SQL Programming

Value Functions

Page A-1: Intro

The mathematical, character, and datetime operations that we've studied have been pretty simple.

We've done addition, subtraction, multiplication, division, and concatenation by using the built-in operators that support these operations.

And now the question is, how do we go about performing more complex operations, for which there probably aren't operators?

The solution will be to use ***functions***. Functions are generally accessible with a keyword (ie. The name of the function). In this regard, we might say that functions are similar to the comparison verbs (LIKE, and BETWEEN) that we've used.

Page B-1: Function types

The first thing you need to learn about functions is that every function ‘returns a value’, and a function is defined by the type of value that it returns.

A character function returns a single character value.

A numeric function returns a single numeric value.

A datetime function returns a single datetime value.

A character function may be used anywhere in your program that a character value may be used.

A numeric function may be used anywhere in your program that a numeric value is permitted.

And datetime functions may be used anywhere in a SQL program that a datetime value can be used.

Page B-2: Function Names

One popular character function is the UPPERCASE function. This function takes a character value and transforms it into its uppercase equivalent.

The name of this function is UPPER, and just as we saw with character literals, this, and all character functions, may be used anywhere in the program that permits a character value. For example:

```
SELECT  upper(last_name)
FROM     talent;
```

```
SELECT  last_name, first_name
FROM     talent
WHERE  upper(last_name) = ‘FORD’;
```

Page B-3: Function Parameters

Did you notice that there's a set of parentheses immediately after the function name surrounding a column name? This column name is a parameter to the function, and in general, each function call includes a list of parameters.

This parameter list can be thought of as the list of ingredients that are needed by the function to perform its task.

In the preceding example,

```
SELECT  upper(last_name)
FROM     talent;
```

the parameter list was simply a single character value, last_name.

When referencing, or calling a function then, the programmer needs to specify two things:

Name of the function, and
Parameter list (aka function list)

The parameter list is enclosed in a pair of parentheses, and items are separated from one another by commas.

Page B-4: Function Type vs/ Parameter Type

Now here's an important point.

Functions are defined by the data type they return (character functions return character values).

Functions are NOT defined by the data type of the arguments in the parameter list.

This is a critical distinction and some textbook authors, and some vendor manuals (eg. Oracle) get it wrong.

Functions are defined by the data type they return (eg. character functions return character values).

If you can learn and use this definition for functions, you'll have a much easier time in any other programming class (Java, C++, Visual BASIC, ...)

Page B-5: Parameter List

More often than not, the parameter list requires only a single parameter. However there are some functions that require 2, 3, 4, or more items in their parameter list. And on rare occasion, you might find a function that doesn't need any parameters.

In the case of a multi-parameter list (also known as a multi-argument list) you might find that the required arguments are drawn from a variety of data types.

In this regard, since the argument list might include different data types, doesn't it make sense that we define the function based on the data type of the single value that it returns, rather than arbitrarily selecting the data type of one of the arguments???

Page B-6: Summary

The general form of a function call is:

Function-name(parameter$_1$, parameter$_2$, …parameter$_n$)

Page C-1: Character Functions

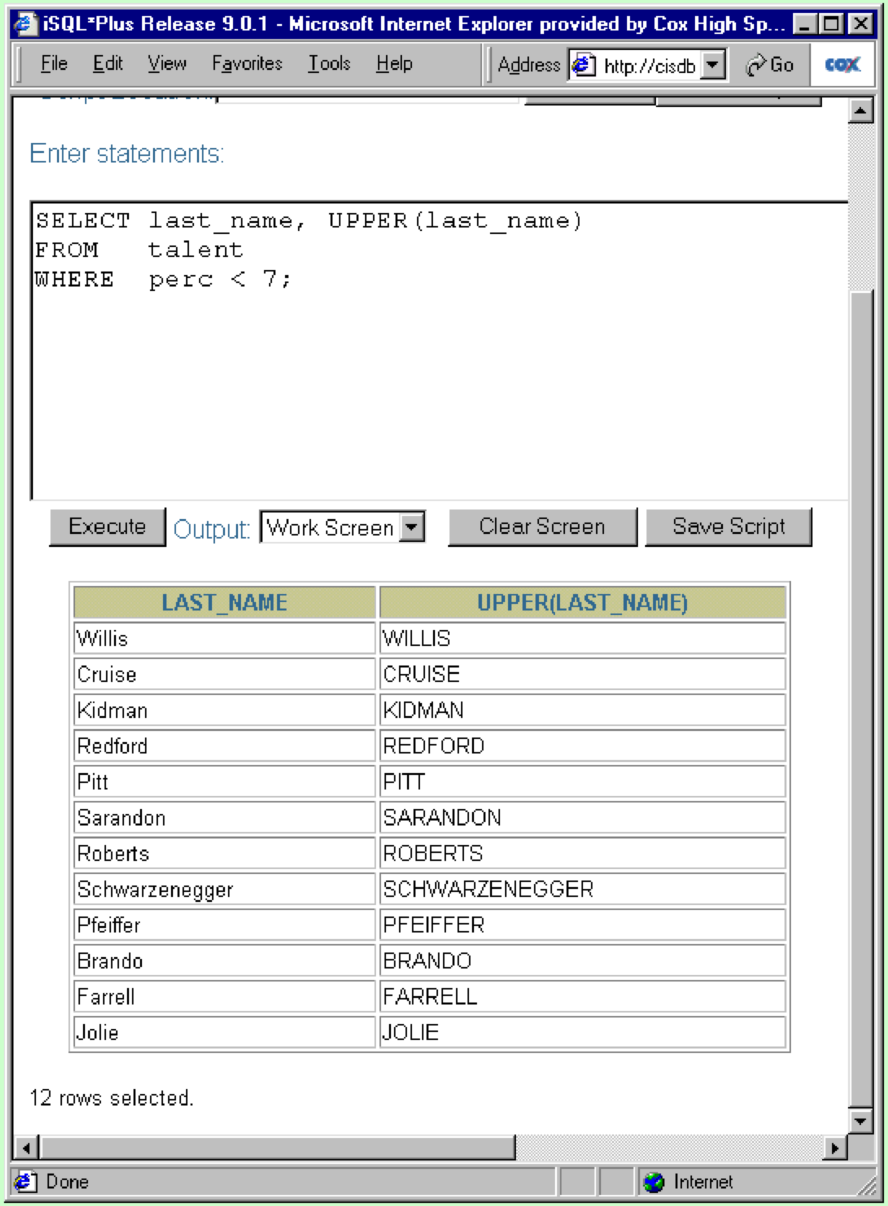
A character function returns a single character value.

A character function may be used anywhere in your program that a character value may be used.

The character value functions that we'll review in this module include:

UPPER
LOWER
TRIM

Page C-2: UPPER Function



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

```
SELECT last_name, UPPER(last_name)
FROM   talent
WHERE  perc < 7;
```

LAST_NAME	UPPER(LAST_NAME)
Willis	WILLIS
Cruise	CRUISE
Kidman	KIDMAN
Redford	REDFORD
Pitt	PITT
Sarandon	SARANDON
Roberts	ROBERTS
Schwarzenegger	SCHWARZENEGGER
Pfeiffer	PFEIFFER
Brando	BRANDO
Farrell	FARRELL
Jolie	JOLIE

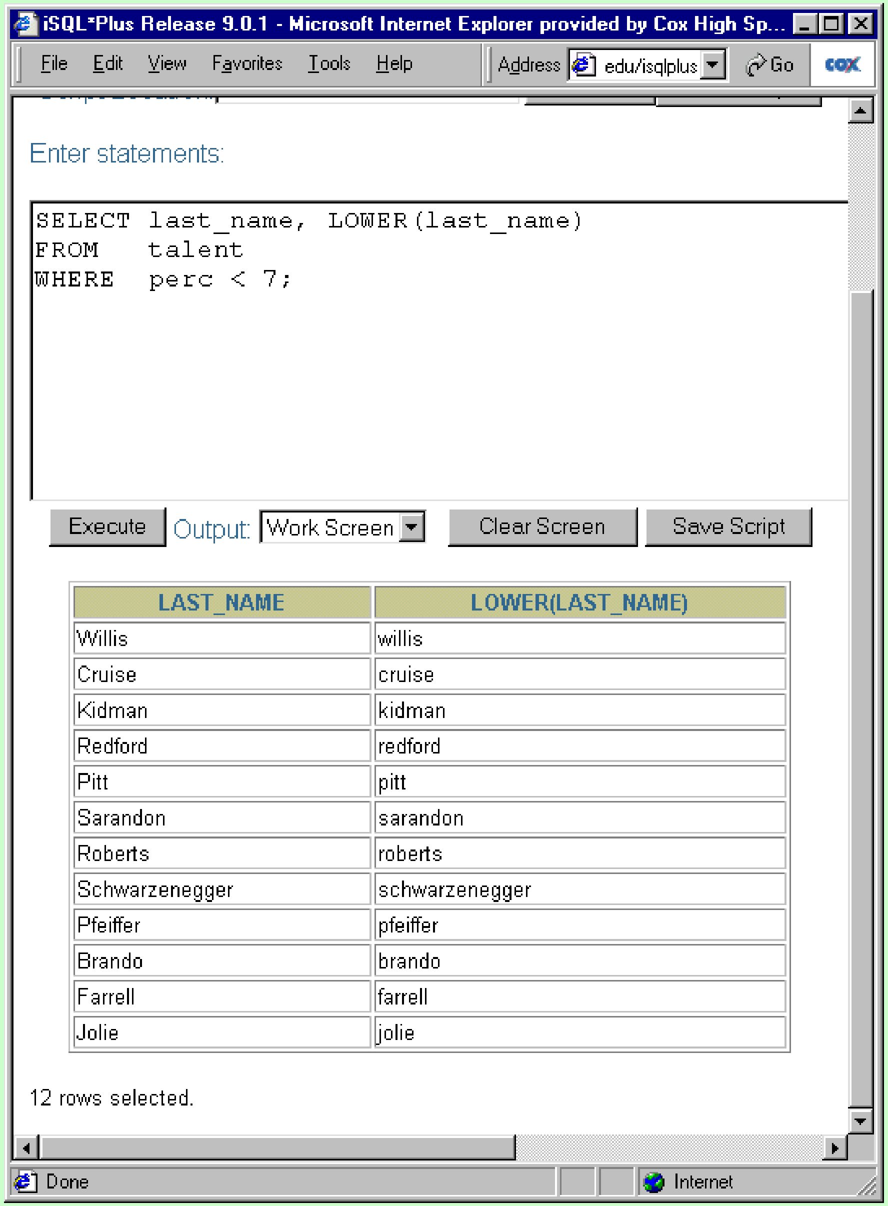
12 rows selected.

The UPPER function transforms a string value into its uppercase equivalent.

UPPER(string value)

Module 09: Value Functions

Page C-3: LOWER Function


iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Sp...
File Edit View Favorites Tools Help
Address edu/isqlplus
Go

Enter statements:

```
SELECT last_name, LOWER(last_name)
FROM   talent
WHERE  perc < 7;
```

Execute
Output: Work Screen
Clear Screen
Save Script

LAST_NAME	LOWER(LAST_NAME)
Willis	willis
Cruise	cruise
Kidman	kidman
Redford	redford
Pitt	pitt
Sarandon	sarandon
Roberts	roberts
Schwarzenegger	schwarzenegger
Pfeiffer	pfeiffer
Brando	brando
Farrell	farrell
Jolie	jolie

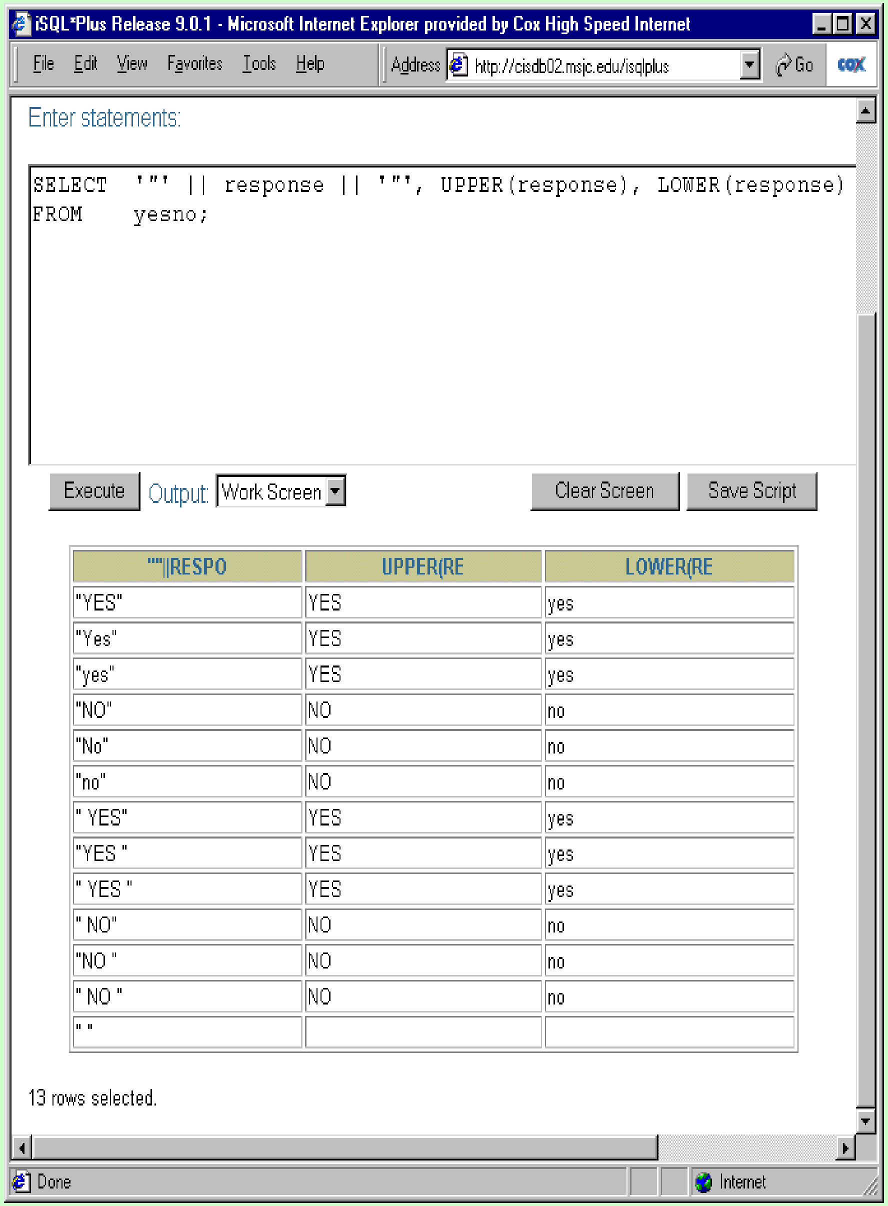
12 rows selected.

The LOWER function converts a string value into its lowercase equivalent,

LOWER(string value)

Module 09: Value Functions

Page C-4: String Functions


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  '"' || response || '"', UPPER(response), LOWER(response)
FROM    yesno;
```

""||RESPO	UPPER(RE	LOWER(RE
"YES"	YES	yes
"Yes"	YES	yes
"yes"	YES	yes
"NO"	NO	no
"No"	NO	no
"no"	NO	no
" YES"	YES	yes
"YES "	YES	yes
" YES "	YES	yes
" NO"	NO	no
"NO "	NO	no
" NO "	NO	no
" "		

13 rows selected.

These functions are particularly useful to the programmer when the data columns contain mixed case values.

Imagine the compound condition that you would have to write if you wanted to test for a 'yes' value in the response column.

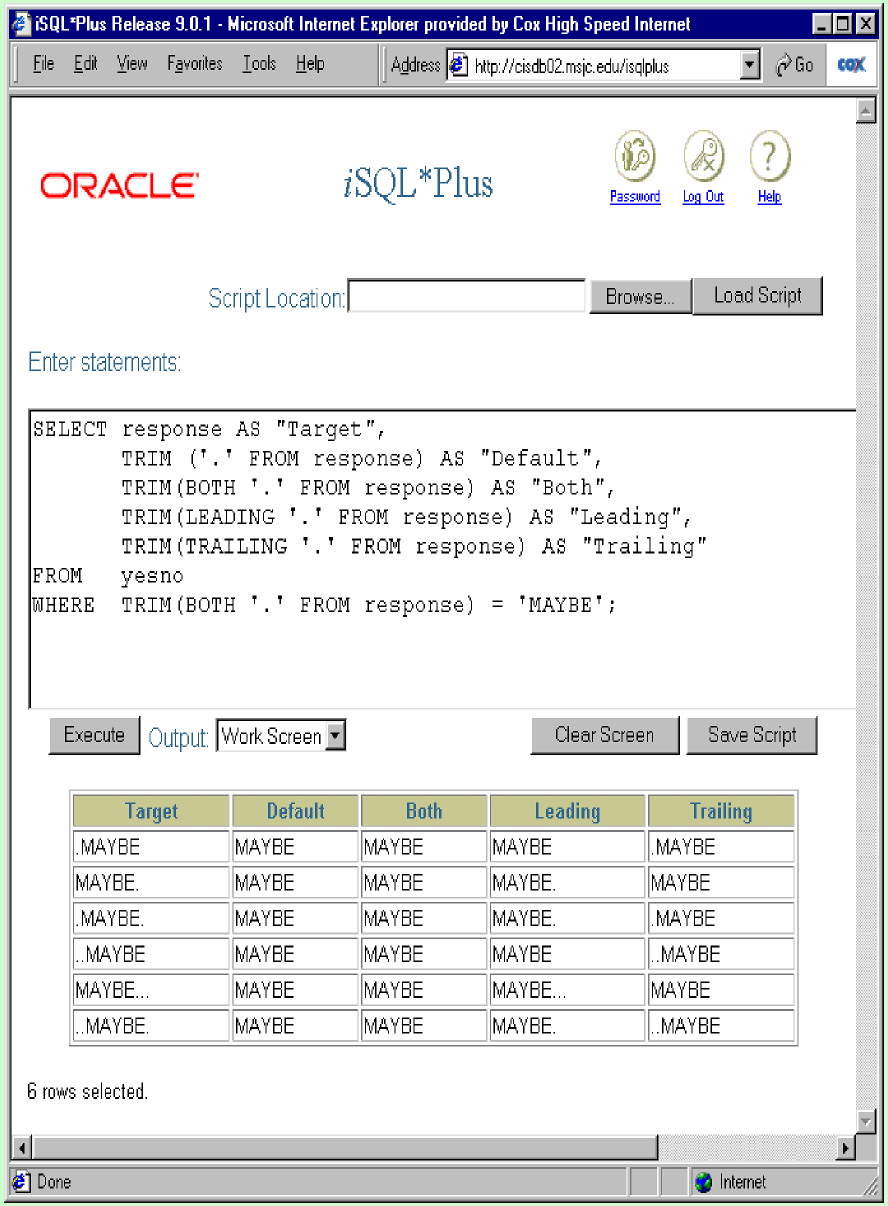
```
WHERE response = 'YES'
      OR response = 'yes'
      OR response = 'Yes'
      OR  ....
```

Rather than writing code to test for all possible values, first convert those values to either upper- or lower-case and write your condition against that value. For example:

```
WHERE  UPPER(response) = 'YES'
```

Module 09: Value Functions

Page C-2: TRIM Function


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
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 response AS "Target",
       TRIM ('.' FROM response) AS "Default",
       TRIM(BOTH '.' FROM response) AS "Both",
       TRIM(LEADING '.' FROM response) AS "Leading",
       TRIM(TRAILING '.' FROM response) AS "Trailing"
FROM   yesno
WHERE  TRIM(BOTH '.' FROM response) = 'MAYBE';
```

Target	Default	Both	Leading	Trailing
.MAYBE	MAYBE	MAYBE	MAYBE	.MAYBE
MAYBE.	MAYBE	MAYBE	MAYBE.	MAYBE
.MAYBE.	MAYBE	MAYBE	MAYBE.	.MAYBE
..MAYBE	MAYBE	MAYBE	MAYBE	..MAYBE
MAYBE...	MAYBE	MAYBE	MAYBE...	MAYBE
..MAYBE.	MAYBE	MAYBE	MAYBE.	..MAYBE

6 rows selected.

The TRIM function is used to remove, or trim, ‘extraneous’ characters from a string.

TRIM may be used to remove leading or trailing spaces.

TRIM (character value FROM character value)
TRIM (LEADING character value FROM character value)
TRIM (TRAILING character value FROM character value)
TRIM (BOTH character value FROM character value)

Be careful here. The FROM phrase in the TRIM clause should not be confused with the FROM clause in a SELECT statement. In my opinion it’s unfortunate that SQL uses the word FROM in both of these places.

TRIM is most frequently used to remove spaces and tabs from character values.

Page D-1: Numeric Functions

A numeric function returns a single numeric value.

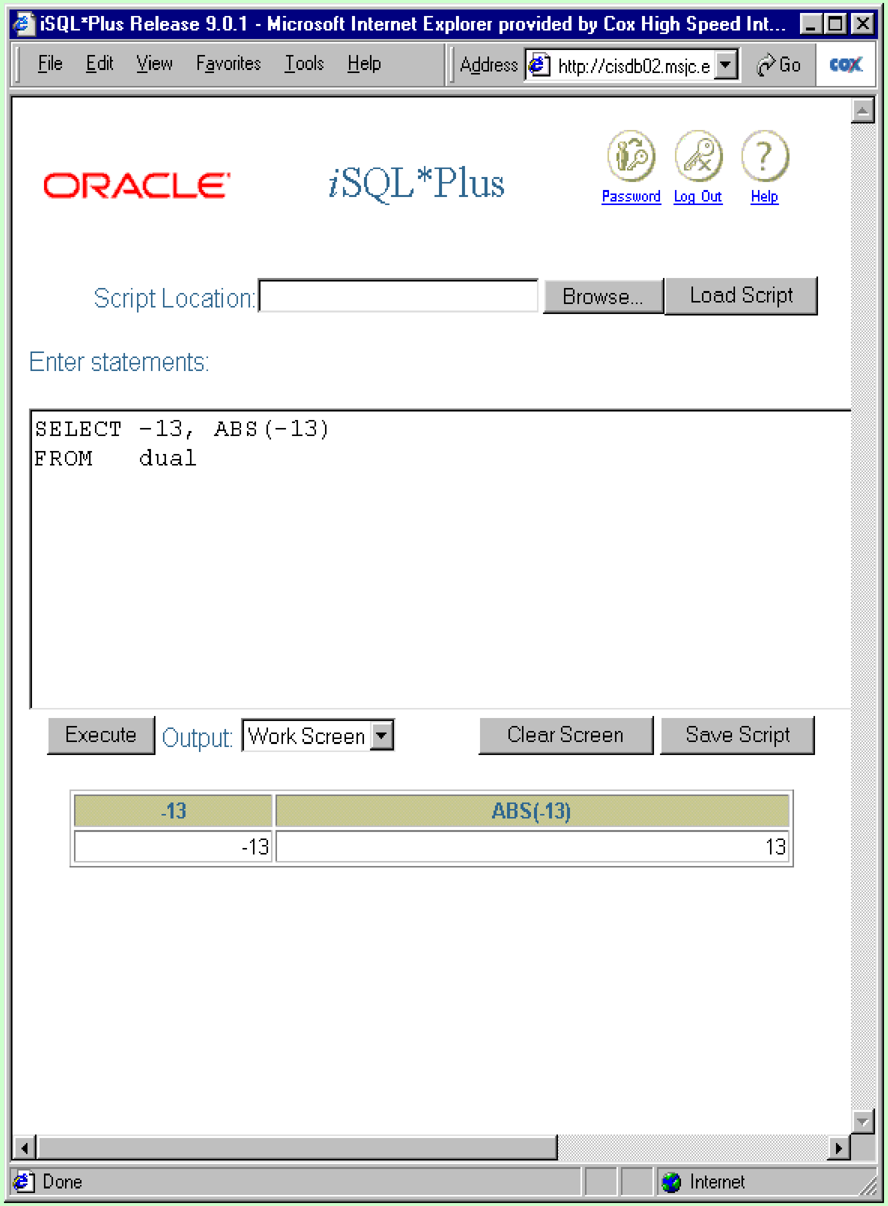
A numeric function may be used anywhere in your program that a numeric value may be used.

The two numeric value functions that we’ll look at in this module are:

ABS

MOD

Page D-2: ABS Function


iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Speed Int...
File Edit View Favorites Tools Help
Address http://cisdb02.msjc.e
Go
ORACLE
iSQL*Plus
Password
Log Out
Help
Script Location:
Browse...
Load Script
Enter statements:

```
SELECT -13, ABS(-13)
FROM   dual
```

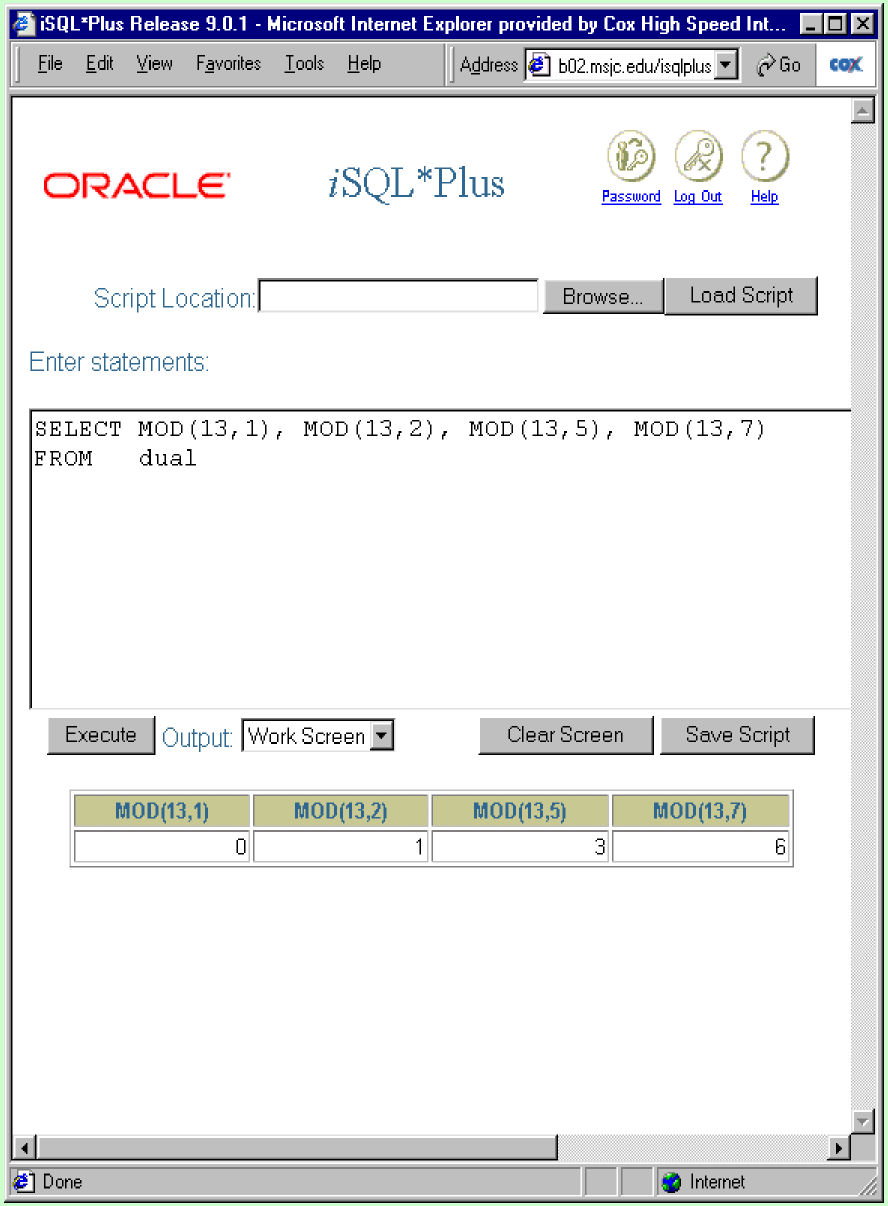
Execute
Output: Work Screen
Clear Screen
Save Script

-13	ABS(-13)
-13	13

The ABS function returns the absolute value of a numeric value. The absolute value of a number is its distance from zero, hence the absolute value of any number is always a positive value.

ABS(numeric value)

Page D-3: MOD Function


iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Speed Int...
File Edit View Favorites Tools Help
Address b02.msjc.edu/isqlplus
Go
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 MOD(13,1), MOD(13,2), MOD(13,5), MOD(13,7)
FROM   dual
```

MOD(13,1)	MOD(13,2)	MOD(13,5)	MOD(13,7)
0	1	3	6

The MOD function is also referred to as the *remainder* function.

In integer math, the number 5, divided by 2, results in 2, with a remainder of 1.

MOD returns the remainder value in integer division. The general form of the function call is:

MOD(integer value, integer value)

Notice that the parameters that are used by the MOD function are not simply numeric values, but are a more limited subset of the class of numbers -> integers.

Or to say that again in English, the parameters that you use in the mod function have to be whole numbers.

Module 09: Value Functions　Page D-4: Numeric Functions

Page E-1: Datetime Functions

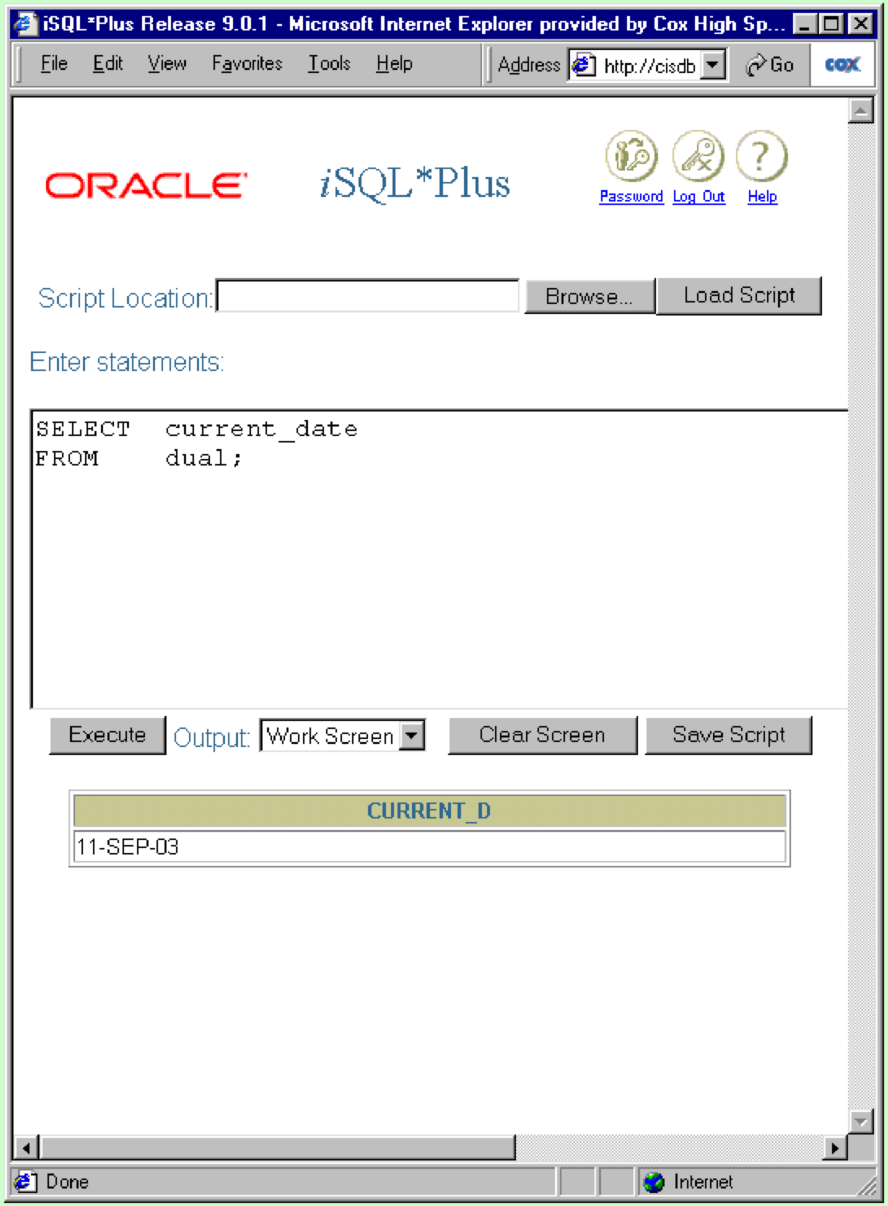
A datetime function returns a single date value.

A datetime function may be used anywhere in your program that a date value may be used.

The datetime functions that we'll review in this module include:

CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
LOCALTIME
LOCALTIMESTAMP

Module 09: Value Functions Page E-2: CURRENT_DATE Function



```
SELECT   current_date
FROM     dual;
```

CURRENT_D
11-SEP-03

CURRENT_DATE returns the current date.

This particular function does not accept any parameters.

CURRENT_DATE

©1998-2018 / Bergin

Page E-3: CURRENT_TIME Function

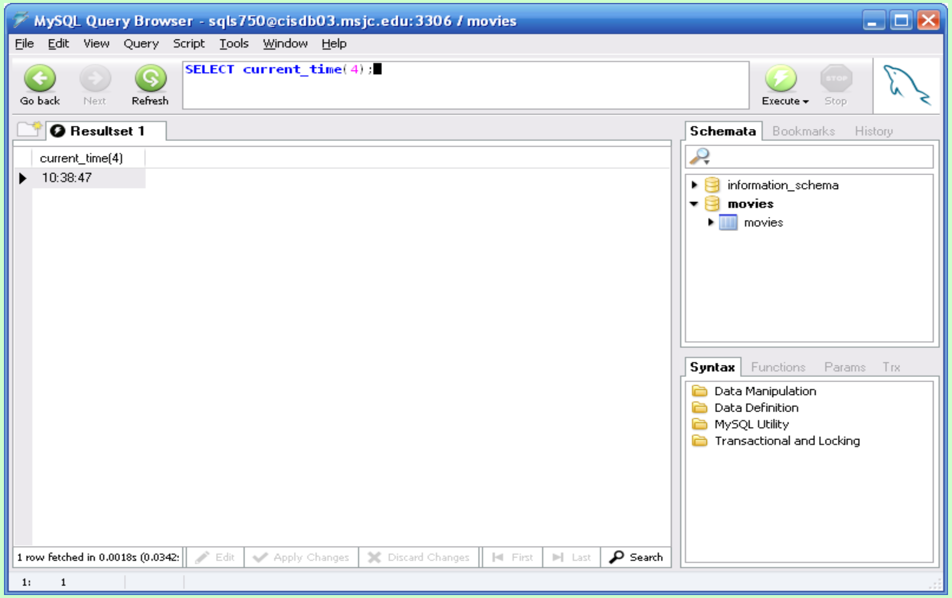
CURRENT_TIME returns the current time.

CURRENT_TIME
CURRENT_TIME(precision)

Where precision specifies the number of decimal places to be displayed.

CURRENT_TIME is not available in Oracle.

And although it is available in MySQL, the precision argument seems to be ignored.

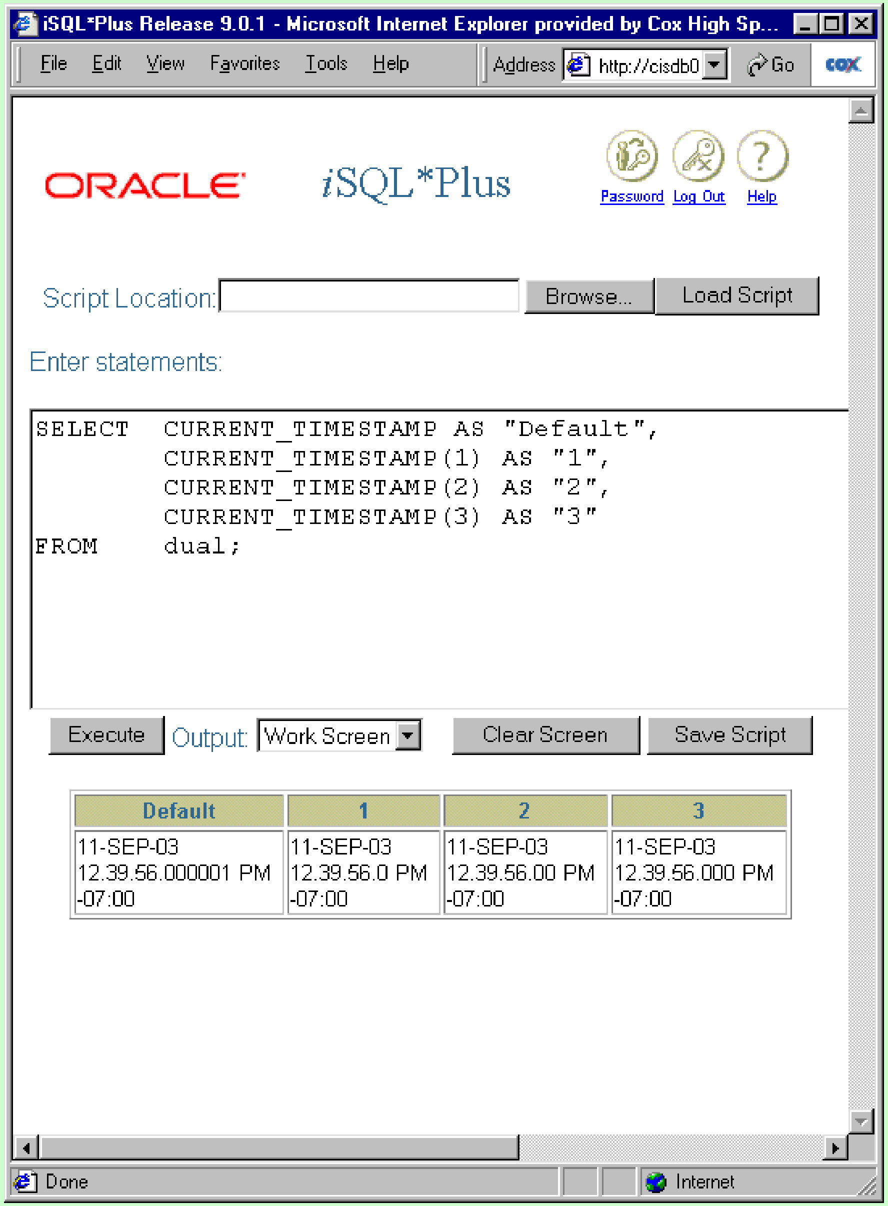


```
SELECT current_time( 4 );
```

```
current_time(4)
10:38:47
```

1 row fetched in 0.0018s (0.0342:

Page E-4: CURRENT_TIMESTAMP Function



```
SELECT   CURRENT_TIMESTAMP AS "Default",
         CURRENT_TIMESTAMP(1) AS "1",
         CURRENT_TIMESTAMP(2) AS "2",
         CURRENT_TIMESTAMP(3) AS "3"
FROM     dual;
```

Default	1	2	3
11-SEP-03 12.39.56.000001 PM -07:00	11-SEP-03 12.39.56.0 PM -07:00	11-SEP-03 12.39.56.00 PM -07:00	11-SEP-03 12.39.56.000 PM -07:00

The CURRENT_TIMESTAMP function returns a timestamp, that is, the combination of today’s date, and today's time.

CURRENT_TIMESTAMP

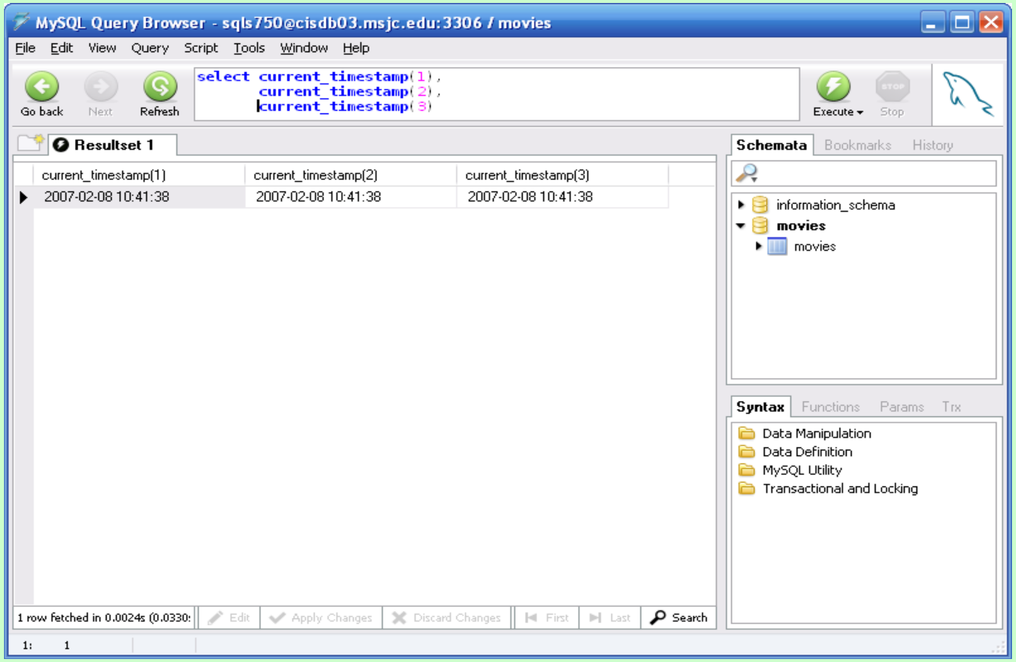
CURRENT_TIMESTAMP(precision)

It also provides an offset value that specifies the timezone relative to the Greenwich meridian.

Module 09: Value Functions

Page E-5: CURRENT_TIMESTAMP Function

Once again, do note that the precision argument seems to be ignored in MySQL, and the timezone offset is not displayed.

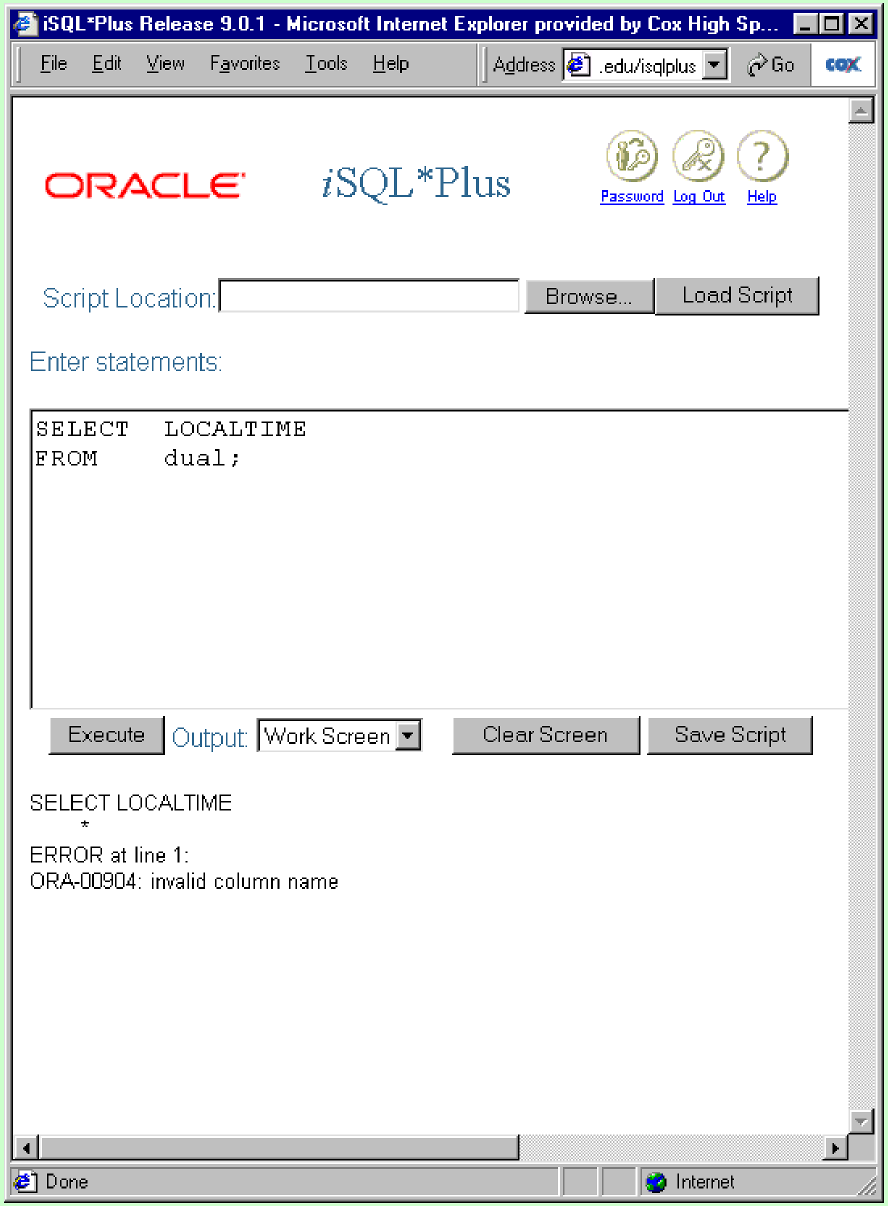


```
select current_timestamp(1),
       current_timestamp(2),
       current_timestamp(3)
```

current_timestamp(1)	current_timestamp(2)	current_timestamp(3)
2007-02-08 10:41:38	2007-02-08 10:41:38	2007-02-08 10:41:38

1 row fetched in 0.0024s (0.0330:

Page E-6: LOCALTIME Function



```
SELECT   LOCALTIME
FROM     dual;
```

```
SELECT LOCALTIME
       *
ERROR at line 1:
ORA-00904: invalid column name
```

LOCALTIME returns the time in the local time zone.

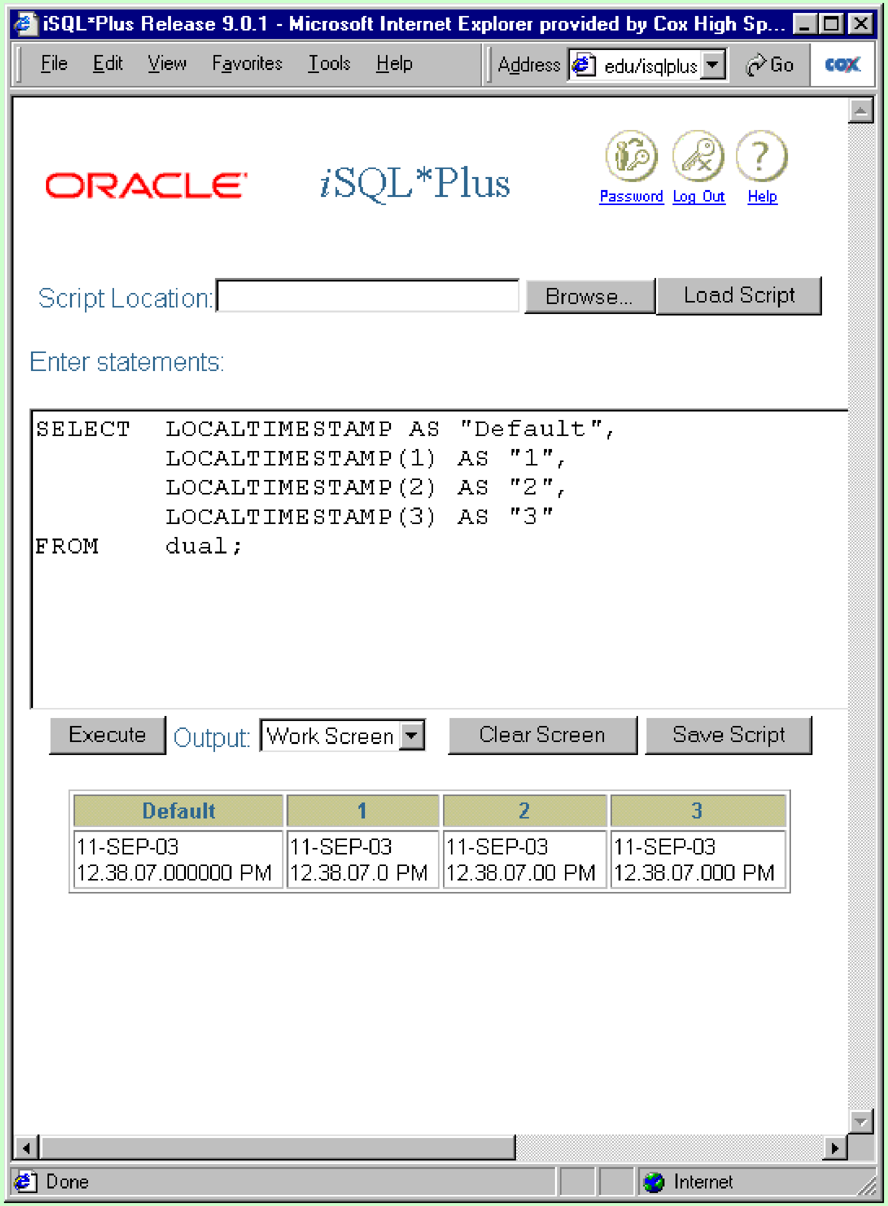
This feature of the SQL:1999 standard is not supported by Oracle.

LOCALTIME

LOCALTIME(precision)

Since these values are to be expressed in 'local time' (ie within the local time zone) the time zone offset value is not displayed

Page E-7: LOCALTIMESTAMP Function



```
SELECT   LOCALTIMESTAMP AS "Default",
         LOCALTIMESTAMP(1) AS "1",
         LOCALTIMESTAMP(2) AS "2",
         LOCALTIMESTAMP(3) AS "3"
FROM     dual;
```

Default	1	2	3
11-SEP-03 12.38.07.000000 PM	11-SEP-03 12.38.07.0 PM	11-SEP-03 12.38.07.00 PM	11-SEP-03 12.38.07.000 PM

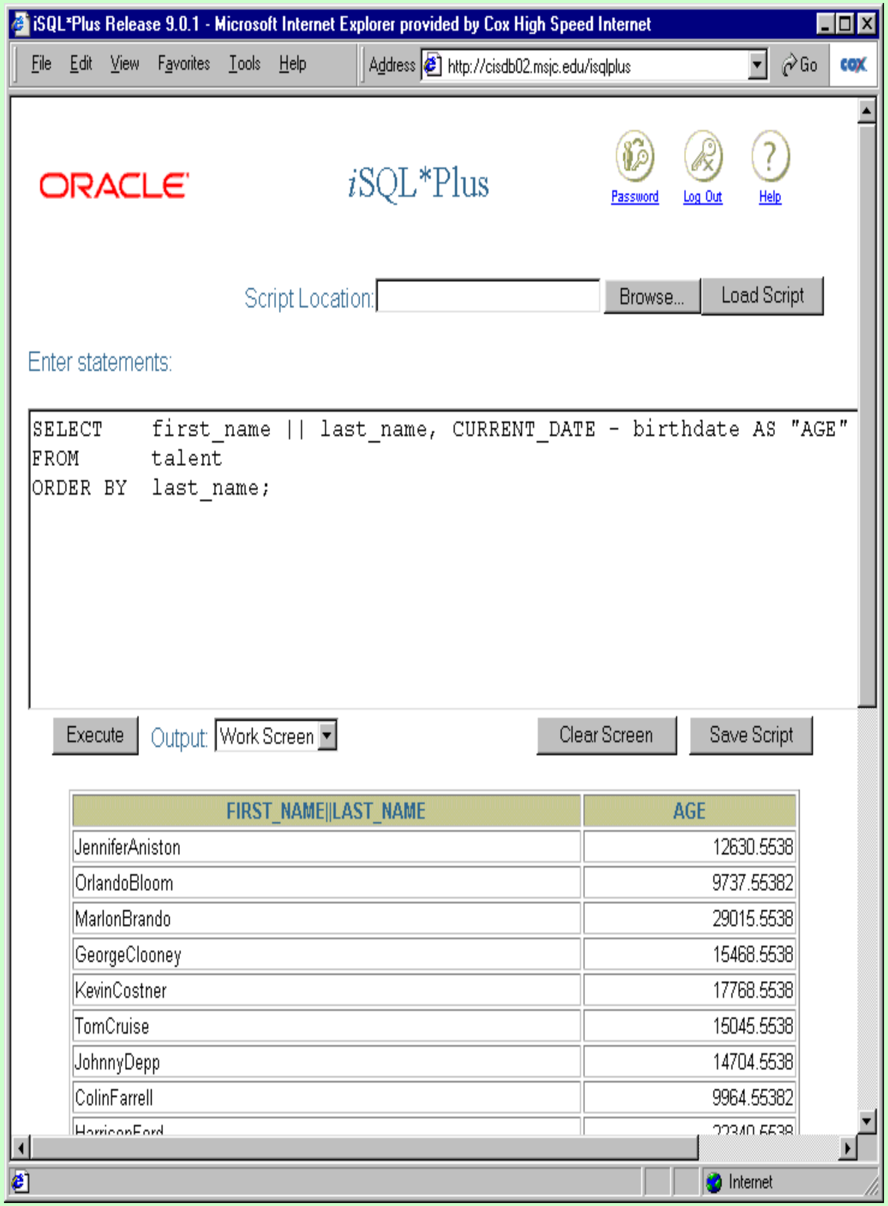
LOCALTIMESTAMP returns the timestamp showing today's date, and the time now.

LOCALTIMESTAMP

LOCALTIMESTAMP(precision)

As with LOCALTIME, since this references a value within the local time zone, there is no need to display the timezone offset value.

Page E-8: Date Functions


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
COX
ORACLE
iSQL*Plus
Password
Log Out
Help
Script Location:
Browse...
Load Script
Enter statements:
Execute
Output:
Work Screen
Clear Screen
Save Script
Internet

```
SELECT    first_name || last_name, CURRENT_DATE - birthdate AS "AGE"
FROM      talent
ORDER BY  last_name;
```

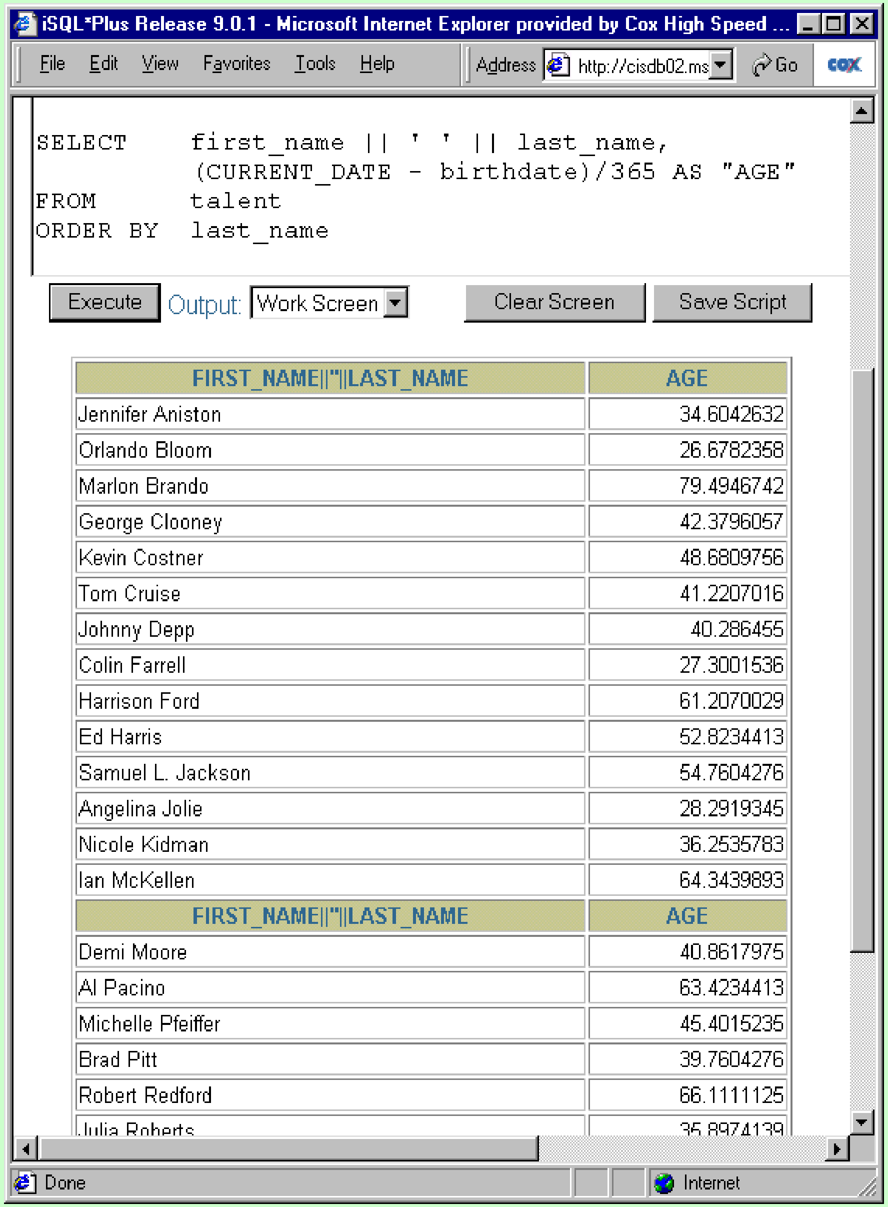
FIRST_NAME||LAST_NAME	AGE
JenniferAniston	12630.5538
OrlandoBloom	9737.55382
MarlonBrando	29015.5538
GeorgeClooney	15468.5538
KevinCostner	17768.5538
TomCruise	15045.5538
JohnnyDepp	14704.5538
ColinFarrell	9964.55382
HarrisonFord	22340.5538

The datatime functions that we've just reviewed are commonly used in date operations.

For example, to calculate someone's age, given their birthdate, subtract their birthdate from today's date.

```
SELECT   CURRENT_DATE – birthdate AS "AGE"
FROM      talent'
```

Module 09: Value Functions   Page E-9: Date Functions revised


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

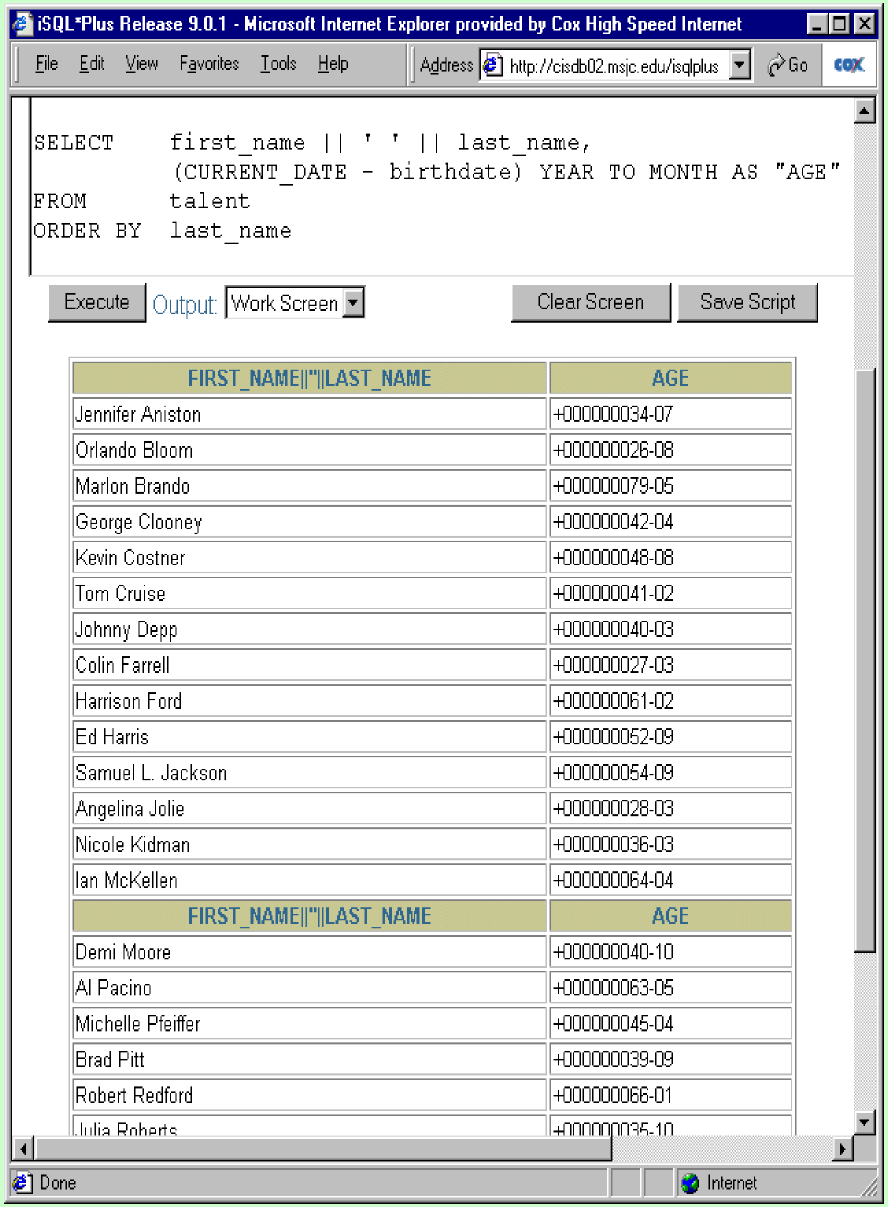
```
SELECT    first_name || ' ' || last_name,
          (CURRENT_DATE - birthdate)/365 AS "AGE"
FROM      talent
ORDER BY  last_name
```

FIRST_NAME||''||LAST_NAME	AGE
Jennifer Aniston	34.6042632
Orlando Bloom	26.6782358
Marlon Brando	79.4946742
George Clooney	42.3796057
Kevin Costner	48.6809756
Tom Cruise	41.2207016
Johnny Depp	40.286455
Colin Farrell	27.3001536
Harrison Ford	61.2070029
Ed Harris	52.8234413
Samuel L. Jackson	54.7604276
Angelina Jolie	28.2919345
Nicole Kidman	36.2535783
Ian McKellen	64.3439893
FIRST_NAME||''||LAST_NAME	AGE
Demi Moore	40.8617975
Al Pacino	63.4234413
Michelle Pfeiffer	45.4015235
Brad Pitt	39.7604276
Robert Redford	66.1111125
Julia Roberts	35.8974139

I think it might be better to show the age in years:

```
SELECT   first_name || ' ' || last_name,
         (CURRENT_DATE – birthdate)/365 AS "AGE"
FROM     talent
ORDER BY  last_name
```

Module 09: Value Functions   Page E-10: Date Functions →More Better


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
cox
Execute
Output:
Work Screen
Clear Screen
Save Script
Done
Internet

```
SELECT    first_name || ' ' || last_name,
          (CURRENT_DATE - birthdate) YEAR TO MONTH AS "AGE"
FROM      talent
ORDER BY  last_name
```

FIRST_NAME||''||LAST_NAME	AGE
Jennifer Aniston	+000000034-07
Orlando Bloom	+000000026-08
Marlon Brando	+000000079-05
George Clooney	+000000042-04
Kevin Costner	+000000048-08
Tom Cruise	+000000041-02
Johnny Depp	+000000040-03
Colin Farrell	+000000027-03
Harrison Ford	+000000061-02
Ed Harris	+000000052-09
Samuel L. Jackson	+000000054-09
Angelina Jolie	+000000028-03
Nicole Kidman	+000000036-03
Ian McKellen	+000000064-04
FIRST_NAME||''||LAST_NAME	AGE
Demi Moore	+000000040-10
Al Pacino	+000000063-05
Michelle Pfeiffer	+000000045-04
Brad Pitt	+000000039-09
Robert Redford	+000000066-01
Julia Roberts	+000000035-10

But an even better solution would be to deal with the result as an interval value.

```
SELECT    first_name || ' ' || last_name,
(CURRENT_DATE - birthdate) YEAR TO MONTH AS "AGE"
FROM      talent
ORDER BY  last_name
```

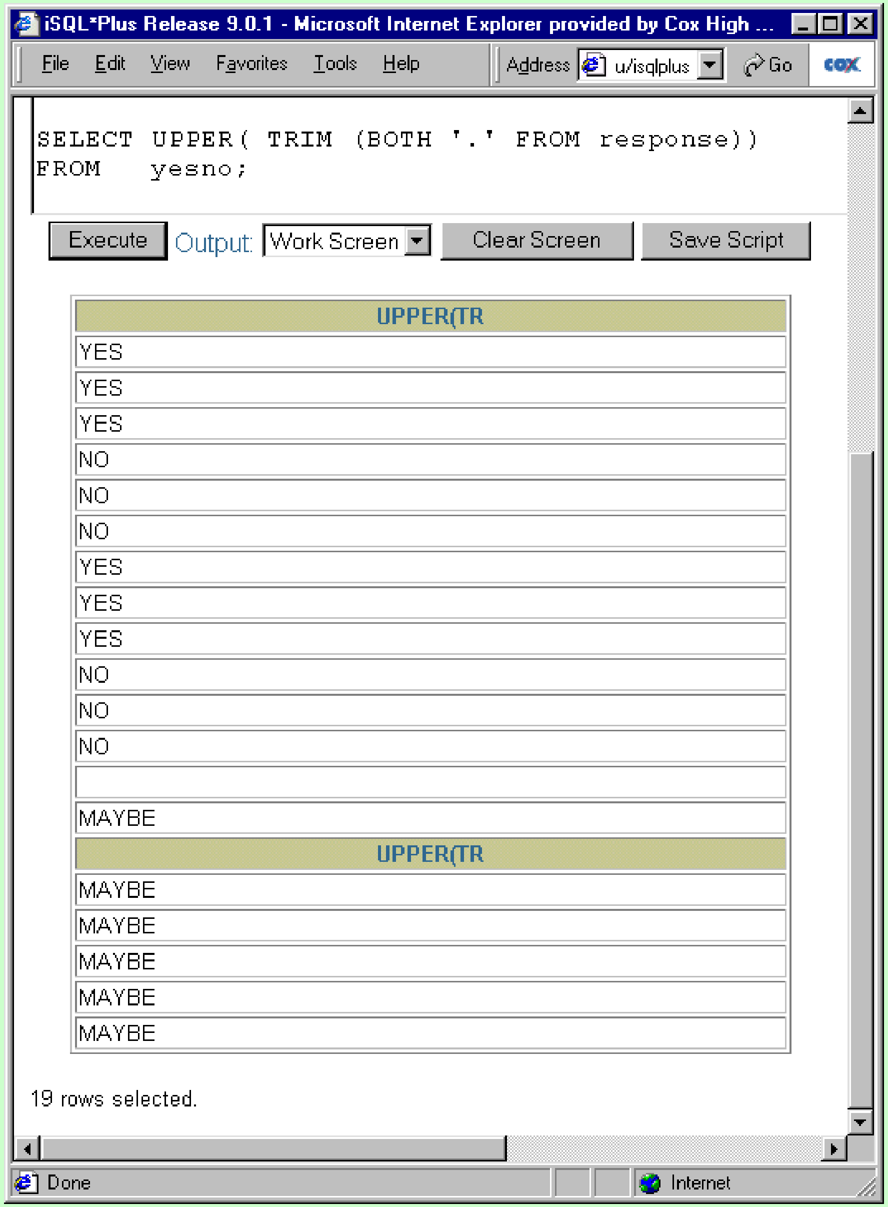
Page F-1: Nesting Functions

Functions may be used anywhere in a SQL program where their return value datatype is permitted.

So in addition to all of the usual places that you might expect to see a function call, don’t overlook the opportunity to replace a function parameter with a call to a function.

Using functions in this fashion, one inside the other, is referred to as nesting functions, and the ‘inside’ functions are ‘nested functions’.

Module 09: Value Functions   Page F-2: Nested Function Example



```
SELECT UPPER( TRIM (BOTH '.' FROM response))
FROM   yesno;
```

```
UPPER(TR
YES
YES
YES
NO
NO
NO
YES
YES
YES
NO
NO
NO

MAYBE
UPPER(TR
MAYBE
MAYBE
MAYBE
MAYBE
MAYBE

19 rows selected.
```

Although this example is somewhat contrived, it's not unlikely that you would need to perform this kind of work.

In many organizations there are standards in place that specify that all (or most) character data in the database is stored in upper case format.

It may be more convenient for the programmer to adjust the data as it's coming into the database than it is to ask the user to reenter the data. In like fashion, a user than leans on the keyboard may generate superfluous characters (spaces, tabs, angle brackets, and the like) that can easily be removed by the programmer.

Page G-1: Vendor Extensions

Every software provider includes features in their products that extend the functionality and usability of their tools.

Even in the case of programming language standards (eg. COBOL, SQL, ...), vendors embellish their offerings with extensions.

Programmers opt to use these extensions as a means to improve their own productivity – generally these extensions simplify the work.

But this short term boost in productivity needs to be offset against longer term maintenance and support issues. Writing code that uses vendor (proprietary) extensions pretty much guarantees that your code will not work on another platform.

In the work place, always check your code against the company's standards and policies. If use of extensions is not permitted, then don't use them.

Page G-2: Vendor Extensions

Oracle and MySQL both offer a number of extensions to the value functions that are mandated by the SQL:1999 standard.

In the next few slides we'll examine these extensions.

I'll highlight the Oracle-specific extensions on pages with a red-colored background, and the MySQL extensions will be highlighted on a blue-colored background.

So here we go - -

Character functions

Number functions

Datetime functions

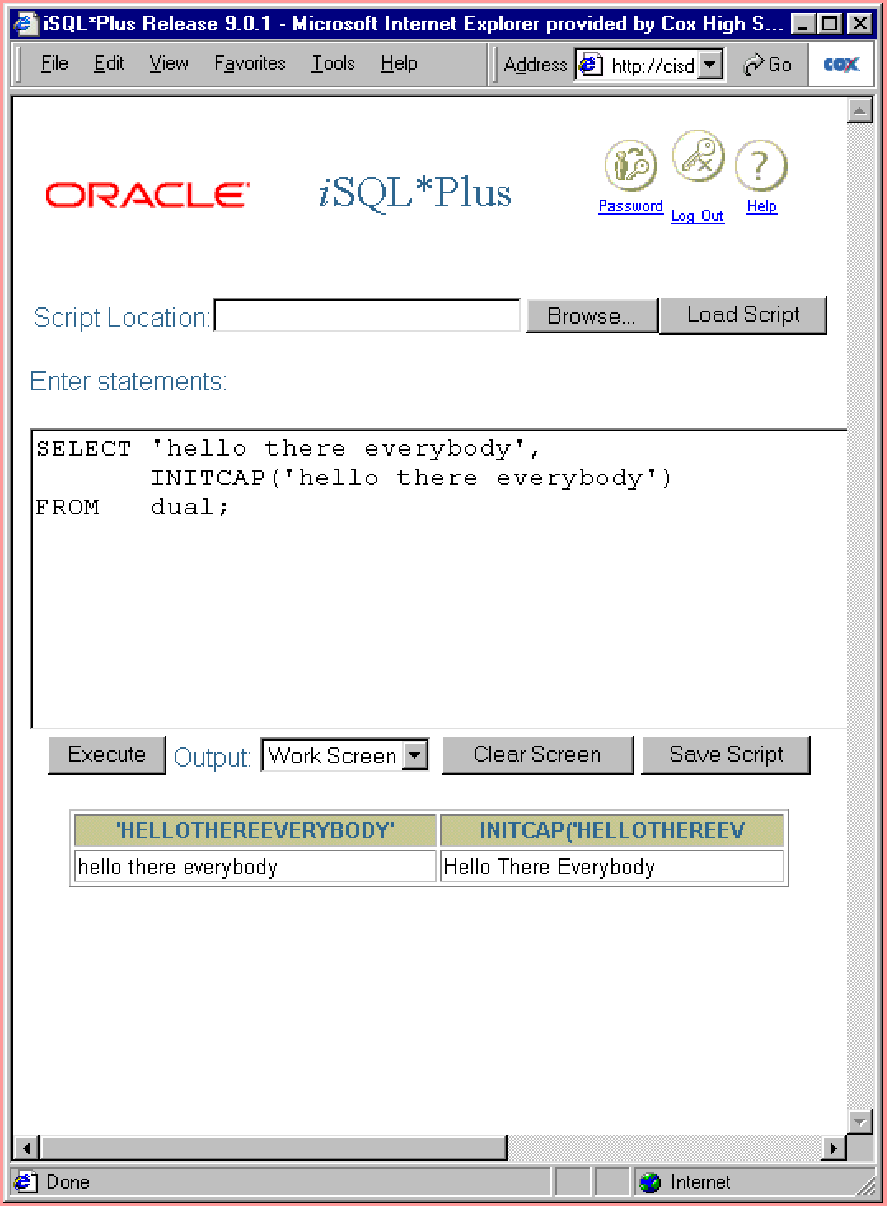
Page G-3: Character Function Extensions

The more commonly used character function extensions are presented here in this table.

Oracle	MySQL
INITCAP()	
CONCAT()	
	REPEAT()
ASC()	ASCII()
CHR()	CHAR()
LPAD()	LPAD()
RPAD()	RPAD()
	RIGHT()
	LEFT()

Module 09: Value Functions

Page G-4: INITCAP



```
SELECT 'hello there everybody',
       INITCAP('hello there everybody')
FROM   dual;
```

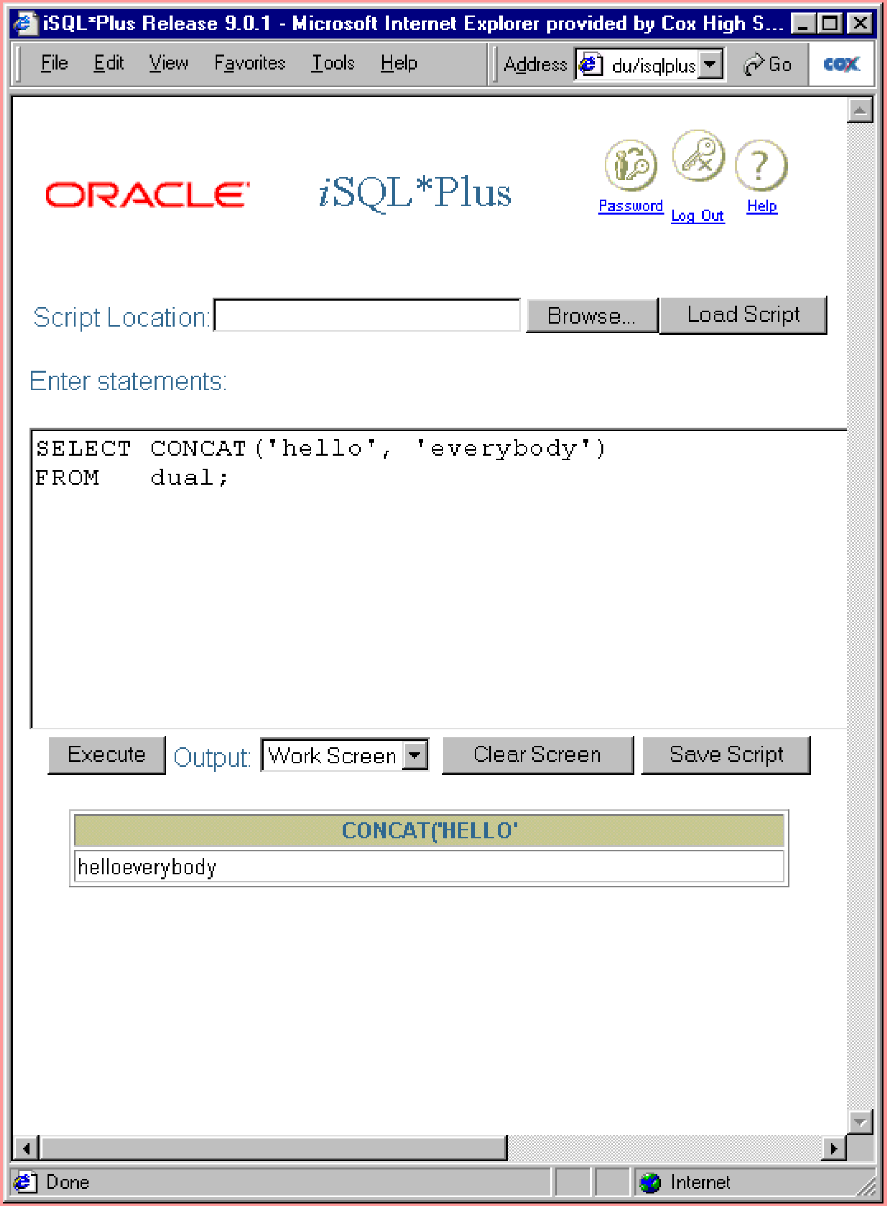
'HELLOTHEREEVERYBODY'	INITCAP('HELLOTHEREEV
hello there everybody	Hello There Everybody

The INITCAP function capitalizes the initial letter (ie. The first letter) in each word in the character string.

This includes the first (leftmost) character in the string, as well as any other character that is preceded by a space.

INITCAP(string value)

Page G-5: CONCAT



```
SELECT  CONCAT('hello', 'everybody')
FROM    dual;
```

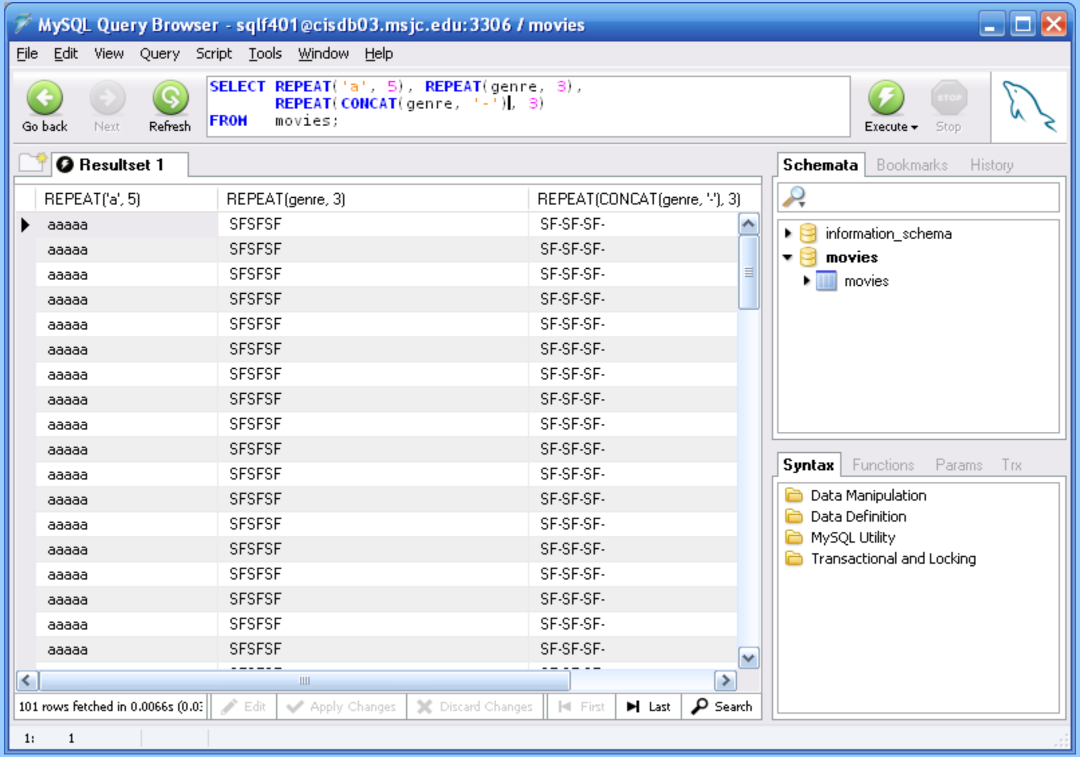
CONCAT('HELLO'
helloeverybody

CONCAT provides the same functionality as the concatenation operator ||, only through an explicit function call.

CONCAT(string value, string value)

Module 09: Value Functions

Page G-6: REPEAT



MySQL Query Browser - sqlf401@cisdb03.msjc.edu:3306 / movies
File Edit View Query Script Tools Window Help
Go back
Next
Refresh
Execute
Stop
Resultset 1
Schemata Bookmarks History
information_schema
movies
movies
Syntax Functions Params Trx
Data Manipulation
Data Definition
MySQL Utility
Transactional and Locking
Edit
Apply Changes
Discard Changes
First
Last
Search

```
SELECT REPEAT('a', 5), REPEAT(genre, 3),
       REPEAT(CONCAT(genre, '-'), 3)
FROM   movies;
```

REPEAT('a', 5)	REPEAT(genre, 3)	REPEAT(CONCAT(genre, '-'), 3)
aaaaa	SFSFSF	SF-SF-SF-
aaaaa	SFSFSF	SF-SF-SF-
aaaaa	SFSFSF	SF-SF-SF-
aaaaa	SFSFSF	SF-SF-SF-
aaaaa	SFSFSF	SF-SF-SF-
aaaaa	SFSFSF	SF-SF-SF-
aaaaa	SFSFSF	SF-SF-SF-
aaaaa	SFSFSF	SF-SF-SF-
aaaaa	SFSFSF	SF-SF-SF-
aaaaa	SFSFSF	SF-SF-SF-
aaaaa	SFSFSF	SF-SF-SF-
aaaaa	SFSFSF	SF-SF-SF-
aaaaa	SFSFSF	SF-SF-SF-
aaaaa	SFSFSF	SF-SF-SF-
aaaaa	SFSFSF	SF-SF-SF-
aaaaa	SFSFSF	SF-SF-SF-
aaaaa	SFSFSF	SF-SF-SF-
aaaaa	SFSFSF	SF-SF-SF-

```
101 rows fetched in 0.0066s (0.0:
1:   1
```

REPEAT(value, number) repeats the value, the number of times specified.

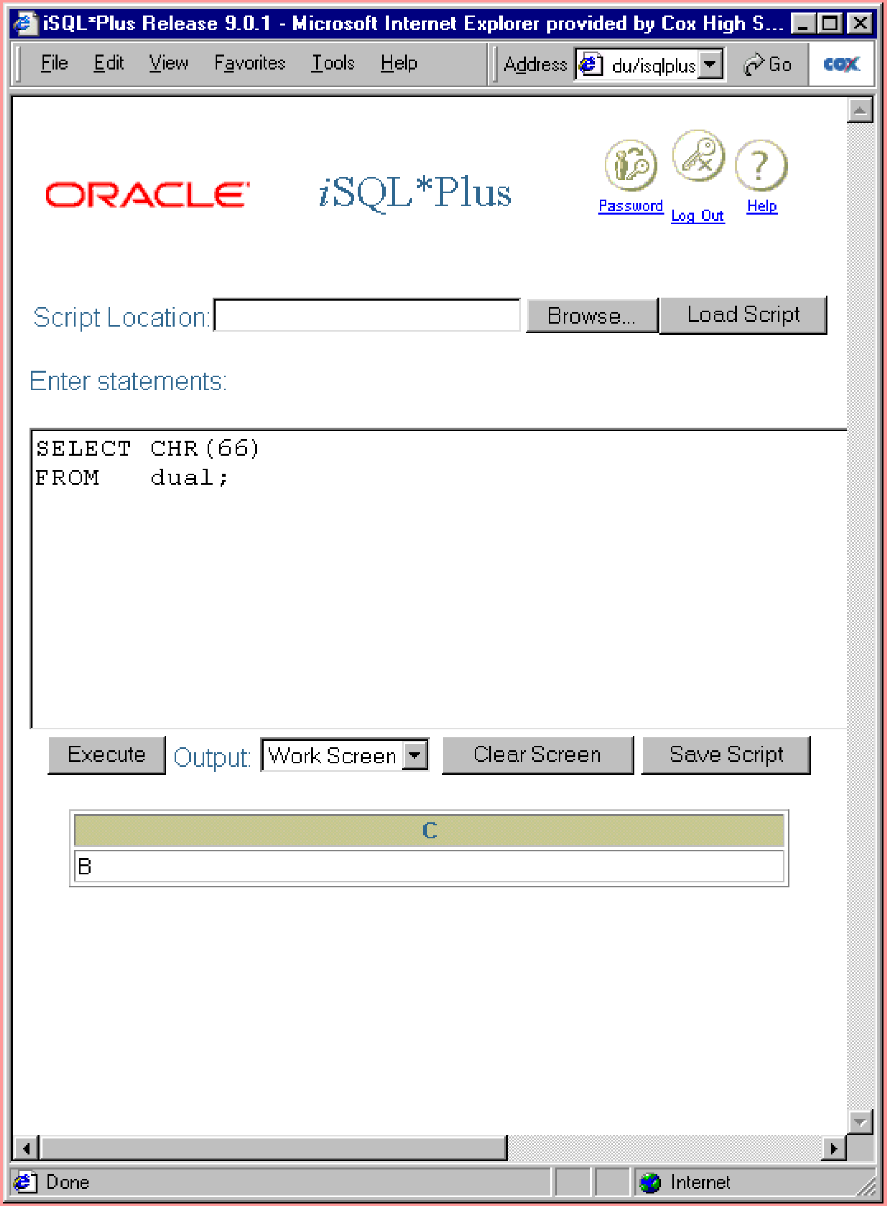
In the first example, the letter ‘a’ is repeated 5 times.

In the next example, the column, genre, is repeated three times.

And in the final example, the column genre is concatenated with the string literal ‘-, and the result is repeated 3 times.

Module 09: Value Functions

Page G-7: CHR



```
SELECT  CHR(66)
FROM    dual;
```

C
B

Returns the character that corresponds to this ordinal position in the collating sequence in the character set used by the database.

CHR(integer value)

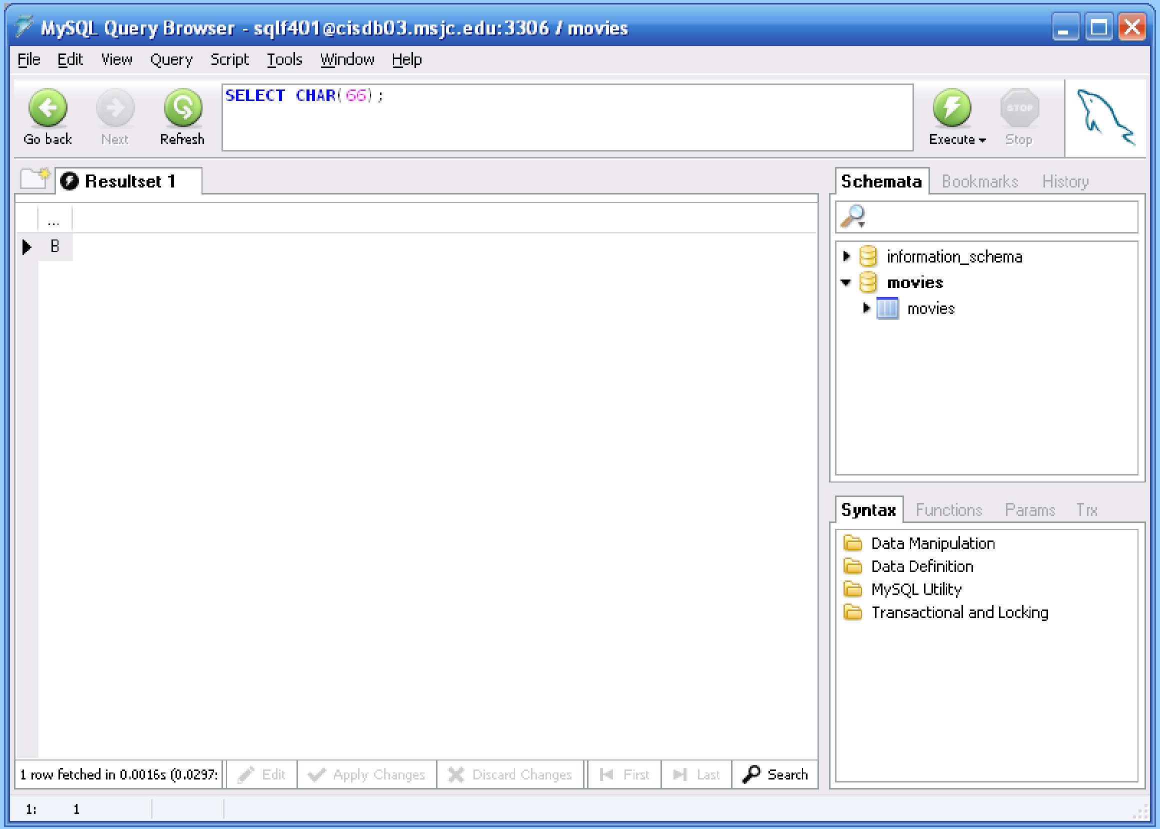
©1998-2018 / Bergin

Module 09: Value Functions

Page G-8: CHAR

Returns the character that corresponds to this ordinal position in the collating sequence in the character set used by the database.

CHR(integer value)



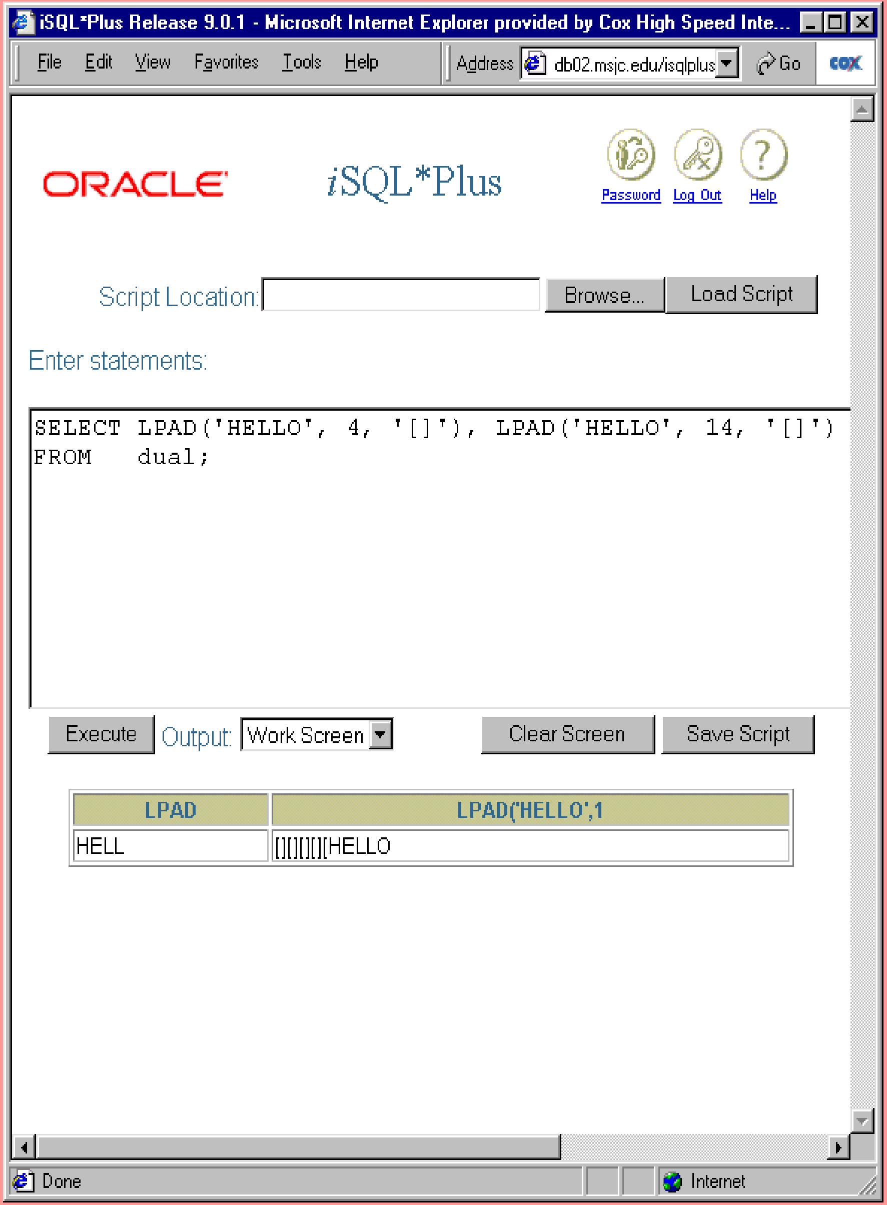
```
SELECT CHAR(66);
```

```
...
B
```

1 row fetched in 0.0016s (0.0297:

1: 1

Page G-9: LPAD


iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Speed Inte...
File Edit View Favorites Tools Help
Address db02.msjc.edu/isqlplus
Go
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 LPAD('HELLO', 4, '[]'), LPAD('HELLO', 14, '[]')
FROM   dual;
```

LPAD	LPAD('HELLO',1
HELL	[][][][][HELLO

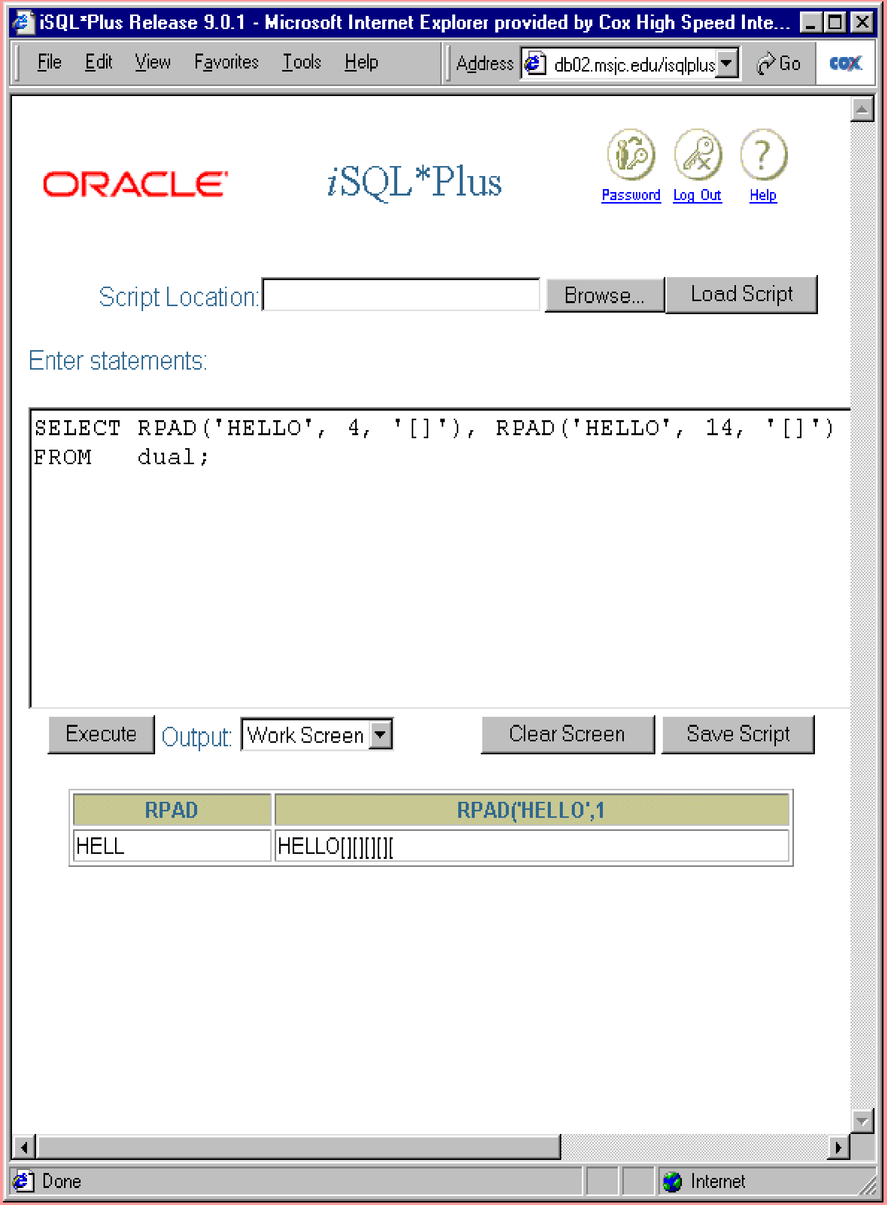
LPAD left pads, or, left fills, a string with however many copies of another string you specify.

LPAD(target-string, number, filler-string)

LPAD will expand the first string son that it contains as many characters as are specified by the number parameter.

If the number value is smaller than the length of the target string, the target string will be truncated. If the length of the target string is smaller than the number value, the target string will be padded with the filler string.

Page G-10: RPAD


iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Speed Inte...
File Edit View Favorites Tools Help
Address db02.msjc.edu/isqlplus
Go
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 RPAD('HELLO', 4, '[]'), RPAD('HELLO', 14, '[]')
FROM   dual;
```

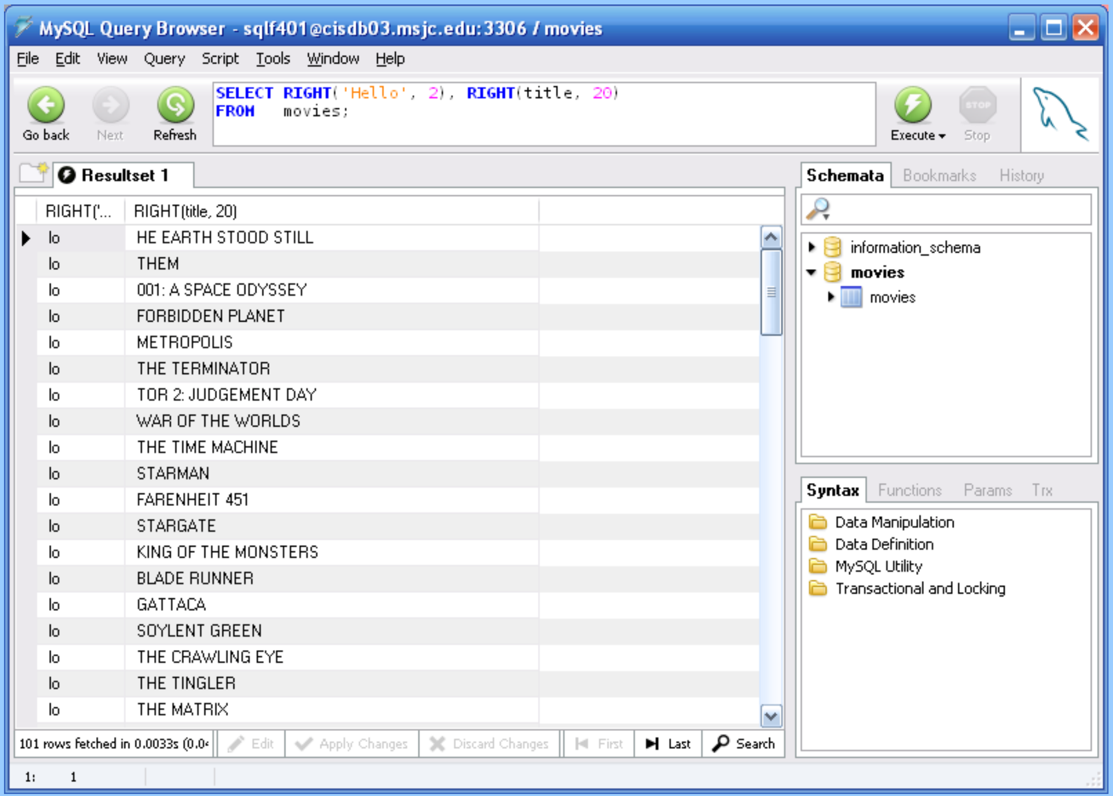
RPAD	RPAD('HELLO',1
HELL	HELLO[][][][][

Same as LPAD, only the padding occurs on the right-hand side of the target string.

Module 09: Value Functions

Page G-11: RIGHT

RIGHT(value, number), returns the rightmost characters in the value string.



```
SELECT RIGHT('Hello', 2), RIGHT(title, 20)
FROM   movies;
```

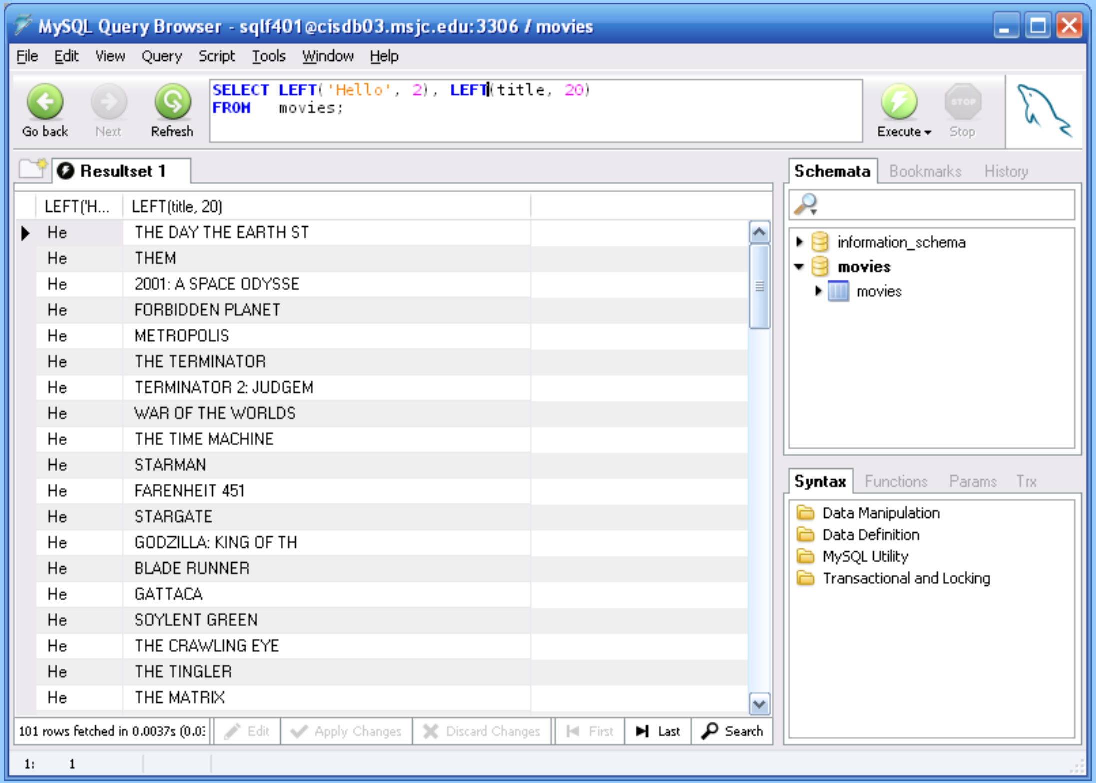
RIGHT('...	RIGHT(title, 20)
lo	HE EARTH STOOD STILL
lo	THEM
lo	001: A SPACE ODYSSEY
lo	FORBIDDEN PLANET
lo	METROPOLIS
lo	THE TERMINATOR
lo	TOR 2: JUDGEMENT DAY
lo	WAR OF THE WORLDS
lo	THE TIME MACHINE
lo	STARMAN
lo	FARENHEIT 451
lo	STARGATE
lo	KING OF THE MONSTERS
lo	BLADE RUNNER
lo	GATTACA
lo	SOYLENT GREEN
lo	THE CRAWLING EYE
lo	THE TINGLER
lo	THE MATRIX

101 rows fetched in 0.0033s (0.0(

Module 09: Value Functions

Page G-12: LEFT

LEFT(value, number), returns the leftmost characters in the value string.



```
SELECT LEFT('Hello', 2), LEFT(title, 20)
FROM   movies;
```

LEFT('H...	LEFT(title, 20)
He	THE DAY THE EARTH ST
He	THEM
He	2001: A SPACE ODYSSE
He	FORBIDDEN PLANET
He	METROPOLIS
He	THE TERMINATOR
He	TERMINATOR 2: JUDGEM
He	WAR OF THE WORLDS
He	THE TIME MACHINE
He	STARMAN
He	FARENHEIT 451
He	STARGATE
He	GODZILLA: KING OF TH
He	BLADE RUNNER
He	GATTACA
He	SOYLENT GREEN
He	THE CRAWLING EYE
He	THE TINGLER
He	THE MATRIX

101 rows fetched in 0.0037s (0.0:

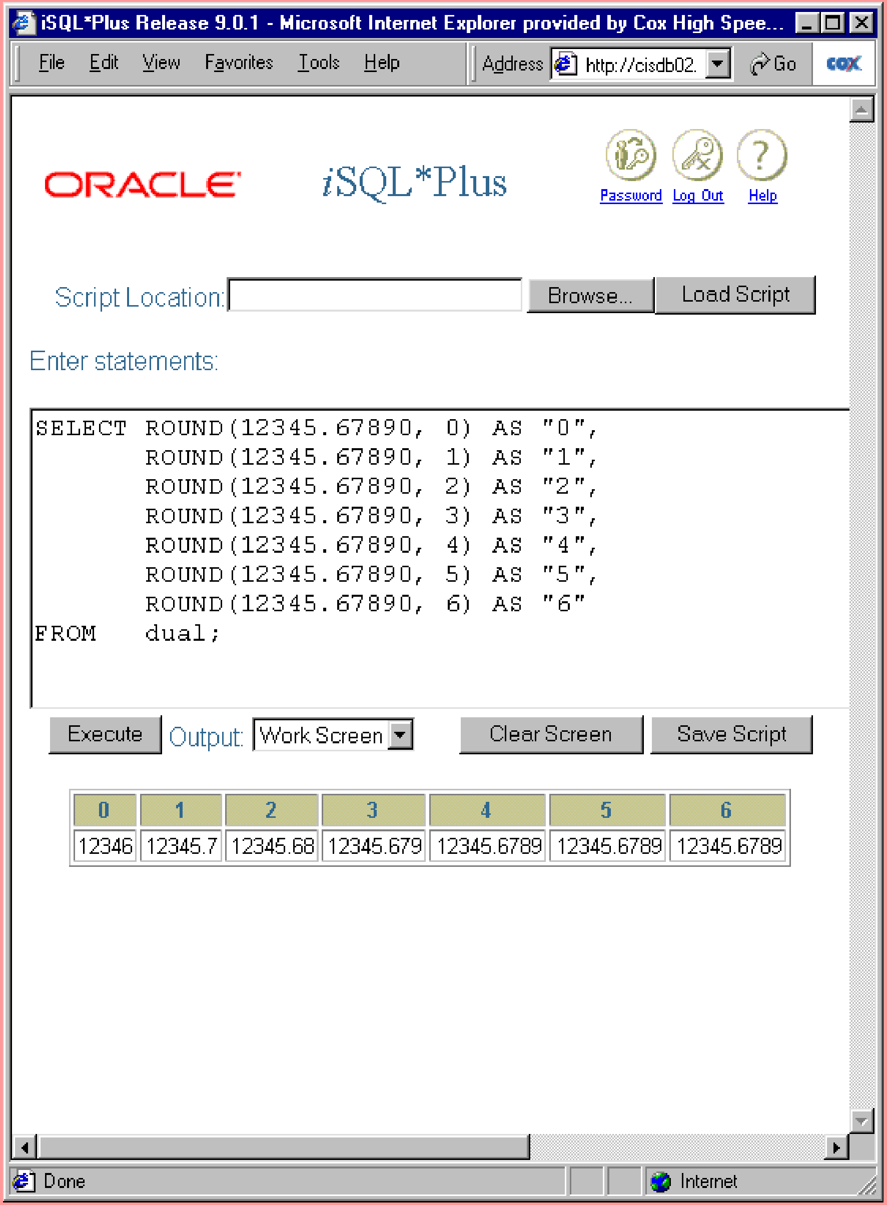
1: 1

Page H-1: Numeric Function Extensions

The more commonly used extensions to the numeric functions are presented here in this table.

Oracle	MySQL
ROUND()	ROUND()
TRUNC()	TRUNCATE()
CEIL()	CEIL(), or CEILING()
FLOOR()	FLOOR()
POWER(x,n)	POW(x,n), or POWER(x,n)
SQRT()	SQRT()

Page H-2: ROUND


iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Spee...
File Edit View Favorites Tools Help
Address http://cisdb02. Go
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 ROUND(12345.67890, 0) AS "0",
       ROUND(12345.67890, 1) AS "1",
       ROUND(12345.67890, 2) AS "2",
       ROUND(12345.67890, 3) AS "3",
       ROUND(12345.67890, 4) AS "4",
       ROUND(12345.67890, 5) AS "5",
       ROUND(12345.67890, 6) AS "6"
FROM   dual;
```

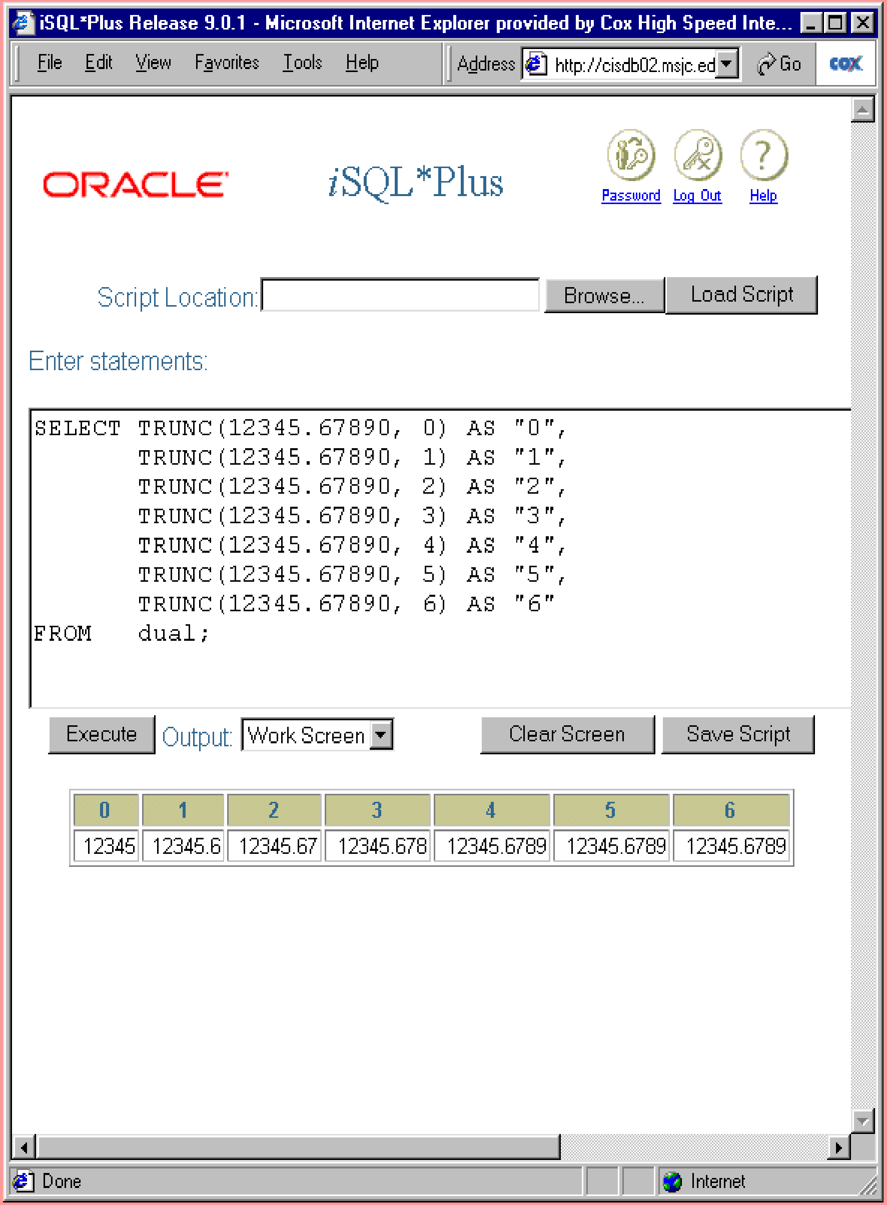
0	1	2	3	4	5	6
12346	12345.7	12345.68	12345.679	12345.6789	12345.6789	12345.6789

The ROUND function rounds off a numeric value to the number of decimal places specified by the precision value.

ROUND(numeric value, precision)

Module 09: Value Functions

Page H-3: TRUNC



iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Speed Inte...
File Edit View Favorites Tools Help
Address http://cisdb02.msjc.ed
Go
ORACLE
iSQL*Plus
Password
Log Out
Help
Script Location:
Browse...
Load Script
Enter statements:

```
SELECT  TRUNC(12345.67890, 0) AS "0",
        TRUNC(12345.67890, 1) AS "1",
        TRUNC(12345.67890, 2) AS "2",
        TRUNC(12345.67890, 3) AS "3",
        TRUNC(12345.67890, 4) AS "4",
        TRUNC(12345.67890, 5) AS "5",
        TRUNC(12345.67890, 6) AS "6"
FROM    dual;
```

Execute
Output: Work Screen
Clear Screen
Save Script

0	1	2	3	4	5	6
12345	12345.6	12345.67	12345.678	12345.6789	12345.6789	12345.6789

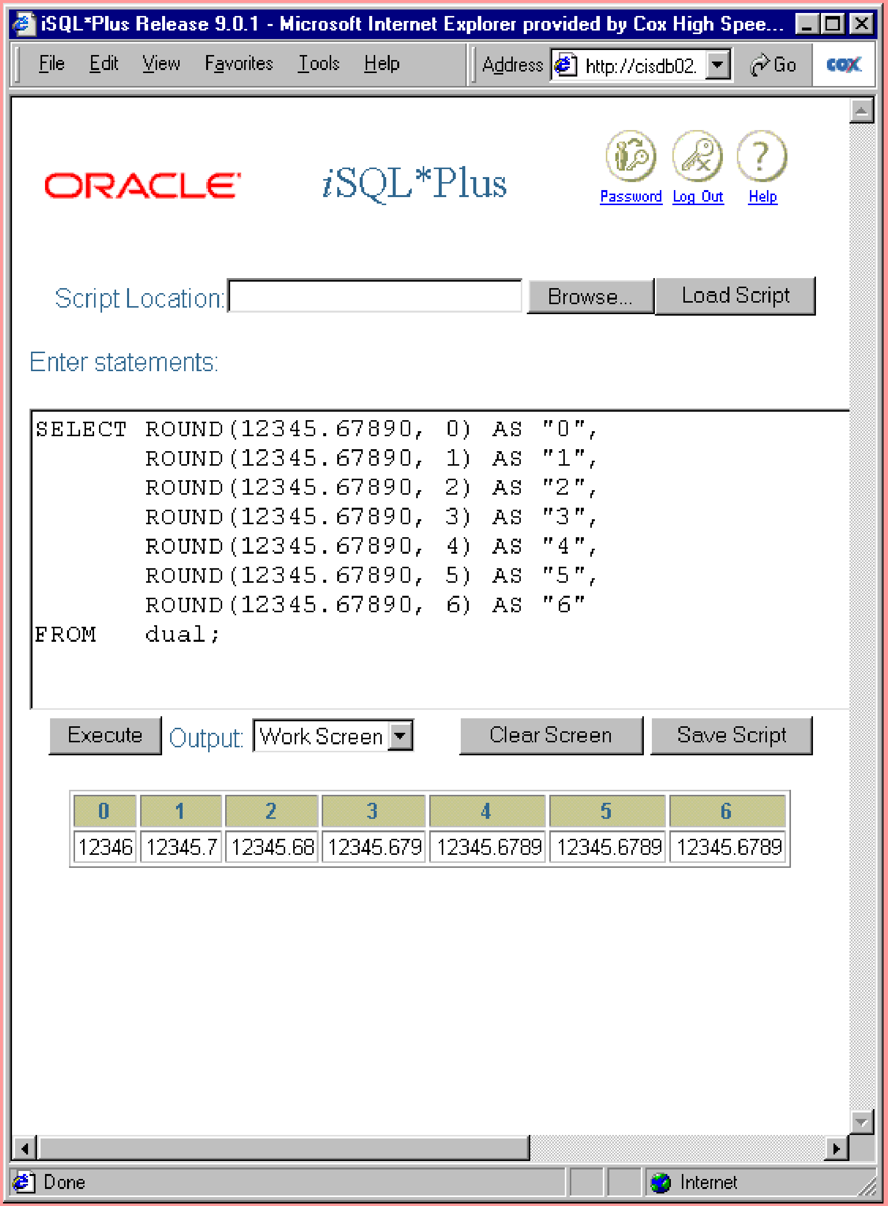
In contrast to the ROUND function, the TRUNC (truncate) function, just ‘chops things off’ according to the precision value parameter.

TRUNC(numeric value, precision)

Compare the results of these two functions on the following slide.

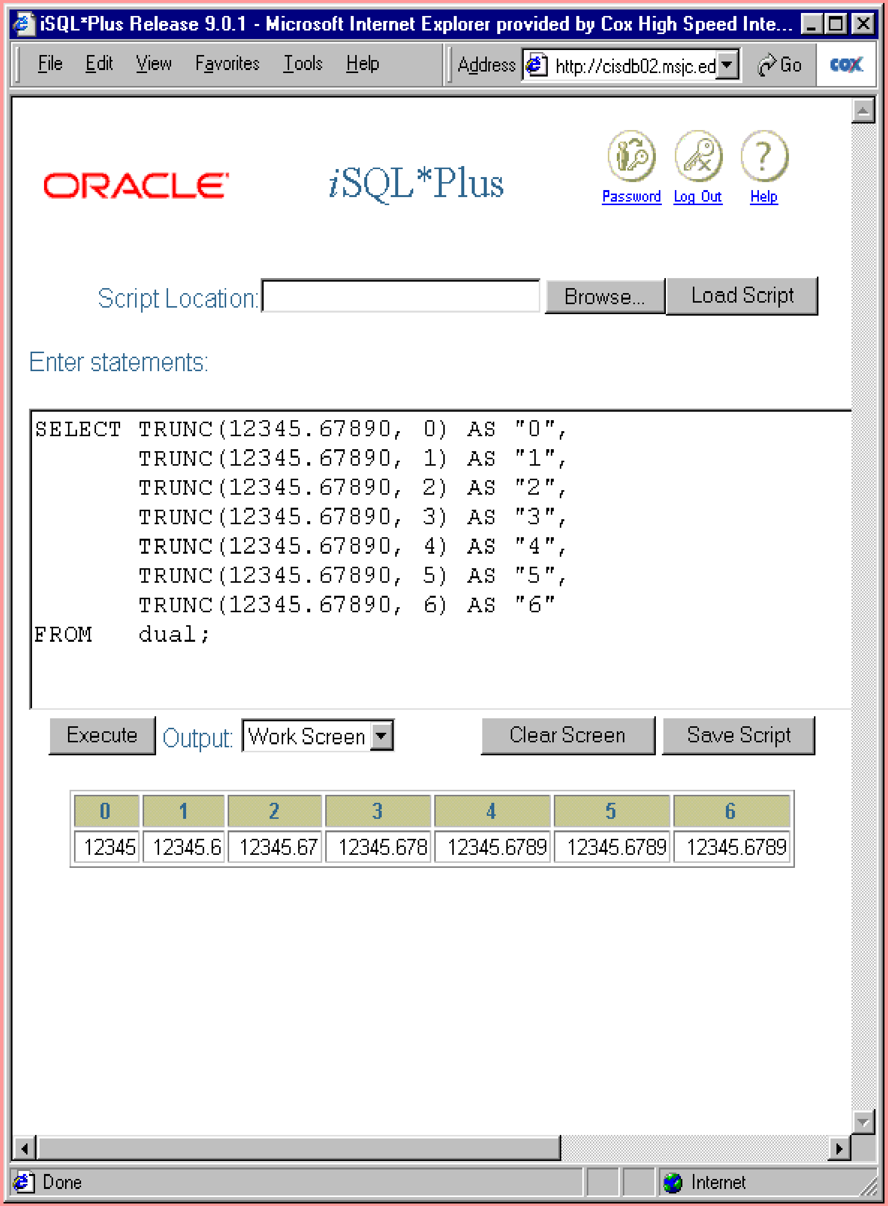
Module 09: Value Functions

Page H-4: TRUNC vs/ ROUND



```
SELECT  ROUND(12345.67890, 0) AS "0",
        ROUND(12345.67890, 1) AS "1",
        ROUND(12345.67890, 2) AS "2",
        ROUND(12345.67890, 3) AS "3",
        ROUND(12345.67890, 4) AS "4",
        ROUND(12345.67890, 5) AS "5",
        ROUND(12345.67890, 6) AS "6"
FROM    dual;
```

0	1	2	3	4	5	6
12346	12345.7	12345.68	12345.679	12345.6789	12345.6789	12345.6789

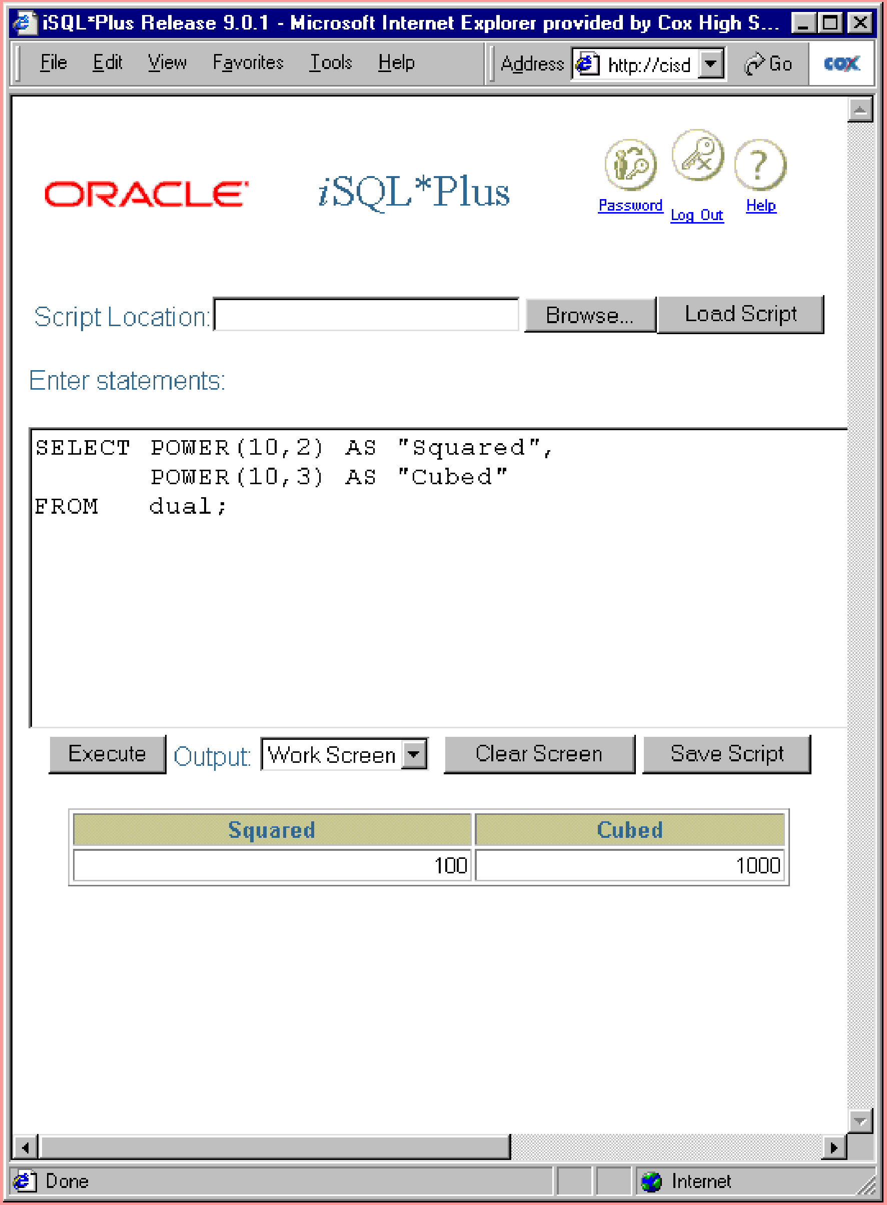

iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Speed Inte...
File Edit View Favorites Tools Help
Address http://cisdb02.msjc.ed
Go
ORACLE iSQL*Plus
Password Log Out Help
Script Location:
Browse...
Load Script
Enter statements:
Execute
Output: Work Screen
Clear Screen
Save Script
Done
Internet

```
SELECT  TRUNC(12345.67890, 0) AS "0",
        TRUNC(12345.67890, 1) AS "1",
        TRUNC(12345.67890, 2) AS "2",
        TRUNC(12345.67890, 3) AS "3",
        TRUNC(12345.67890, 4) AS "4",
        TRUNC(12345.67890, 5) AS "5",
        TRUNC(12345.67890, 6) AS "6"
FROM    dual;
```

0	1	2	3	4	5	6
12345	12345.6	12345.67	12345.678	12345.6789	12345.6789	12345.6789

Module 09: Value Functions

Page H-5: POWER



```
SELECT  POWER(10,2)  AS  "Squared",
        POWER(10,3)  AS  "Cubed"
FROM    dual;
```

Squared	Cubed
100	1000

The POWER function provides a simple way for the programmer to handle exponentiation.

POWER(numeric value, power value)

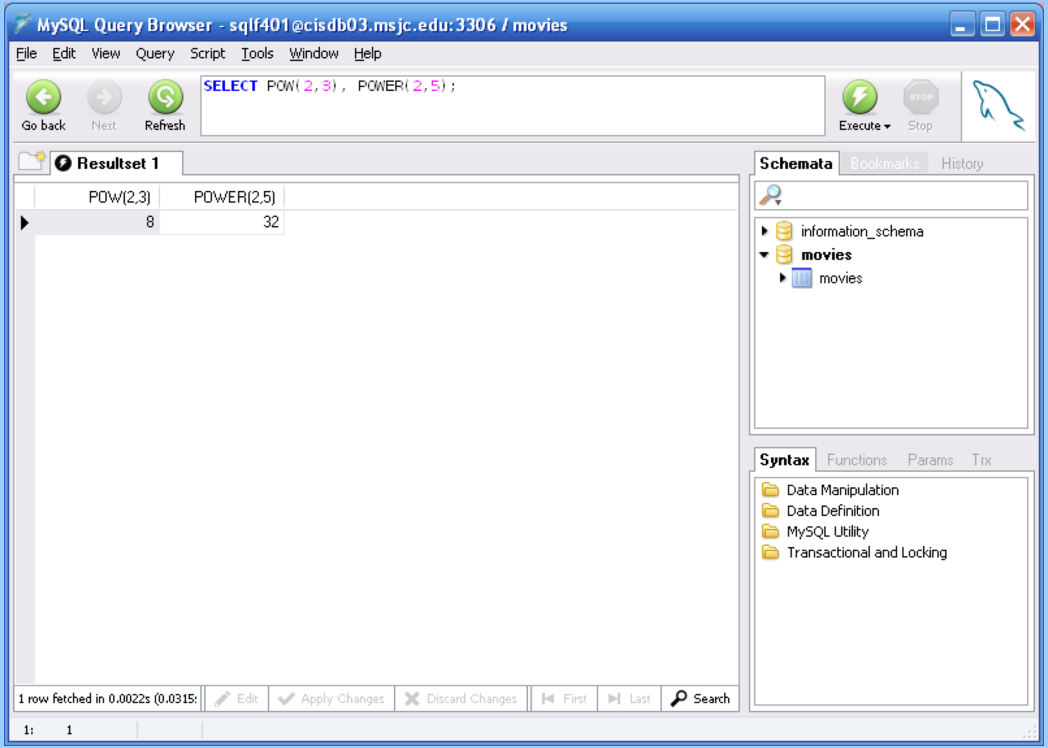
Module 09: Value Functions

Page H-6: POW / POWER

MySQL provides the same functionality with the POWER, or POW functions.

POWER(numeric value, power value)

POW(numeric value, power value)



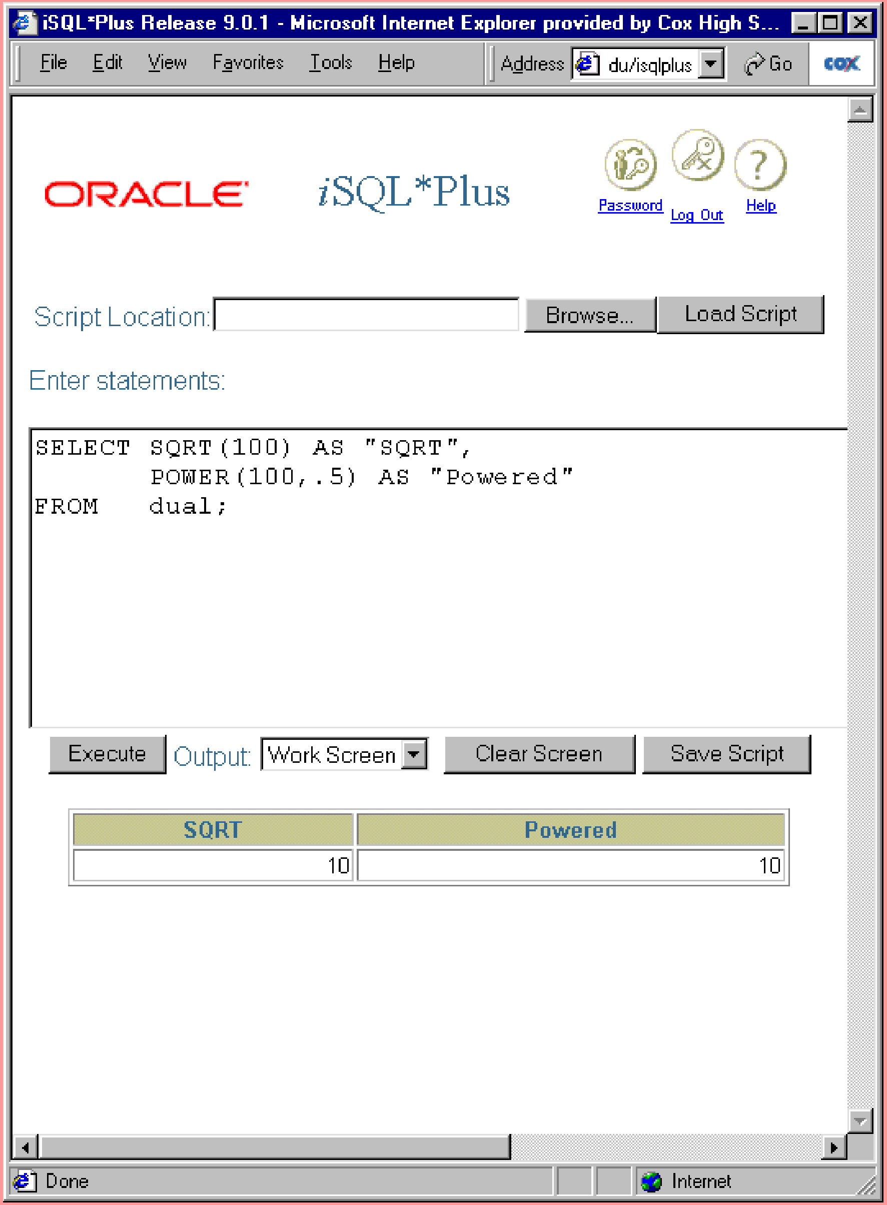
MySQL Query Browser - sqlf401@cisdb03.msjc.edu:3306 / movies

```
SELECT POW(2,3), POWER(2,5);
```

POW(2,3)	POWER(2,5)
8	32

1 row fetched in 0.0022s (0.0315:

Page H-7: SQRT



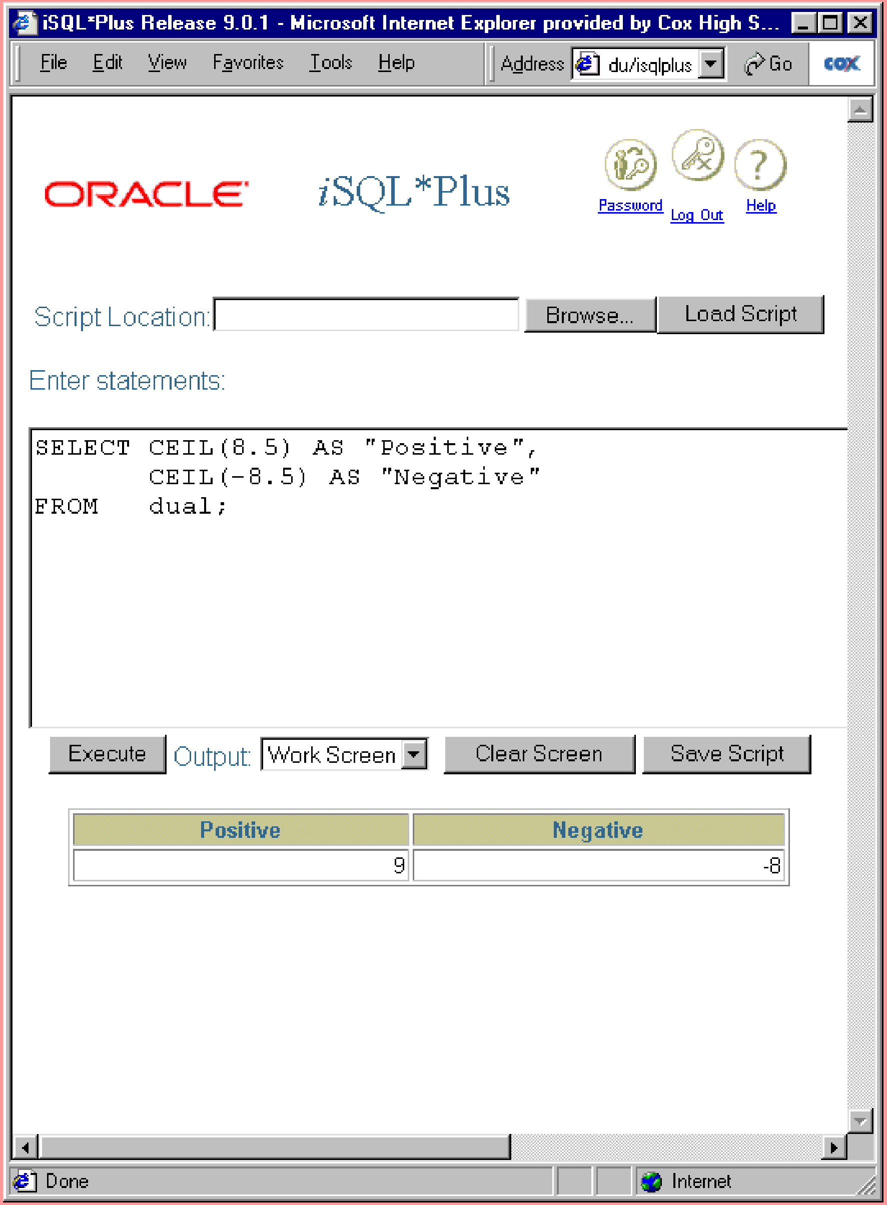
```
SELECT  SQRT(100)  AS  "SQRT",
        POWER(100,.5)  AS  "Powered"
FROM    dual;
```

SQRT	Powered
10	10

SQRT is the square root function. It has a nice mnemonic, but it is rather superfluous since all square roots are ‘simply’ a value raised to the one-half power.

SQRT(numeric value)

Page H-8: CEIL



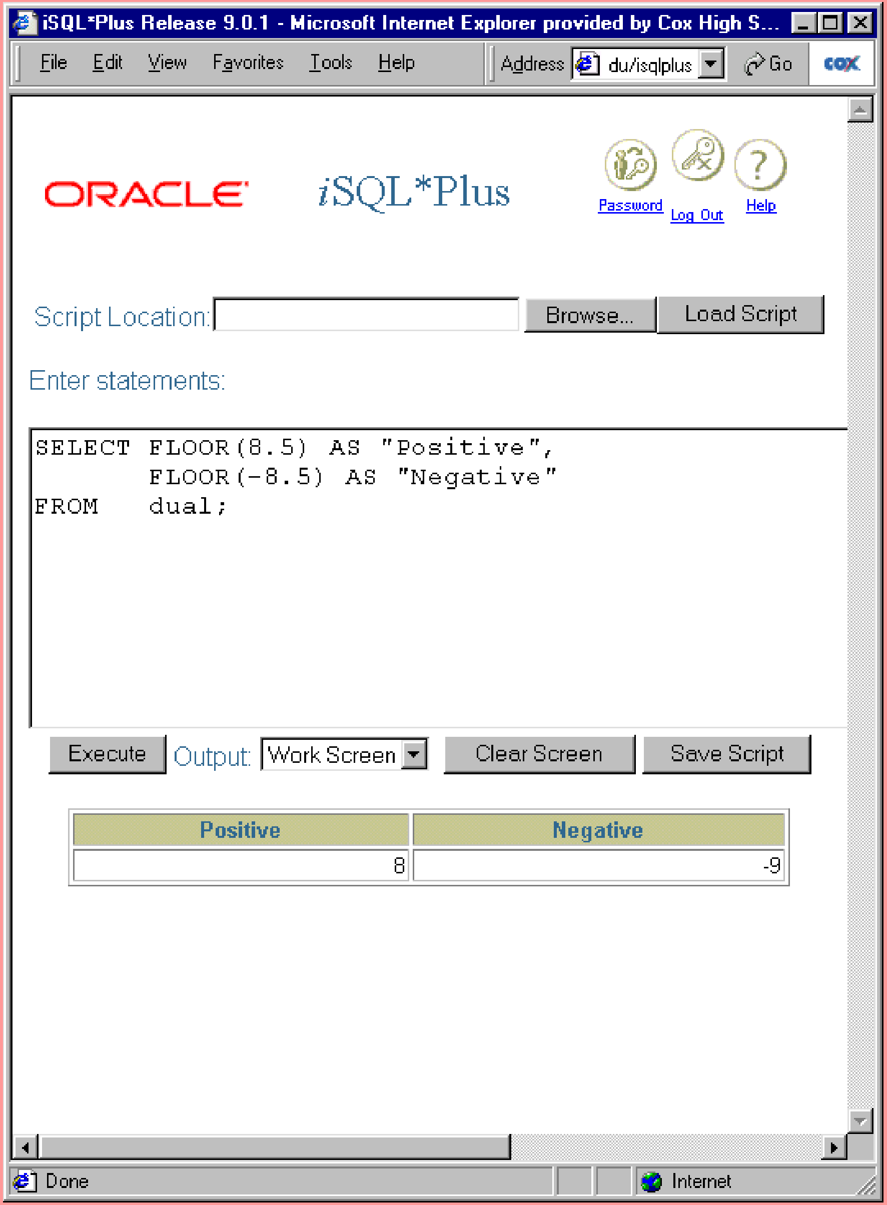
```
SELECT  CEIL(8.5)  AS  "Positive",
        CEIL(-8.5)  AS  "Negative"
FROM    dual;
```

Positive	Negative
9	-8

The CEIL function takes a fractional value and returns the largest integer value. In all cases, CEIL moves rightward on the number line. Negative values towards zero, positive values away from zero.

CEIL(numeric value)

Page H-9: FLOOR



```
SELECT  FLOOR(8.5)  AS  "Positive",
        FLOOR(-8.5)  AS  "Negative"
FROM    dual;
```

Positive	Negative
8	-9

FLOOR is the opposite of CEIL.

In all cases, FLOOR moves leftward on the number line. Negative values away from zero, positive values towards zero.

FLOOR(numeric value)

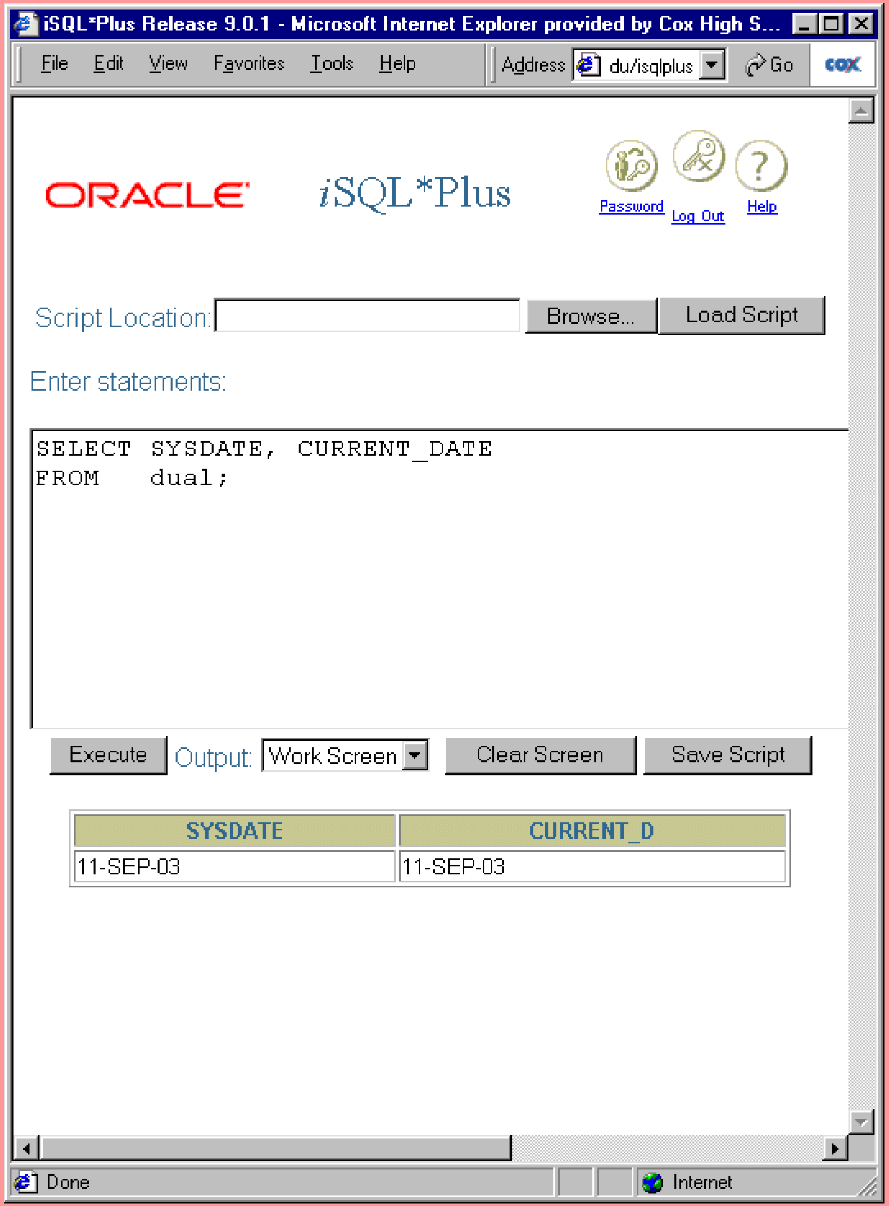
Page I-1: Datetime Functions

We’ll now consider the Oracle extensions to the datetime functions:

Oracle	MySQL
SYSDATE	SYSDATE
NEXT_DAY	
LAST_DAY	
	ADDDATE
ADD_MONTHS	DATE_ADD
MONTHS_BETWEEN	

Module 09: Value Functions

Page I-2: SYSDATE



iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High S...
File Edit View Favorites Tools Help
Address du/isqlplus Go
ORACLE iSQL*Plus
Password Log Out Help
Script Location:
Browse... Load Script
Enter statements:

```
SELECT SYSDATE, CURRENT_DATE
FROM   dual;
```

Execute Output: Work Screen Clear Screen Save Script

SYSDATE	CURRENT_D
11-SEP-03	11-SEP-03

Long before the SQL standard mandated a CURRENT_DATE value, Oracle provided that functionality in its database with the use of the SYSDATE function.

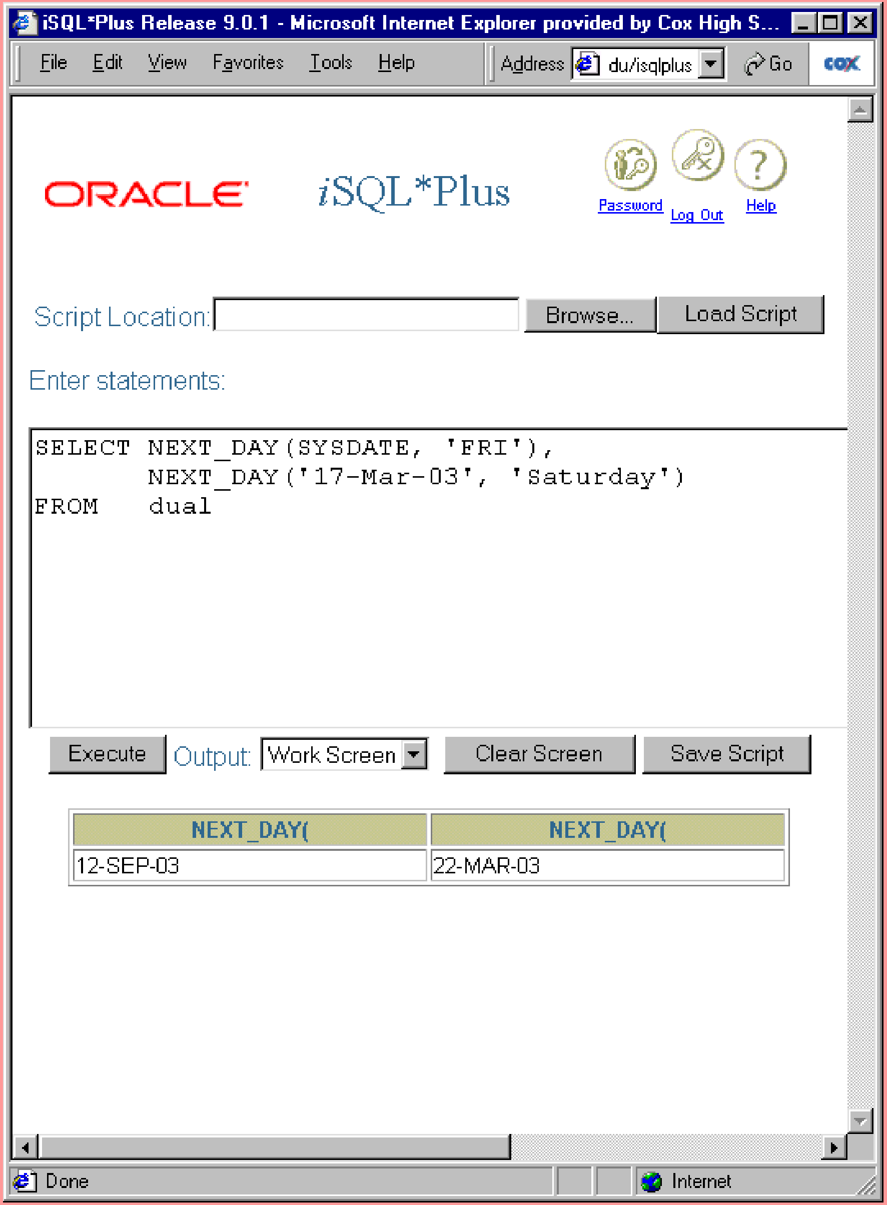
SYSDATE

In my opinion SYSDATE will be carried forward in future release of Oracle, only to insure backwards compatibility for SQL programs that were written prior to the newer standard.

In this regard, my advice is for you to be able to recognize SYSDATE if you were to find it in some SQL code, but I advocate that you use CURRENT_DATE in lieu of SYSDATE.

Module 09: Value Functions

Page I-3: NEXT_DAY



```
SELECT NEXT_DAY(SYSDATE, 'FRI'),
       NEXT_DAY('17-Mar-03', 'Saturday')
FROM   dual
```

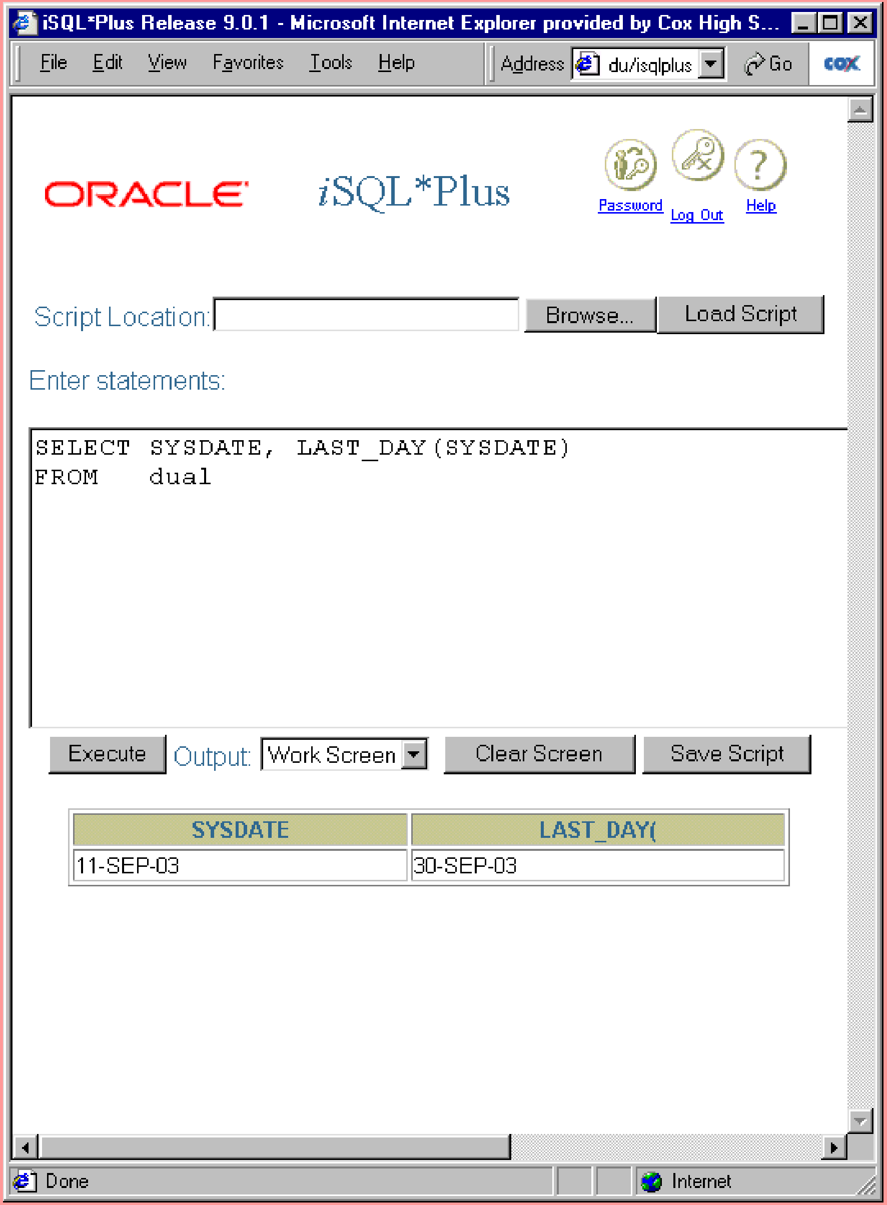
NEXT_DAY(	NEXT_DAY(
12-SEP-03	22-MAR-03

This is a nice little function that will tell you when the next occurrence of a particular day of the week will first fall, after the date value provided.

When's the next Friday after Jan 12?

NEXT_DAY(datetime value, day string)

Page I-4: LAST_DAY



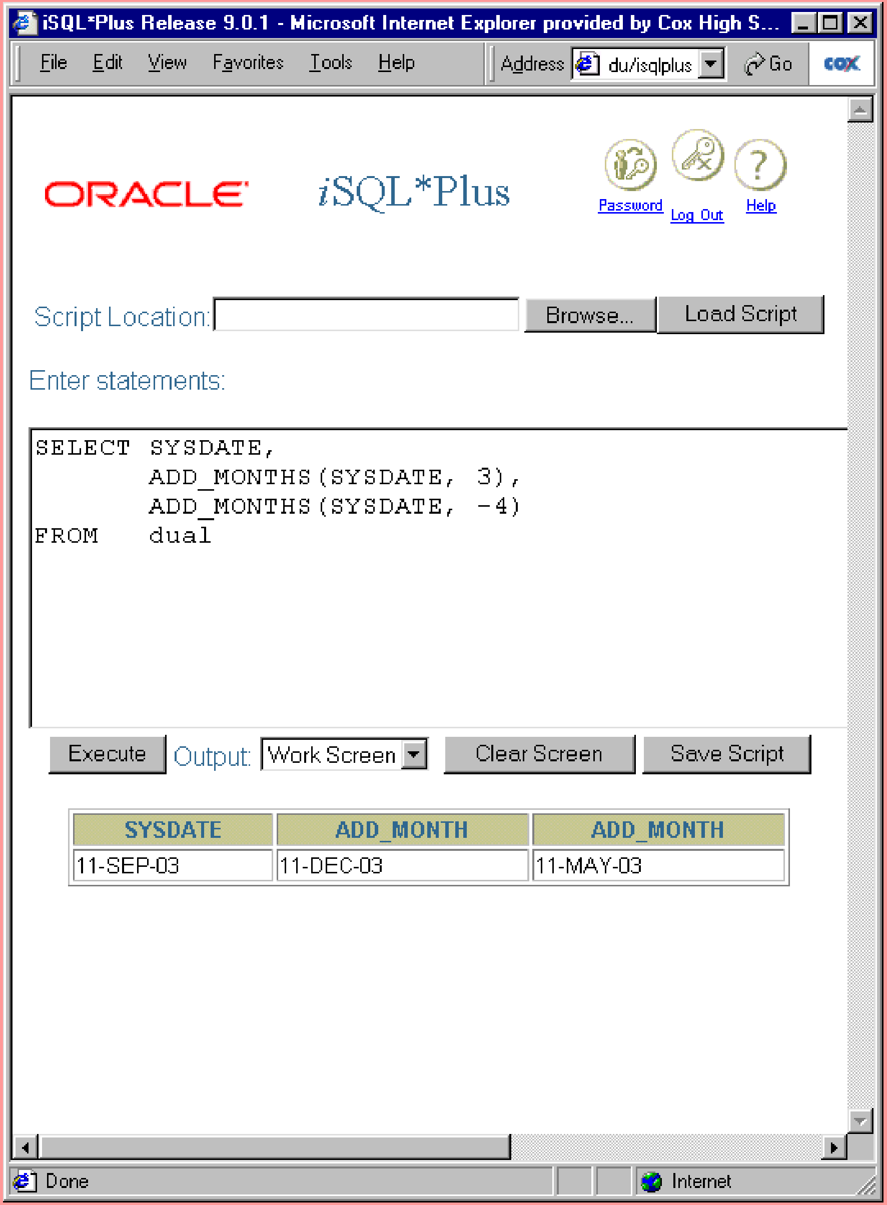
```
SELECT  SYSDATE,  LAST_DAY(SYSDATE)
FROM    dual
```

SYSDATE	LAST_DAY(
11-SEP-03	30-SEP-03

Given a date value, what will the last day of the month be?

LAST_DAY(datetime value)

Page I-5: ADD_MONTHS



```
SELECT  SYSDATE,
        ADD_MONTHS(SYSDATE, 3),
        ADD_MONTHS(SYSDATE, -4)
FROM    dual
```

SYSDATE	ADD_MONTH	ADD_MONTH
11-SEP-03	11-DEC-03	11-MAY-03

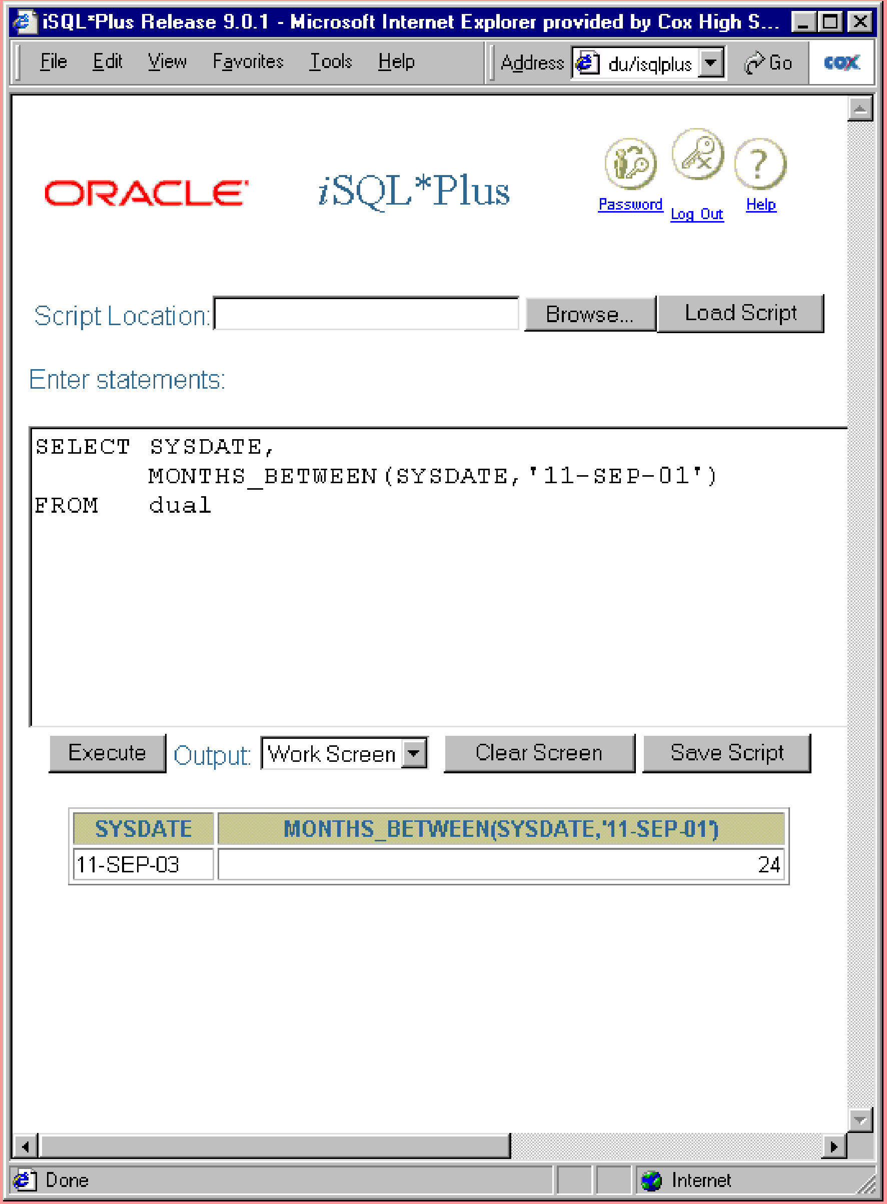
Here is another useful function that Oracle provided, again, long before the standard provided for INTERVAL math.

ADD_MONTHS(datetime value, month value)

Note that the month value may be negative.

Module 09: Value Functions

Page I-6: MONTHS_BETWEEN



```
SELECT  SYSDATE,
        MONTHS_BETWEEN(SYSDATE,'11-SEP-01')
FROM    dual
```

SYSDATE	MONTHS_BETWEEN(SYSDATE,'11-SEP-01')
11-SEP-03	24

This function is not a datetime function since it returns a numeric value. But I've included it in this section because it makes most sense to discuss it in the context of the other date-related functions that we're reviewing.

Page I-7: ADDDATE, DATE_ADD

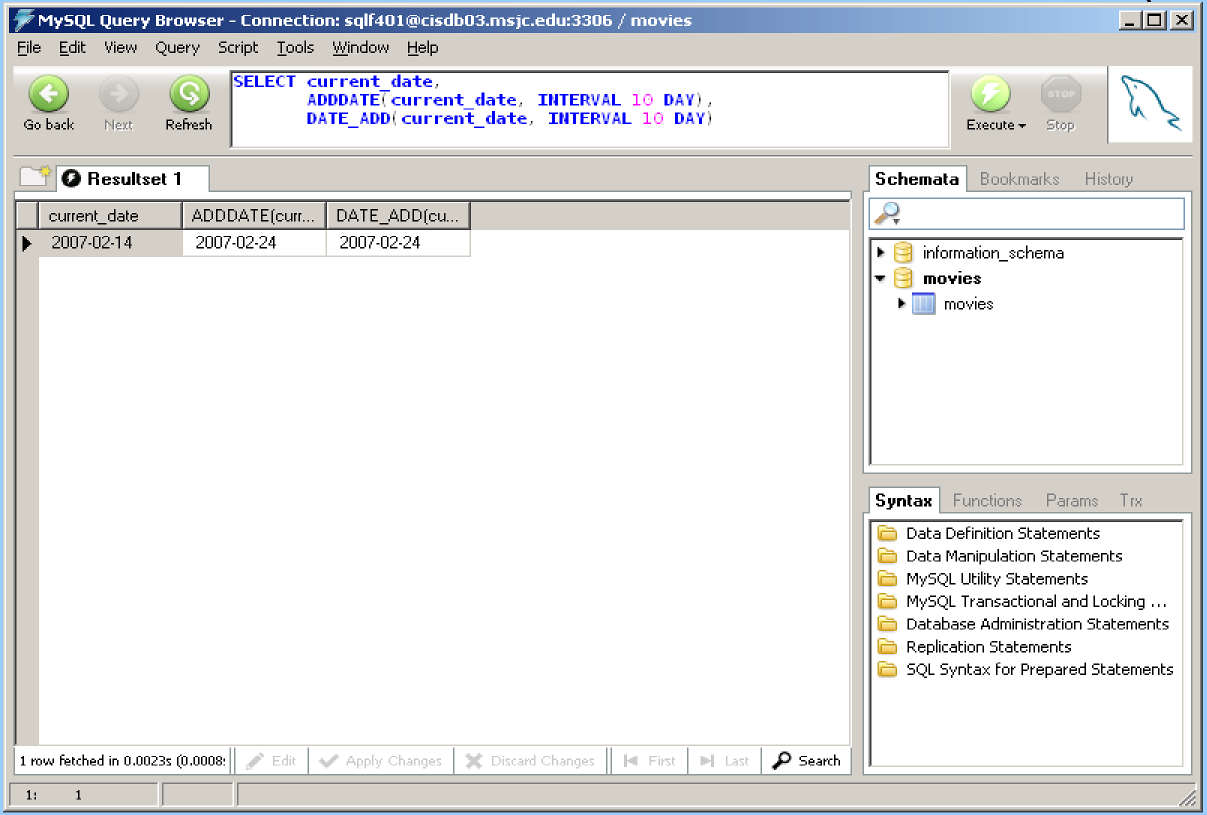
MySQL gives us a couple of functions for ‘advancing’ a date value, and in the first form, these functions are essentially equivalent.

Form 1:

ADDDATE(date, INTERVAL value unit)

DATE ADD(date, INTERVAL value unit)

Form 1


MySQL Query Browser - Connection: sqlf401@cisdb03.msjc.edu:3306 / movies
File Edit View Query Script Tools Window Help
Go back
Next
Refresh
Execute
Stop
Resultset 1
Schemata Bookmarks History
information_schema
movies
movies
Syntax Functions Params Trx
Data Definition Statements
Data Manipulation Statements
MySQL Utility Statements
MySQL Transactional and Locking ...
Database Administration Statements
Replication Statements
SQL Syntax for Prepared Statements
Edit
Apply Changes
Discard Changes
First
Last
Search
1: 1

```
SELECT current_date,
       ADDDATE(current_date, INTERVAL 10 DAY),
       DATE_ADD(current_date, INTERVAL 10 DAY)
```

current_date	ADDDATE(curr...	DATE_ADD(cu...
2007-02-14	2007-02-24	2007-02-24

1 row fetched in 0.0023s (0.0008:

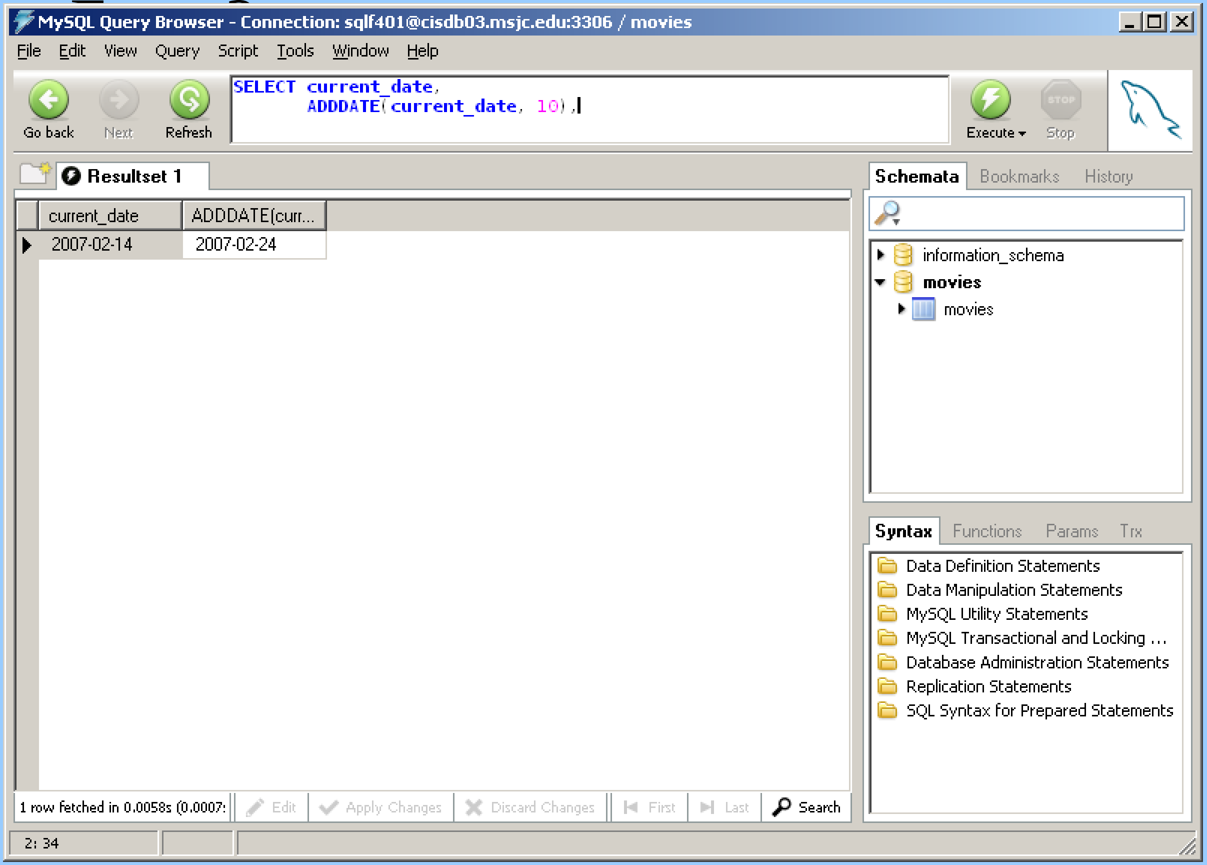
Module 09: Value Functions

Page I-8: ADDDATE, DATE_ADD

The ADDDATE function has a second form, for which there is no similar DATE_ADD counterpart.

Form 2:

ADDDATE(date, dayvalue)



```
SELECT current_date,
       ADDDATE(current_date, 10);
```

current_date	ADDDATE(curr...
2007-02-14	2007-02-24

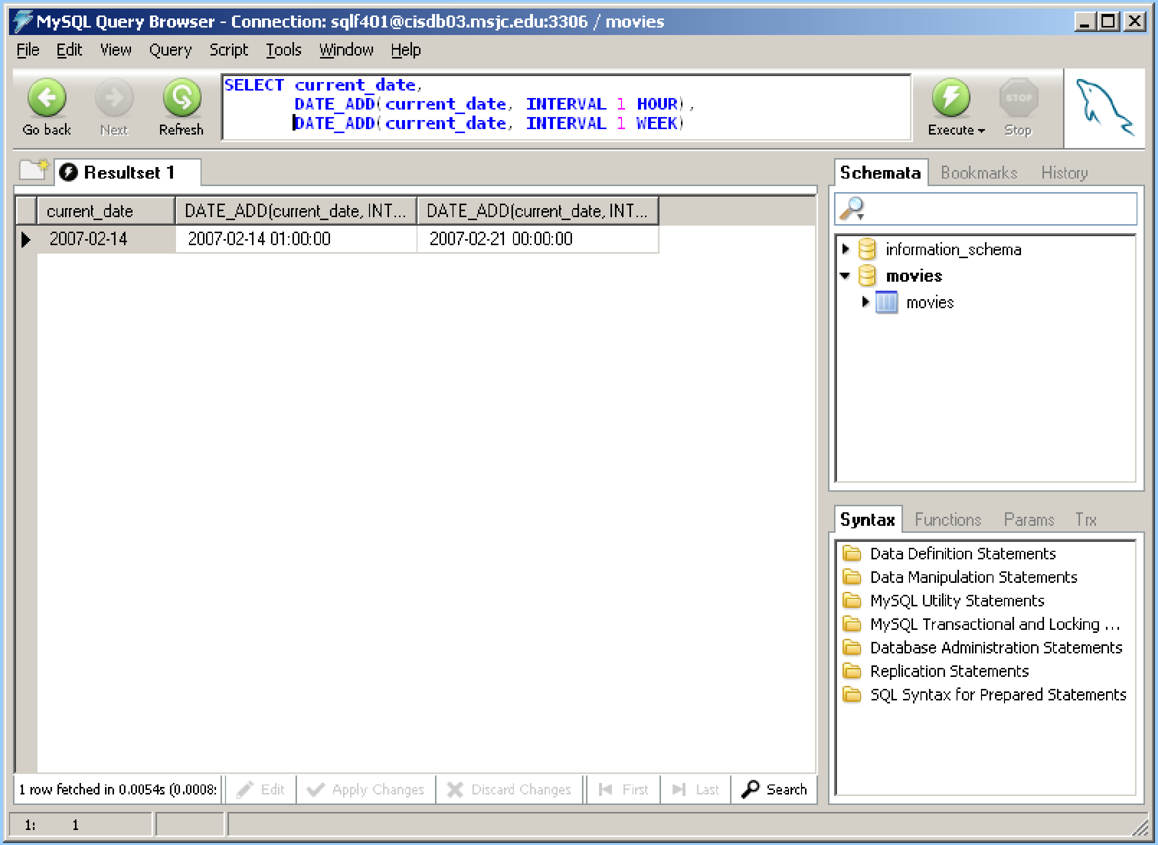
```
1 row fetched in 0.0058s (0.0007:
```

```
2: 34
```

Page I-9: ADDDATE, DATE_ADD

My preference is for the first form. I like to explicitly specify which date/time units are being added to the date value.

In my opinion, this form is unambiguous and anyone reading it will understand exactly what the programmer's intentions



```
SELECT current_date,
       DATE_ADD(current_date, INTERVAL 1 HOUR),
       DATE_ADD(current_date, INTERVAL 1 WEEK)
```

current_date	DATE_ADD(current_date, INT...	DATE_ADD(current_date, INT...
2007-02-14	2007-02-14 01:00:00	2007-02-21 00:00:00

1 row fetched in 0.0054s (0.0008:

1: 1

Interval Units

SECOND
MINUTE
HOUR

DAY
WEEK
MONTH
QUARTER
YEAR

Page J-1: Casting Functions

The casting functions are used to convert values that are stored as one data type into ‘essentially’ the same value but stored as another data type.

In Oracle there are three casting functions that you need to be vary familiar with:

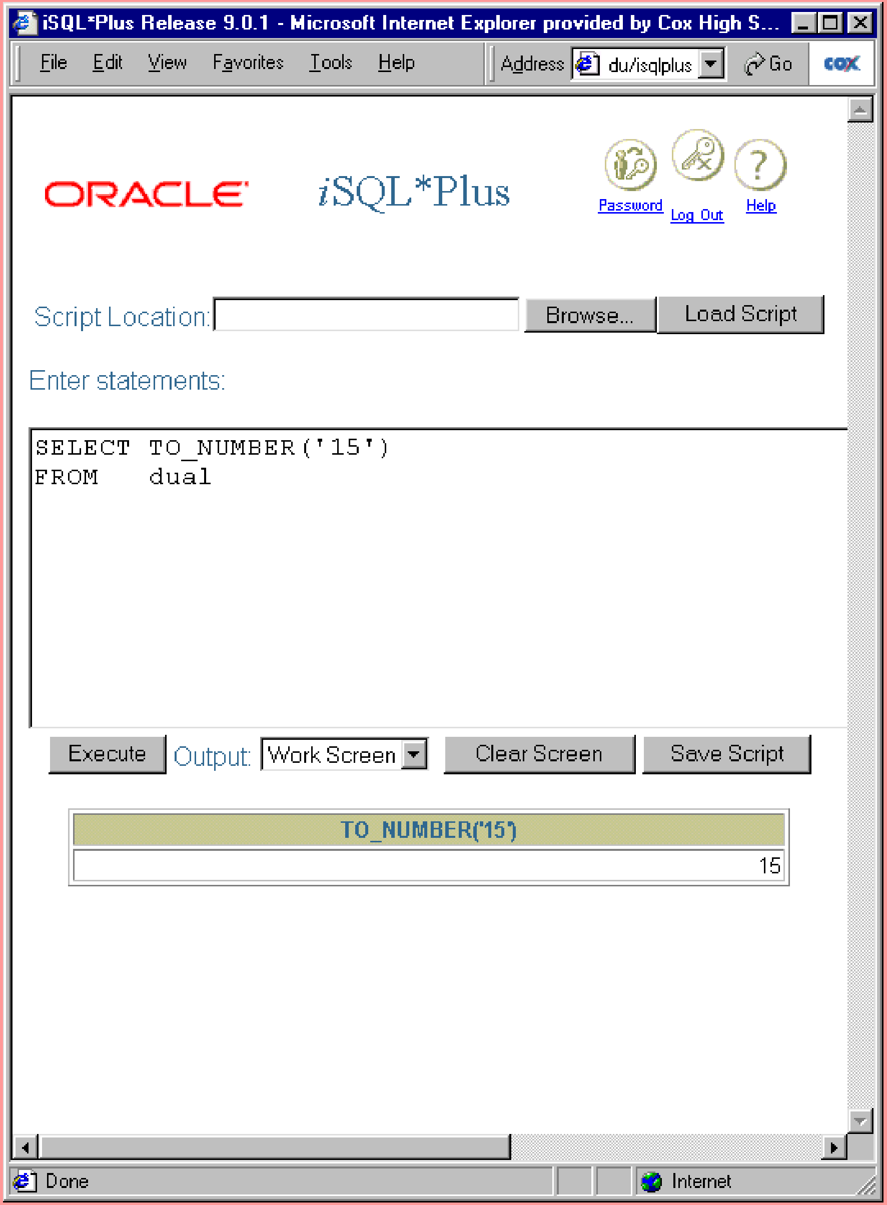
TO_CHAR (numeric or datetime value)

TO_DATE(character value)

TO_NUMBER(character value)

Module 09: Value Functions

Page J-2: TO_NUMBER

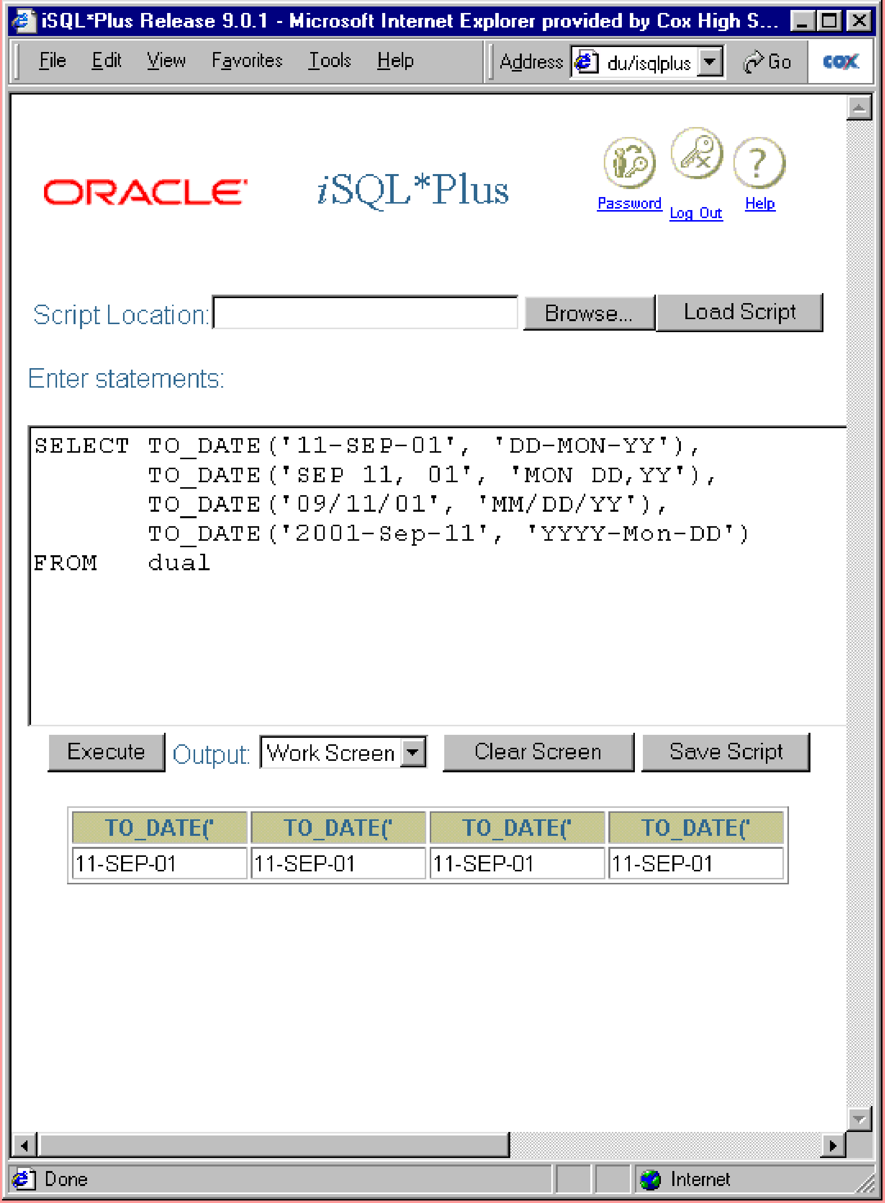

iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High S...
File Edit View Favorites Tools Help
Address du/isqlplus Go
ORACLE iSQL*Plus
Password Log Out Help
Script Location:
Browse... Load Script
Enter statements:
Execute Output: Work Screen Clear Screen Save Script
Done Internet

```
SELECT  TO_NUMBER('15')
FROM    dual
```

TO_NUMBER('15')
15

TO_NUMBER casts a character value as a numeric data type. ***Provided that the character value can map into the numeric domain.***

Page J-3: TO_DATE


iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High S...
File Edit View Favorites Tools Help
Address du/isqlplus Go
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  TO_DATE('11-SEP-01', 'DD-MON-YY'),
        TO_DATE('SEP 11, 01', 'MON DD,YY'),
        TO_DATE('09/11/01', 'MM/DD/YY'),
        TO_DATE('2001-Sep-11', 'YYYY-Mon-DD')
FROM    dual
```

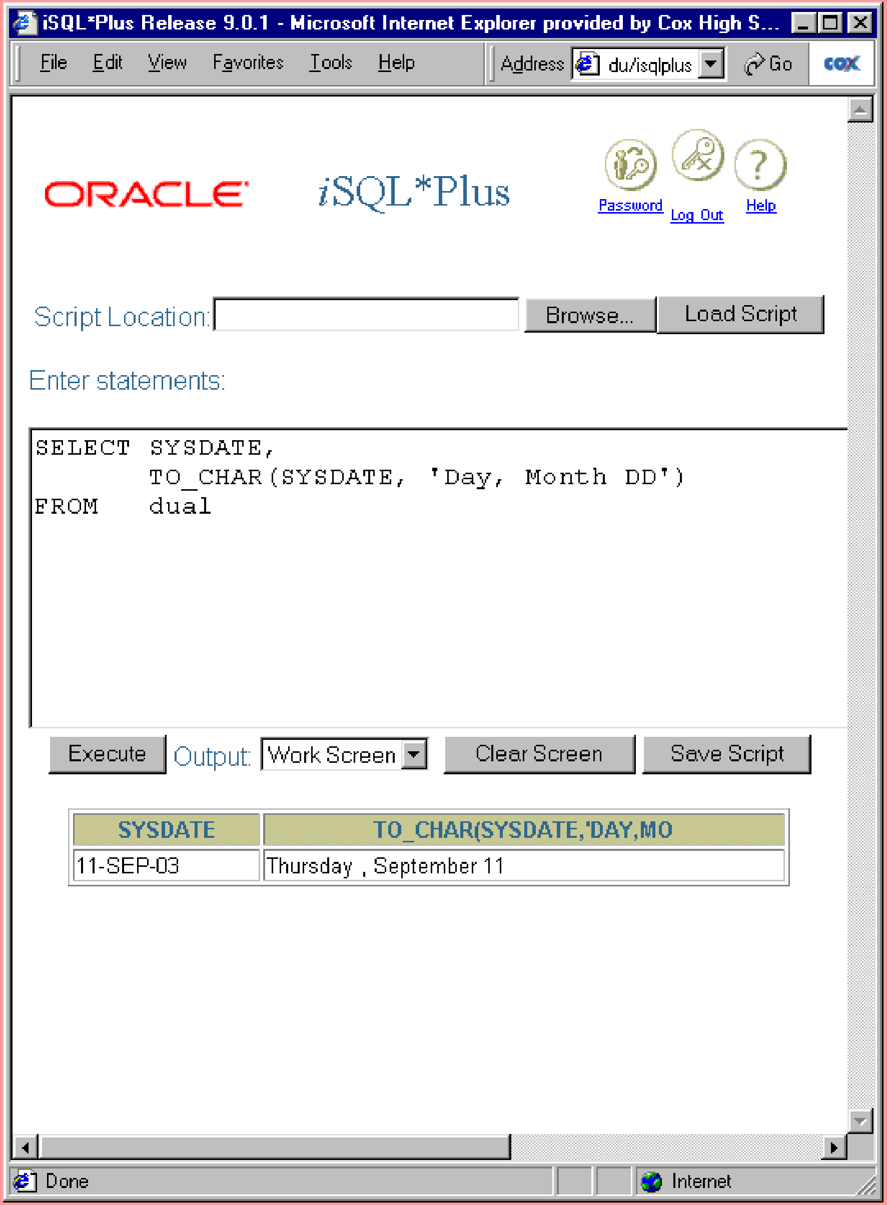
TO_DATE('	TO_DATE('	TO_DATE('	TO_DATE('
11-SEP-01	11-SEP-01	11-SEP-01	11-SEP-01

TO_DATE casts a character value as a datetime data type. The value must cast as a valid date, otherwise Oracle will throw an error.

TO_DATE(character value, format mask)

The character value carries the date information, the format mask describes how to interpret that information.

Page J-4: TO_CHAR



```
SELECT  SYSDATE,
        TO_CHAR(SYSDATE, 'Day, Month DD')
FROM    dual
```

SYSDATE	TO_CHAR(SYSDATE,'DAY,MO
11-SEP-03	Thursday , September 11

TO_CHAR converts a numeric or datetime value into a character value.

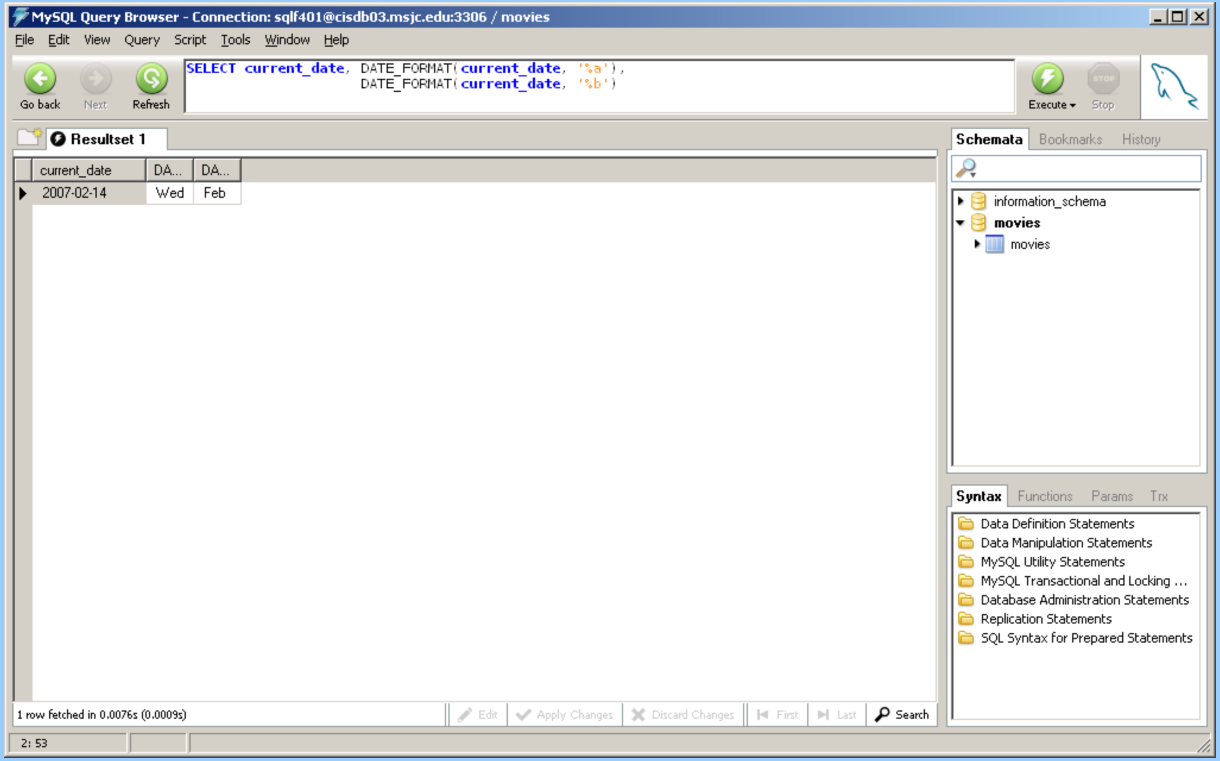
TO_CHAR(number)

TO_CHAR(datetime value, format mask)

Page J-5: DATE_FORMAT

MySQL provides a date formatting function that is analagous to Oracle’s TO_CHAR function: DATE_FORMAT.

This function can strip off ‘date parts’ and display them, and it can also format these ‘parts’. Each of these formats is defined with a percent sign, and many formats can be strung together.



```
SELECT current_date, DATE_FORMAT(current_date, '%a'),
                     DATE_FORMAT(current_date, '%b')
```

current_date	DA...	DA...
2007-02-14	Wed	Feb

1 row fetched in 0.0076s (0.0009s)

Module 09: Value Functions

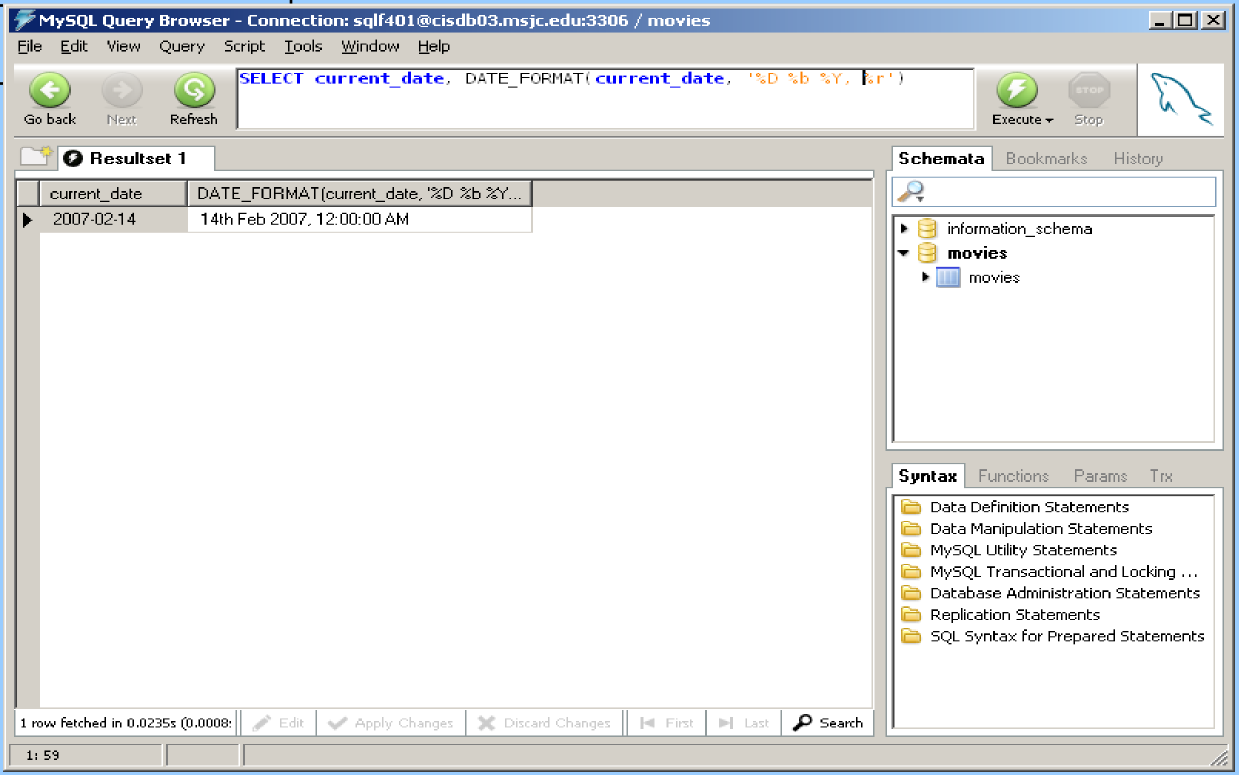
Page J-6: DATE_FORMAT

Specifier	Description
%a	Abbreviated weekday name (Sun..Sat)
%b	Abbreviated month name (Jan..Dec)
%c	Month, numeric (0..12)
%D	Day of the month with English suffix (0th, 1st, 2nd, 3rd, …)
%d	Day of the month, numeric (00..31)
%e	Day of the month, numeric (0..31)
%f	Microseconds (000000..999999)
%H	Hour (00..23)
%h	Hour (01..12)
%I	Hour (01..12)
%i	Minutes, numeric (00..59)
%j	Day of year (001..366)
%k	Hour (0..23)
%l	Hour (1..12)
%M	Month name (January..December)
%m	Month, numeric (00..12)
%p	AM or PM
%r	Time, 12-hour (hh:mm:ss followed by AM or PM)
%S	Seconds (00..59)
%s	Seconds (00..59)
%T	Time, 24-hour (hh:mm:ss)
%U	Week (00..53), where Sunday is the first day of the week
%u	Week (00..53), where Monday is the first day of the week
%V	Week (01..53), where Sunday is the first day of the week; used with %X
%v	Week (01..53), where Monday is the first day of the week; used with %x
%W	Weekday name (Sunday..Saturday)
%w	Day of the week (0=Sunday..6=Saturday)

Module 09: Value Functions

Page J-7: DATE_FORMAT

%X	Year for the week where Sunday is the first day of the week, numeric, four digits; used with %V
%x	Year for the week, where Monday is the first day of the week, numeric, four digits; used with %v
%Y	Year, numeric, four digits
%y	Year, numeric (two digits)
%%	A literal '%' character
%*x*	*x*, for any '*x*' not listed above


MySQL Query Browser - Connection: sqlf401@cisdb03.msjc.edu:3306 / movies
File Edit View Query Script Tools Window Help
Go back
Next
Refresh
Execute
Stop
Resultset 1
Schemata Bookmarks History
information_schema
movies
movies
Syntax Functions Params Trx
Data Definition Statements
Data Manipulation Statements
MySQL Utility Statements
MySQL Transactional and Locking ...
Database Administration Statements
Replication Statements
SQL Syntax for Prepared Statements
Edit
Apply Changes
Discard Changes
First
Last
Search
1: 59

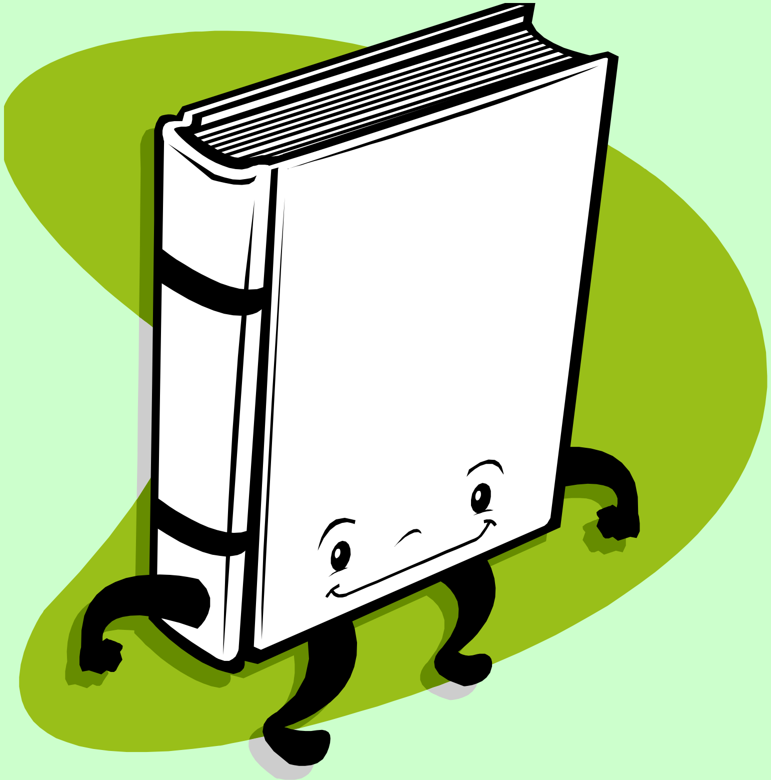
```
SELECT current_date, DATE_FORMAT(current_date, '%D %b %Y, %r')
```

current_date	DATE_FORMAT(current_date, '%D %b %Y...
2007-02-14	14th Feb 2007, 12:00:00 AM

1 row fetched in 0.0235s (0.0008:

©1998-2018 / Bergin

Module 09: Value Functions



Page T-1: Terminology

Value function
Numeric function, numeric value function
Character function, character value function
Date function, date value function

Function name
Function type
Parameter list, argument list

Nesting functions, nested functions

Casting

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