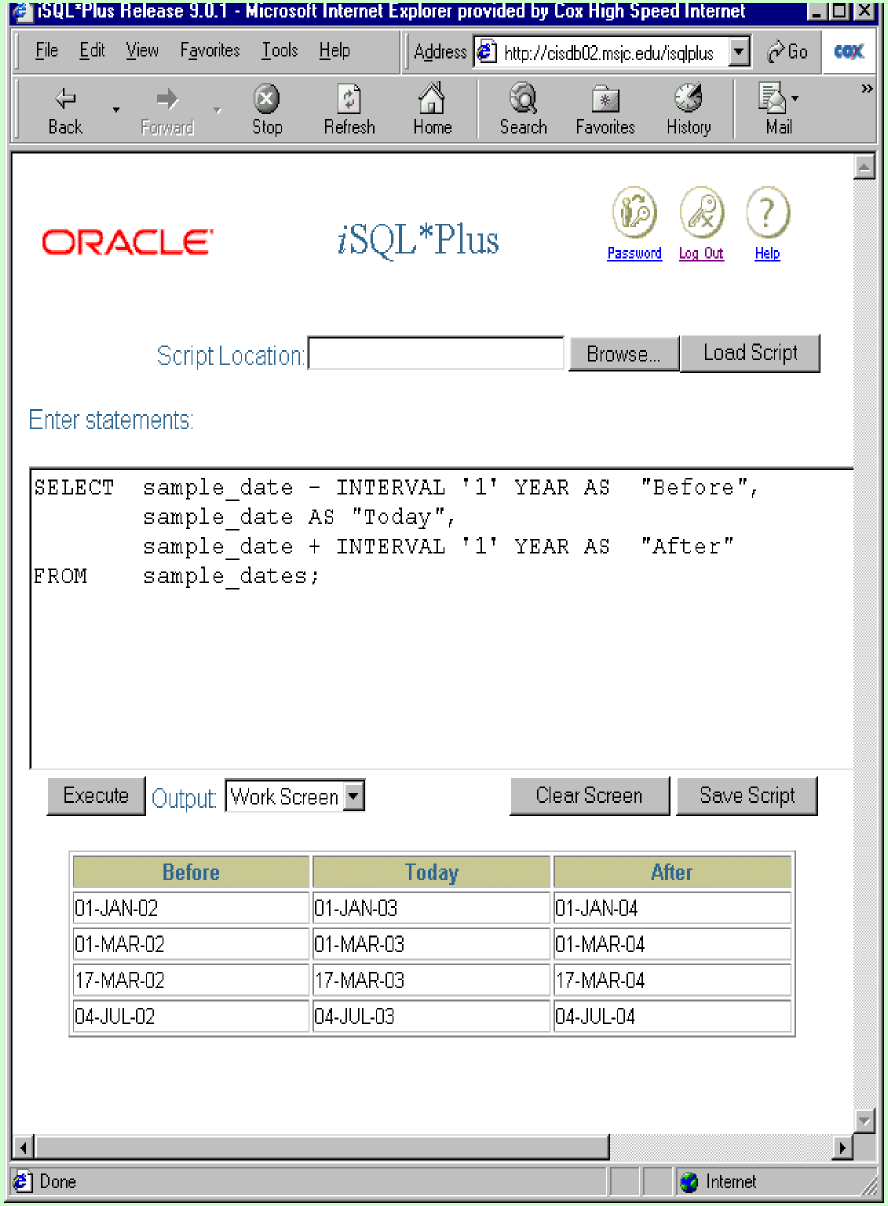
Before	Today	After
01-DEC-02	01-JAN-03	01-FEB-03
01-FEB-03	01-MAR-03	01-APR-03
17-FEB-03	17-MAR-03	17-APR-03
04-JUN-03	04-JUL-03	04-AUG-03

Intervals of MONTH units.

Page F-8 Date Difference

We'll defer our discussion of difference (date – date = interval) until a later time.

Page F-9 Interval - Year


iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Speed Internet
File Edit View Favorites Tools Help
Address http://cisdb02.msjc.edu/isqlplus
Go
Back Forward Stop Refresh Home Search Favorites History Mail
ORACLE
iSQL*Plus
Password Log Out Help
Script Location:
Browse...
Load Script
Enter statements:
Execute
Output: Work Screen
Clear Screen
Save Script
Done
Internet

```
SELECT  sample_date - INTERVAL '1' YEAR AS  "Before",
        sample_date AS "Today",
        sample_date + INTERVAL '1' YEAR AS  "After"
FROM    sample_dates;
```

Before	Today	After
01-JAN-02	01-JAN-03	01-JAN-04
01-MAR-02	01-MAR-03	01-MAR-04
17-MAR-02	17-MAR-03	17-MAR-04
04-JUL-02	04-JUL-03	04-JUL-04

Intervals of YEAR units.

Page S-1 Operations Summary

Numeric operations

Operation	Operator
Addition	+
Subtraction	-
Multiplication	*
Division	/

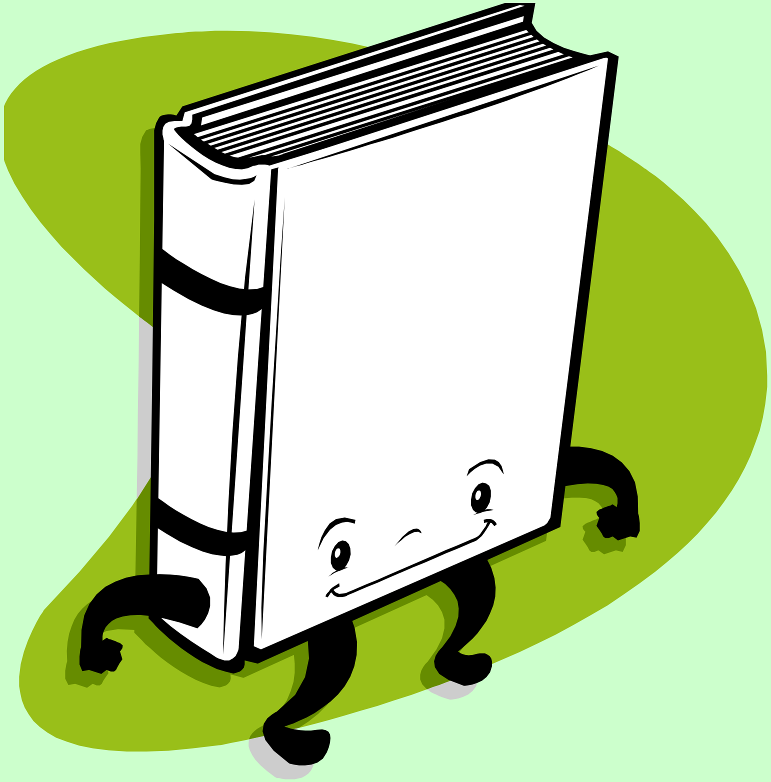
Character operations

Operation	Operator
Concatenation	||

Date operations

Operation	Operator
Addition	+ INTERVAL ‘value’ UNIT
Subtraction	- INTERVAL ‘value’ UNIT
Difference	

Module 05: Operations



Page T-1: Terminology

Operations

Numeric operations

addition operation, addition operator, addition symbol, +

subtraction operation, subtraction operator, subtraction symbol, -

multiplication operation, multiplication operator, multiplication symbol, *

division operation, division operator, division symbol, /

Alphanumeric operations, character operations

Concatenation operation, concatenation operator, concatenation symbol, ||

Date operations

Date arithmetic

Precedence of operations

Test your designs, test your code

DUAL

unit of measurement

Interval

Page Z-1: End Notes

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