SQL Programming

Like is Like, Like, ya know?

Page A-1: Intro

One of the features of SQL that you'll hear me ranting about is the benefits attendant proper use of data types.

Number values behave like numbers should, you can add them and subtract them, you can multiply them and divide them… You can do number kinds of things with them.

Date values behave like date values ought to. January dates precede April dates, even though alphabetically April precedes January. It's cool that SQL knows the difference, that it knows what dates are.

Character values also behave like character values should. So far though, we've only seen that they can be smooshed together. But one of the coolest things we naturally do with characters, words, and letters, is to look for patterns.

Ever see the television program: Wheel of Fortune? It's a lot like the game of Hangman that we played as kids.

Page A-2: Intro

Pattern matching is something we do naturally with words and letters, and in SQL we have a powerful comparison operation that will do pattern searching on character data.

That comparison operation uses the LIKE operator.

And recently (at least recent for the SQL standard), a more powerful tool for pattern matching has been added to the language: regular expressions (regex).

Page B-1: Comparison Verbs

LIKE is a comparison operator or comparison *verb*.

My preference when talking about comparison operators such as LIKE and BETWEEN is to refer to them as verbs rather than as operators, because, in my mind, operators have a special symbol associated with them, and there is no special symbol for either the BETWEEN or the LIKE operations.

Addition has the plus sign, subtraction has the minus sign, and concatenation has the smooshing sign (vertical bars). As there aren't any symbols, or operators, for the BETWEEN or LIKE operations, I prefer to refer to them as verbs.

Page B-2: LIKE Verb

The LIKE verb is used primarily in the WHERE clause to test for the presence of a string pattern.

We can use LIKE to answer such exquisite pattern matching questions as:

Is there a vowel in the value that is stored in this column?

Does the LOC_code contain the character string ‘PA’?

Does this character value end with the string pattern ‘DOC’?

If the pattern is present in the target string, then the expression evaluates to TRUE.

Page B-3: LIKE Verb (cont)

LIKE employs two special characters, known as wildcards, to define a wide range of candidate target strings.

The underscore character (_) is used to represent a single character position of any value.

The percent character (%) is used to represent a string of any number of characters.

Page B-4: Examples

Assume that phone number information is stored in a single column in this format:

xxx.yyy.zzzz (909.487.6752)

area code.exchange.subscriber number

If we wanted to find all of the records in the 909 area code, we could use this LIKE phrase:

```
WHERE  phone_number LIKE ‘909.%’
```

The pattern 909.% will evaluate TRUE for any phone number that has ‘909.’ in the first four character positions.

Page B-5: Examples

If we were looking for any number in the 487 exchange (regardless of area code) we could try:

```
  WHERE phone_number LIKE '_ _ _.487.%'
```

or

```
   WHERE  phone_number LIKE '%.487.%'
```

In the case of the first solution:

```
   WHERE  phone_number LIKE '___.487.%'
```

this predicate is testing for: Any 3 characters, followed by dot 487 dot, followed by any number of characters.

In the case of:

```
    WHERE  phone_number LIKE '%.487.%'
```

the predicate is looking for any number of characters followed by dot 487 dot, followed by any number of characters.

Page B-6: Problem 7-1

Our talent agency assigns a rep to each of our clients. At present these assignments are based on the first letter of the client's last name.

All the A's go to Franka, the B's go to Harry, the C's go to Simone, …

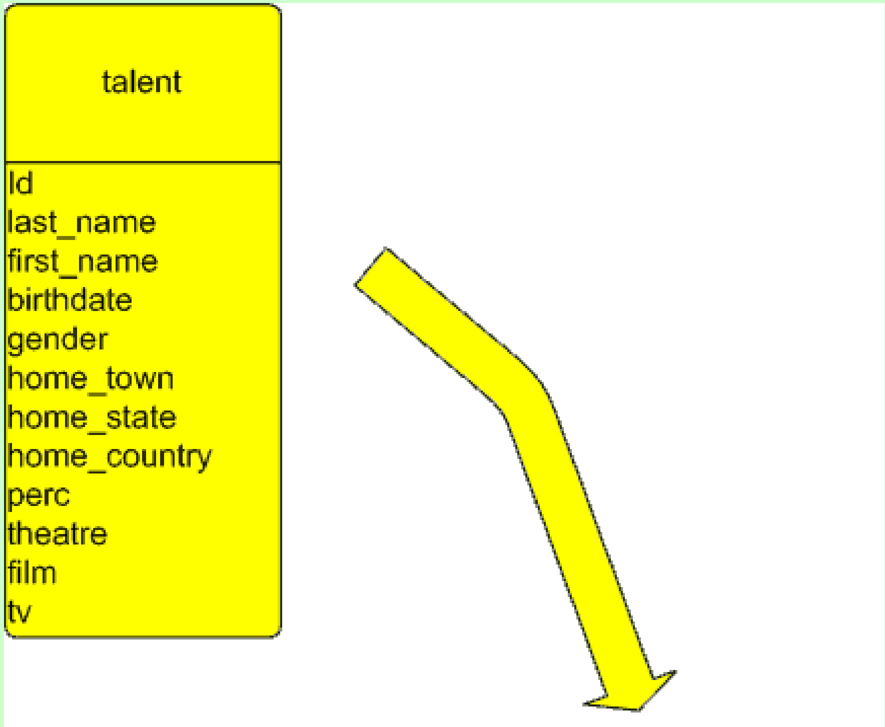
Eddy is handling all of the W's and he needs a listing of all of the available information for his client base.

Module 06: Character Data   Page B-7: Problem 7-1 Design & Code

Step 1: Build the Table Build Chart (TBC)

Step 2: Double check your TBC solution

Step 3: Transform the TBC into code.



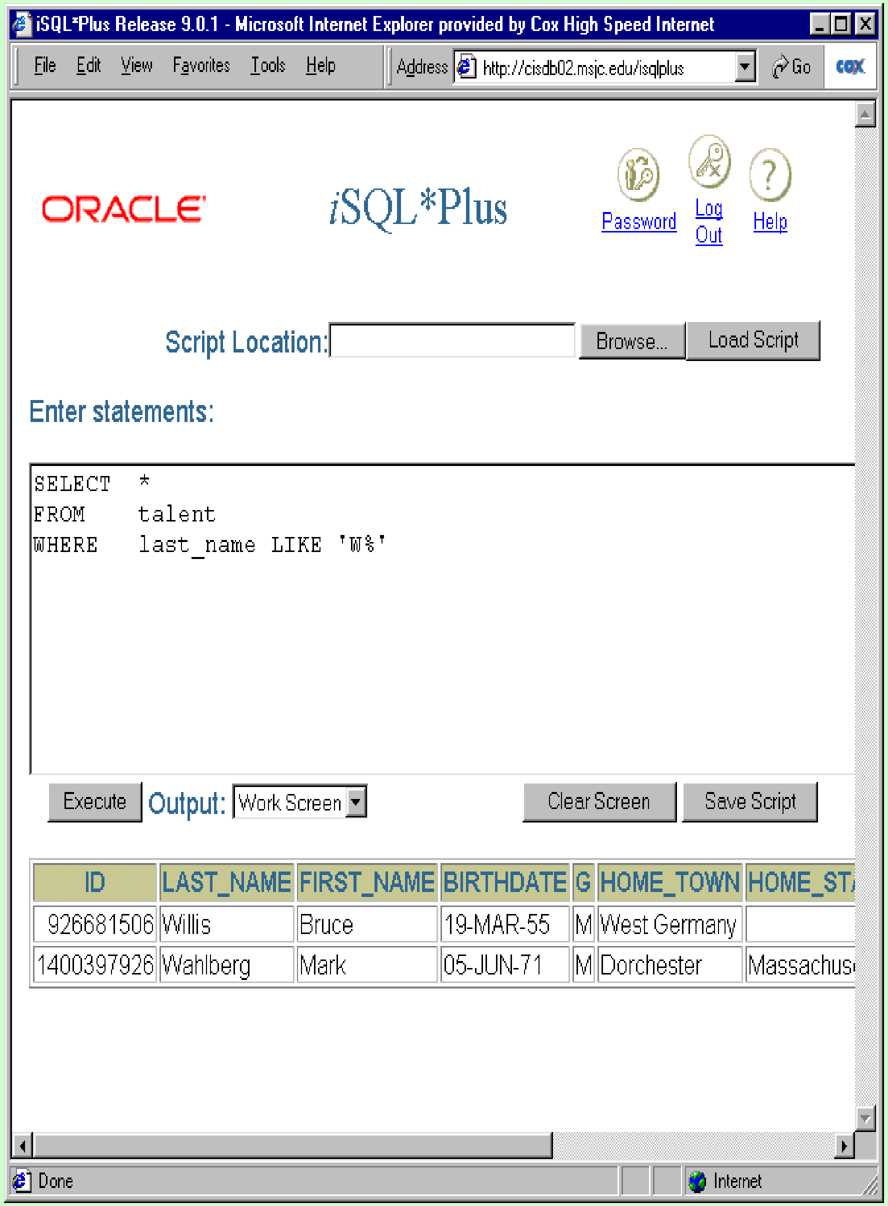
talent

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

Column Name/Expression	last_name	*
Table Name	talent	talent
Alias		
Criteria	LIKE 'W%'	...
Display		

```
SELECT   *
FROM     talent
WHERE  last_name LIKE ‘W%’
```

Page B-8: Problem 7-1 Analysis


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
Done
Internet

```
SELECT  *
FROM    talent
WHERE   last_name LIKE 'W%'
```

ID	LAST_NAME	FIRST_NAME	BIRTHDATE	G	HOME_TOWN	HOME_ST
926681506	Willis	Bruce	19-MAR-55	M	West Germany	
1400397926	Wahlberg	Mark	05-JUN-71	M	Dorchester	Massachus

Page B-9: Problem 7-2

Remember what the users told us about the rep assignments:

Our talent agency assigns a rep to each of our clients. At present these assignments are based on the first letter of the client's last name.

All the A's go to Franka, the B's go to Harry, the C's go to Simone, …

Turns out that description was only half-right. We have a special rep (Thalia) assigned to assist all of the foreign born clients. Otherwise the breakdown on rep assignments is as they said.

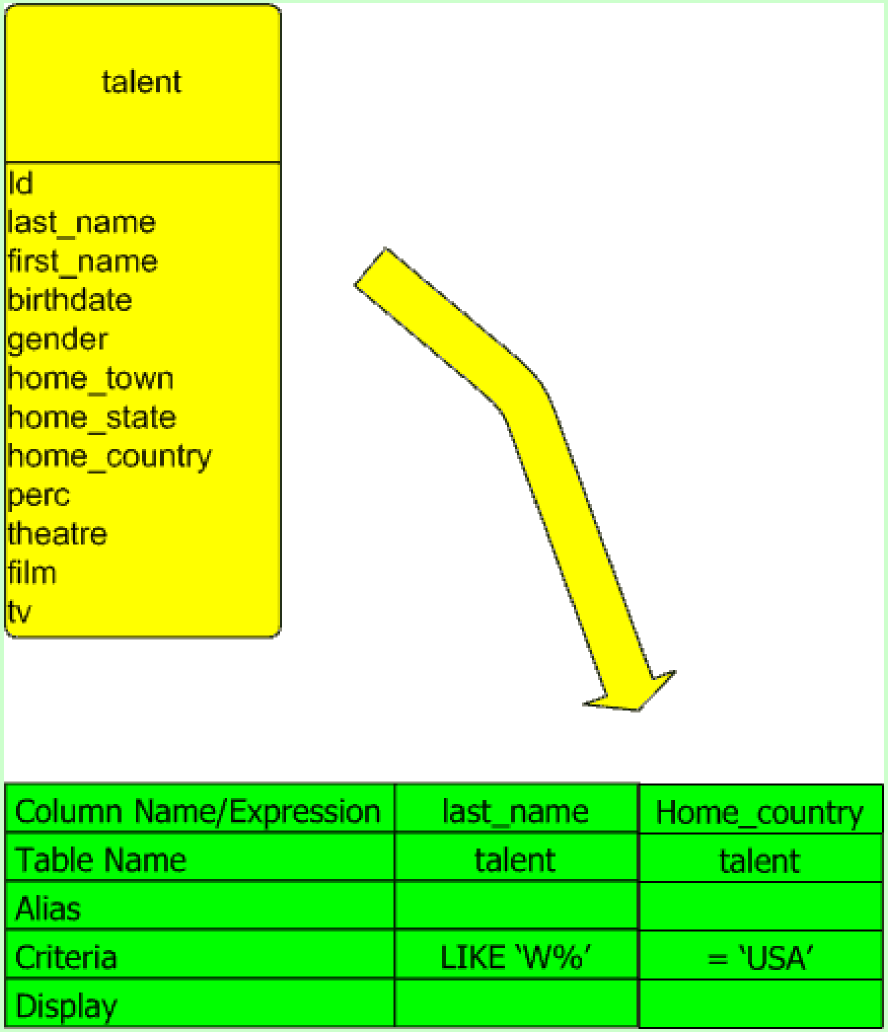
Eddy is handling all of the W's and he needs a listing of all of the available information for his client base.

Module 06: Character Data   Page B-10: Problem 7-2 Design & Code

Step 1: Build the Table Build Chart (TBC)

Step 2: Double check your TBC solution

Step 3: Transform the TBC into code.



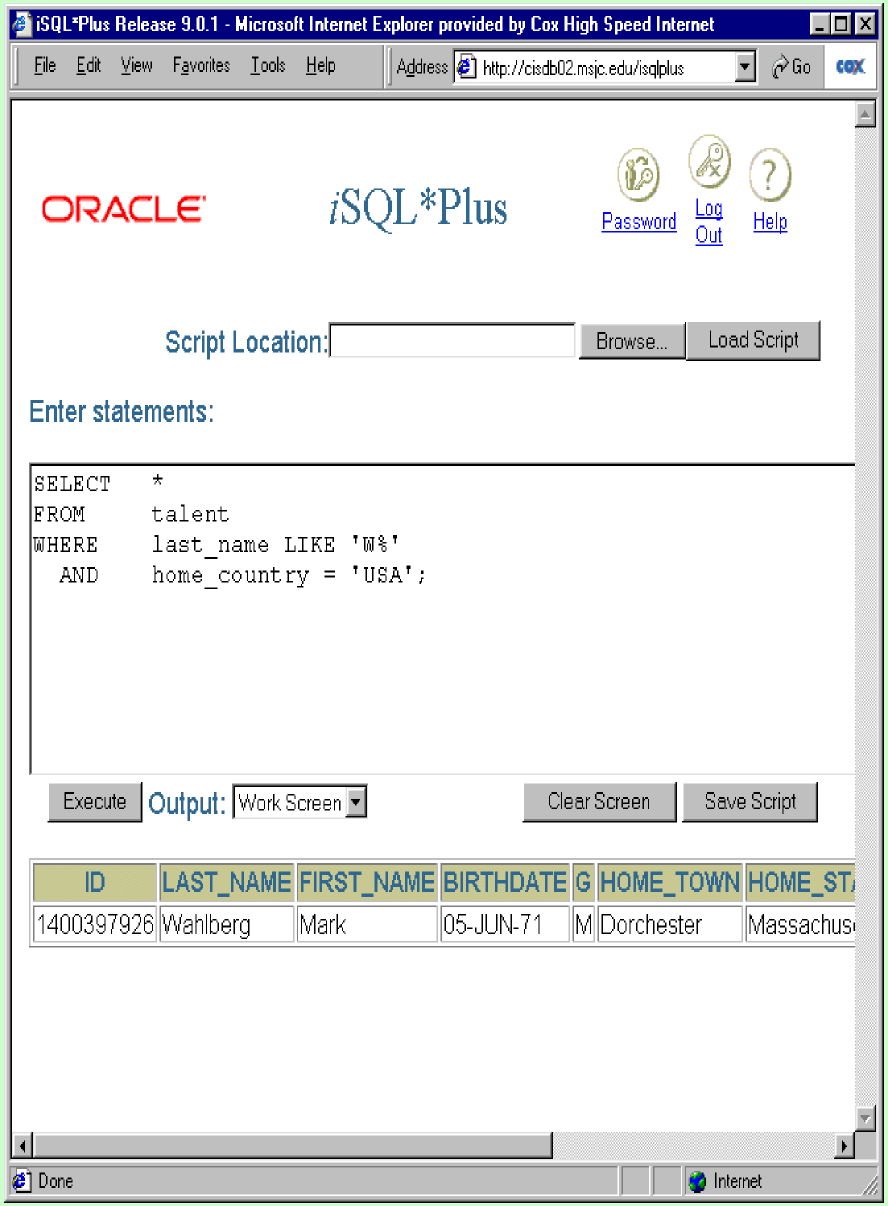
talent

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

Column Name/Expression	last_name	Home_country
Table Name	talent	talent
Alias		
Criteria	LIKE 'W%'	= 'USA'
Display		

```
SELECT   *
FROM     talent
WHERE  last_name LIKE ‘W%’
    AND  home_country = ‘USA’;
```

Page B-11: Problem 7-2 Analysis


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   *
FROM     talent
WHERE    last_name LIKE 'W%'
  AND    home_country = 'USA';
```

ID	LAST_NAME	FIRST_NAME	BIRTHDATE	G	HOME_TOWN	HOME_STA
1400397926	Wahlberg	Mark	05-JUN-71	M	Dorchester	Massachus

Page B-12: Problem 7-3

I’ve got one of the reps on the phone, and he’s in a bit of a tizzy. He just got chewed out by one of our clients who hung up before he could confirm their name.

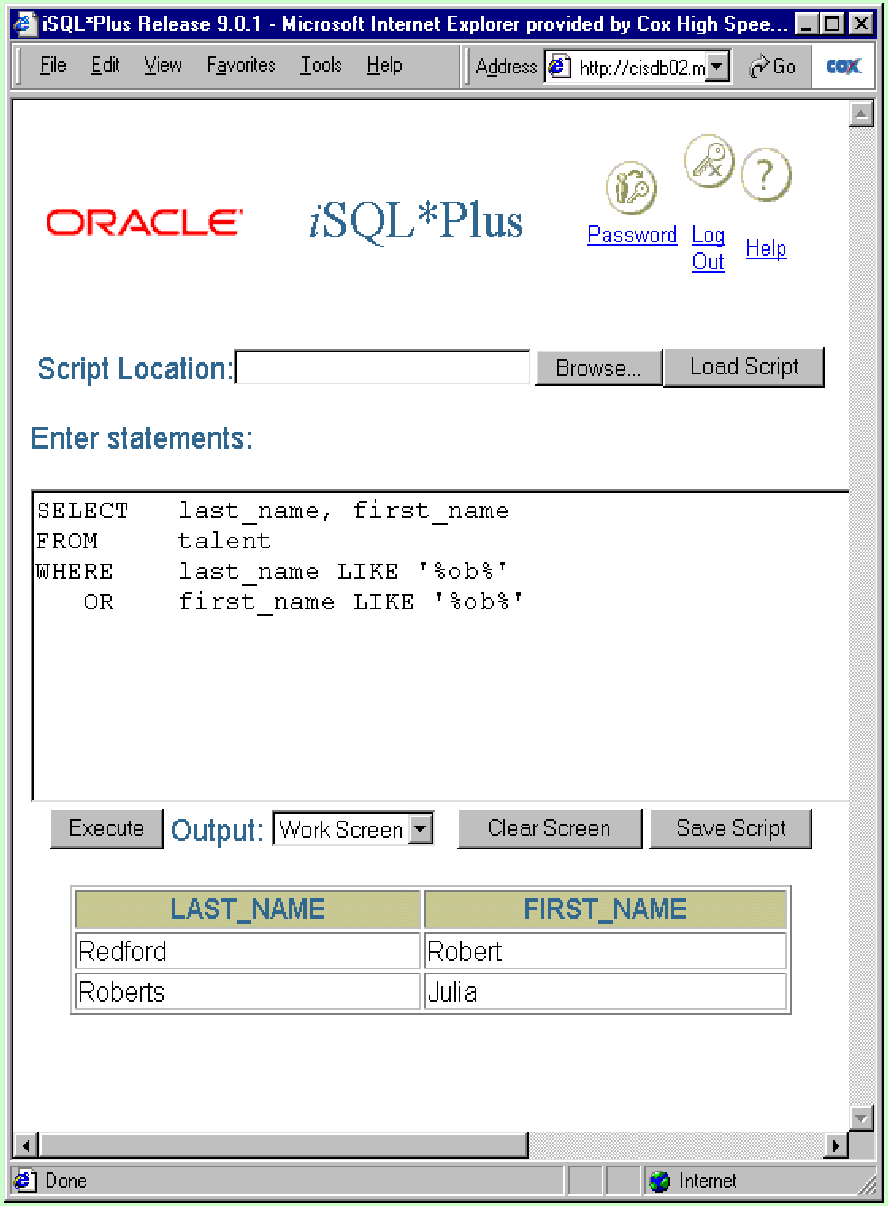
It was a guy. And he thinks the name was something like Rob, Bob, Cobb, Hobbs, … something like that.

Can we search the database for any client with a name that sounds like that?

Rephrased:

Do a search on first_names and last_names (just in case) for the pattern ‘OB’

Page B-13: Problem 7-3 Analysis

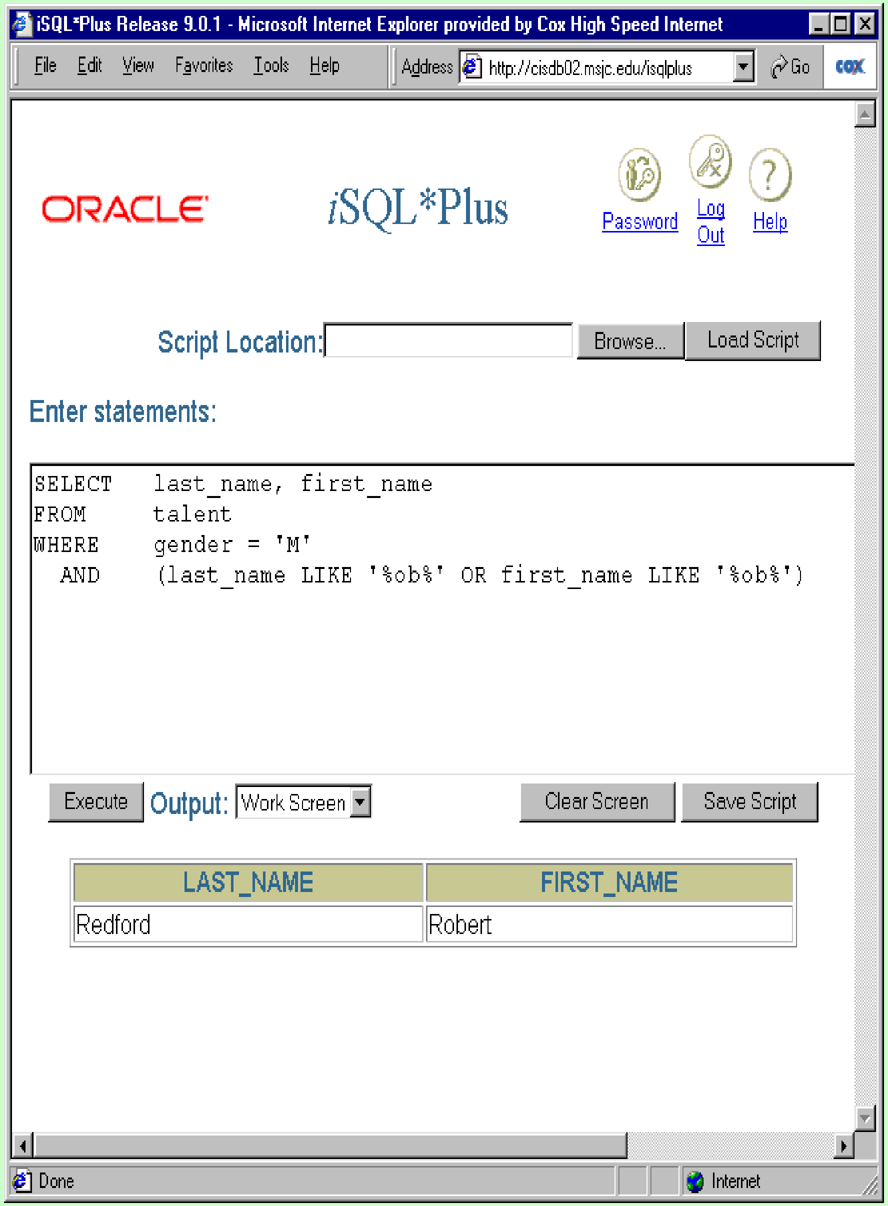


```
SELECT   last_name, first_name
FROM     talent
WHERE    last_name LIKE '%ob%'
   OR    first_name LIKE '%ob%'
```

LAST_NAME	FIRST_NAME
Redford	Robert
Roberts	Julia

He did say he thought it was a guy?

Page B-14: Problem 7-3 Redux


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:

```
SELECT   last_name, first_name
FROM     talent
WHERE    gender = 'M'
  AND    (last_name LIKE '%ob%' OR first_name LIKE '%ob%')
```

Execute
Output: Work Screen
Clear Screen
Save Script

LAST_NAME	FIRST_NAME
Redford	Robert

There, that's more better. ☺

Page B-15: Problem 7-4

There’s an office pool, $5 an entry, winner take all, for the name of the client with the shortest first name.

You don’t have any qualms about misappropriating company resources for a private wager, so you plan to write a SQL program to find any clients whose first names are either 1 or 2 characters long.

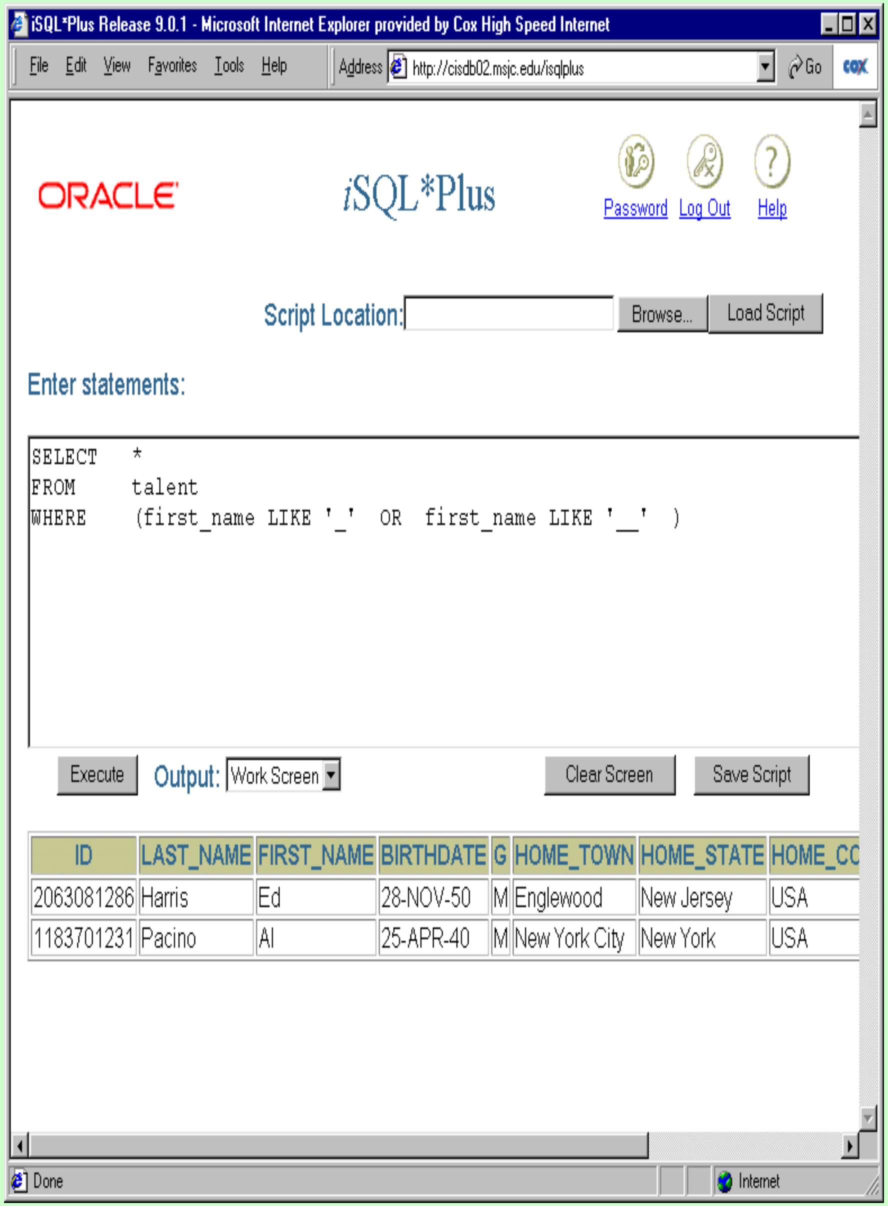
Page B-16: Problem 7-4 Design & Code

You have read the company Acceptable Use Policy (AUP)?

A prank like this could cost you your job!

You’ve been warned!

Page B-17: Problem 7-4 Analysis


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   *
FROM     talent
WHERE    (first_name LIKE '_'  OR  first_name LIKE '__'  )
```

ID	LAST_NAME	FIRST_NAME	BIRTHDATE	G	HOME_TOWN	HOME_STATE	HOME_CC
2063081286	Harris	Ed	28-NOV-50	M	Englewood	New Jersey	USA
1183701231	Pacino	Al	25-APR-40	M	New York City	New York	USA

The first condition uses LIKE to test for a single underscore character (1 letter name). The second condition uses two underscores in the test.

Page C-1: General Form for NOT

Let's go back a couple of modules to the discussion about Boolean operators AND, OR, and NOT.

You should remember that the general form for negating a condition is:

```
            NOT (comparison operation)
```

For example

```
            NOT (perc < 8)
            NOT (home_country = 'USA')
```

So, if we want to negate a LIKE comparison it would be written in the form:

```
        NOT  (value LIKE target string)
```

For example

```
        NOT (home_country LIKE '%US%')
```

Page C-2: Special Case for NOT

SQL strives to be user friendly, and apparently the SQL gurus thought that reading a phrase such as:

NOT (home_country LIKE ‘%US%’)

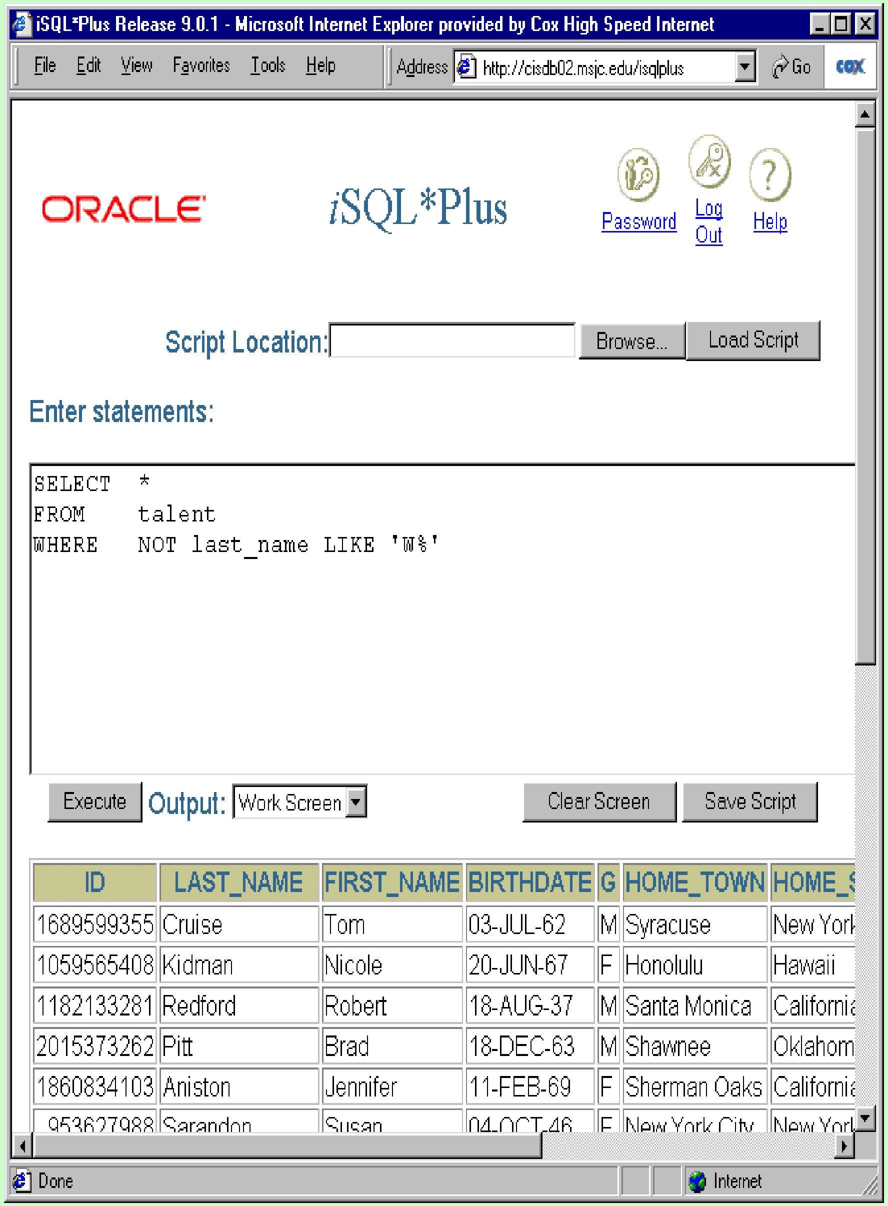
might be a little cumbersome.

So, to improve readability, in only a few instances, SQL permits the NOT operator to be inserted in the middle of a conditional expression.

LIKE is one of these special comparison verbs that allows NOT to be inserted in the middle of the comparison.

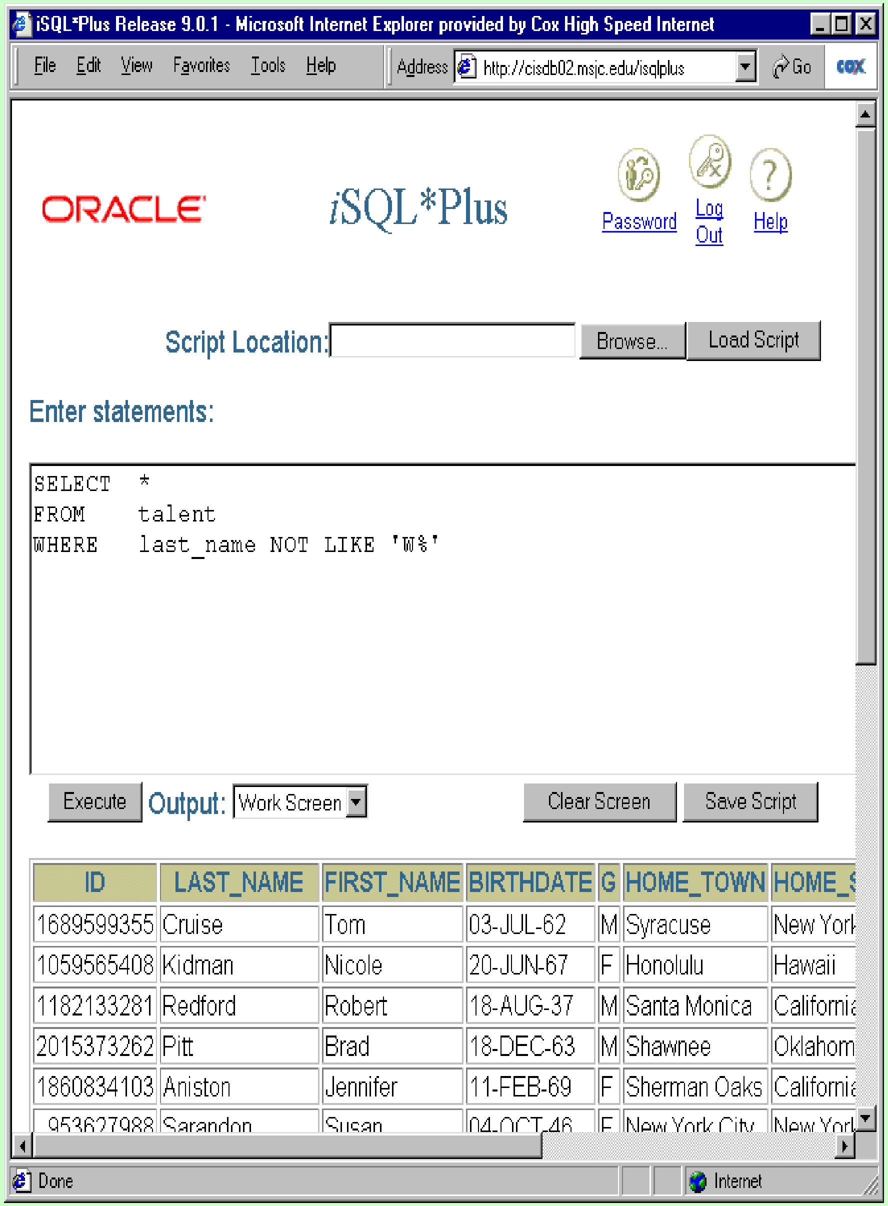
Module 06: Character Data

Page C-3: Special Case Example


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  *
FROM    talent
WHERE   NOT last_name LIKE 'W%'
```

ID	LAST_NAME	FIRST_NAME	BIRTHDATE	G	HOME_TOWN	HOME_S
1689599355	Cruise	Tom	03-JUL-62	M	Syracuse	New Yorl
1059565408	Kidman	Nicole	20-JUN-67	F	Honolulu	Hawaii
1182133281	Redford	Robert	18-AUG-37	M	Santa Monica	Californi
2015373262	Pitt	Brad	18-DEC-63	M	Shawnee	Oklahom
1860834103	Aniston	Jennifer	11-FEB-69	F	Sherman Oaks	Californi
953627988	Sarandon	Susan	04-OCT-46	F	New York City	New Yorl


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  *
FROM    talent
WHERE   last_name NOT LIKE 'W%'
```

ID	LAST_NAME	FIRST_NAME	BIRTHDATE	G	HOME_TOWN	HOME_S
1689599355	Cruise	Tom	03-JUL-62	M	Syracuse	New Yorl
1059565408	Kidman	Nicole	20-JUN-67	F	Honolulu	Hawaii
1182133281	Redford	Robert	18-AUG-37	M	Santa Monica	Californi
2015373262	Pitt	Brad	18-DEC-63	M	Shawnee	Oklahom
1860834103	Aniston	Jennifer	11-FEB-69	F	Sherman Oaks	Californi
953627988	Sarandon	Susan	04-OCT-46	F	New York City	New Yorl

Page D-1: Regular Expressions

As powerful and as flexible as these pattern matching characters (%, _) are, recent enhancements to the SQL standard now provide a more powerful tool for the programmer: regular expressions.

Regular expressions have been around for decades, and one of the earliest implementations of regular expressions (regex) was in the Unix editor *ed*. Since that time, regex have been a staple in Unix, and Unix-like systems, and have been incorporated into almost all modern programming languages: Java, C++, C#, python, …

As I work through these concepts, I'll be demonstrating the capabilities of regex with a simple Unix tool known as egrep, (egrep is an acronym for Extended Global Regular Expression Print). egrep is a little bit quicker to use than having to write a SQL program each time I want to demonstrate a feature, so I'll use that for a few of these examples.

Note: you have access to the same unix system that I'm using, and I'd encourage you to 'follow along' and practice these techniques on that platform. Once you've mastered the rudiments, you'll be able to apply them to SQL much more easily.

Page D-2: Regular Expressions

Regular expressions describe a pattern of characters, or a sequence of characters.

For example, the regular expression ‘dogs’ will match every sequence of characters that starts with ‘d’ followed immediately by ‘o’, followed immediately by ‘g’, followed immediately by ‘s’.

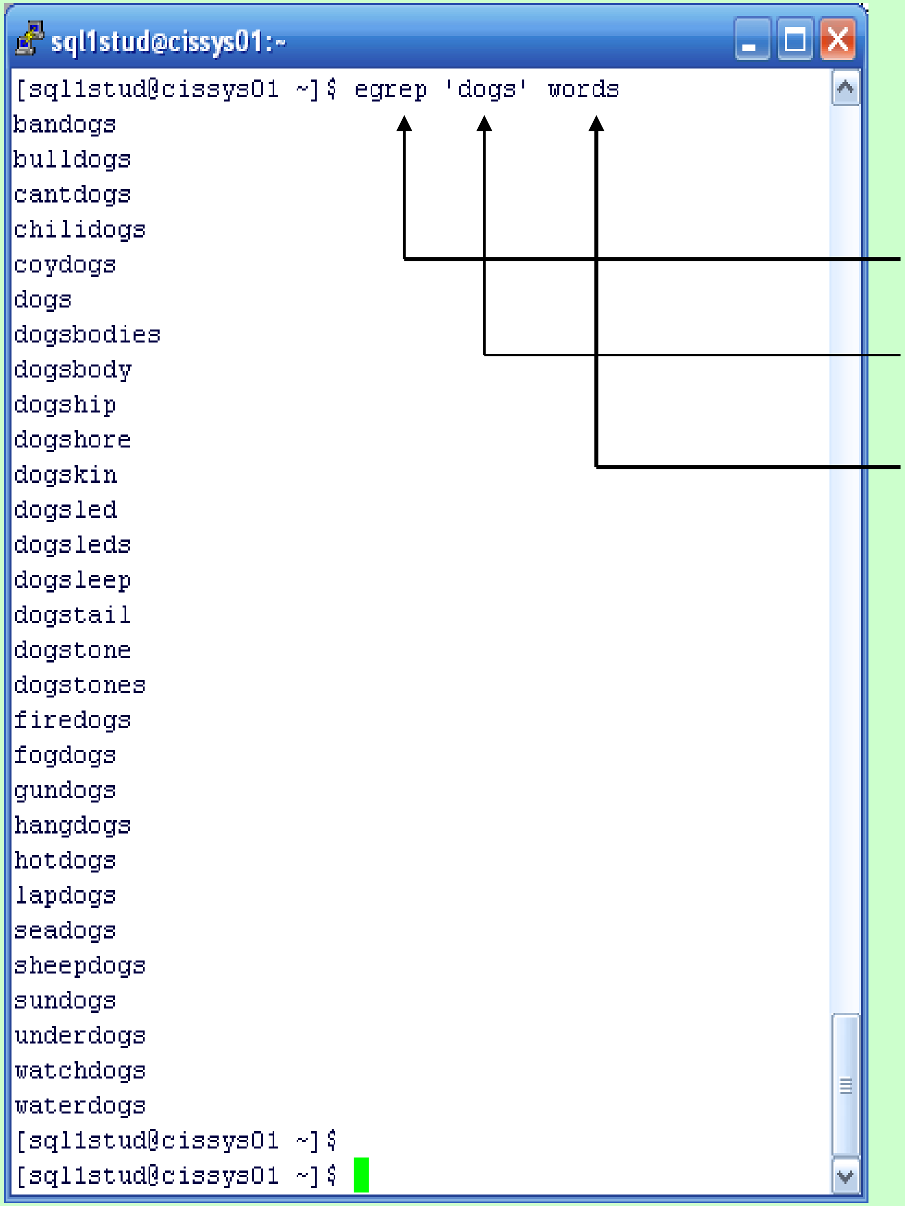
The egrep program uses (evaluates) regular expressions when examining a file, and prints out any lines that it encounters that match.

The egrep invocation:

```
egrep ‘dogs’ words
```

will examine the file named ‘words’, and display every line therein that contains the character sequence ‘dogs’, that is, it will print out every line that has a ‘d’ followed immediately by ‘o’, followed immediately by ‘g’, followed immediately by ‘s’...

Page D-3: Regular Expressions



```
[sql1stud@cissys01 ~]$ egrep 'dogs' words
bandogs
bulldogs
cantdogs
chilidogs
coydogs
dogs
dogsbodies
dogsbody
dogship
dogshore
dogskin
dogsled
dogsleds
dogsleep
dogstail
dogstone
dogstones
firedogs
fogdogs
gundogs
hangdogs
hotdogs
lapdogs
seadogs
sheepdogs
sundogs
underdogs
watchdogs
waterdogs
[sql1stud@cissys01 ~]$
[sql1stud@cissys01 ~]$
```

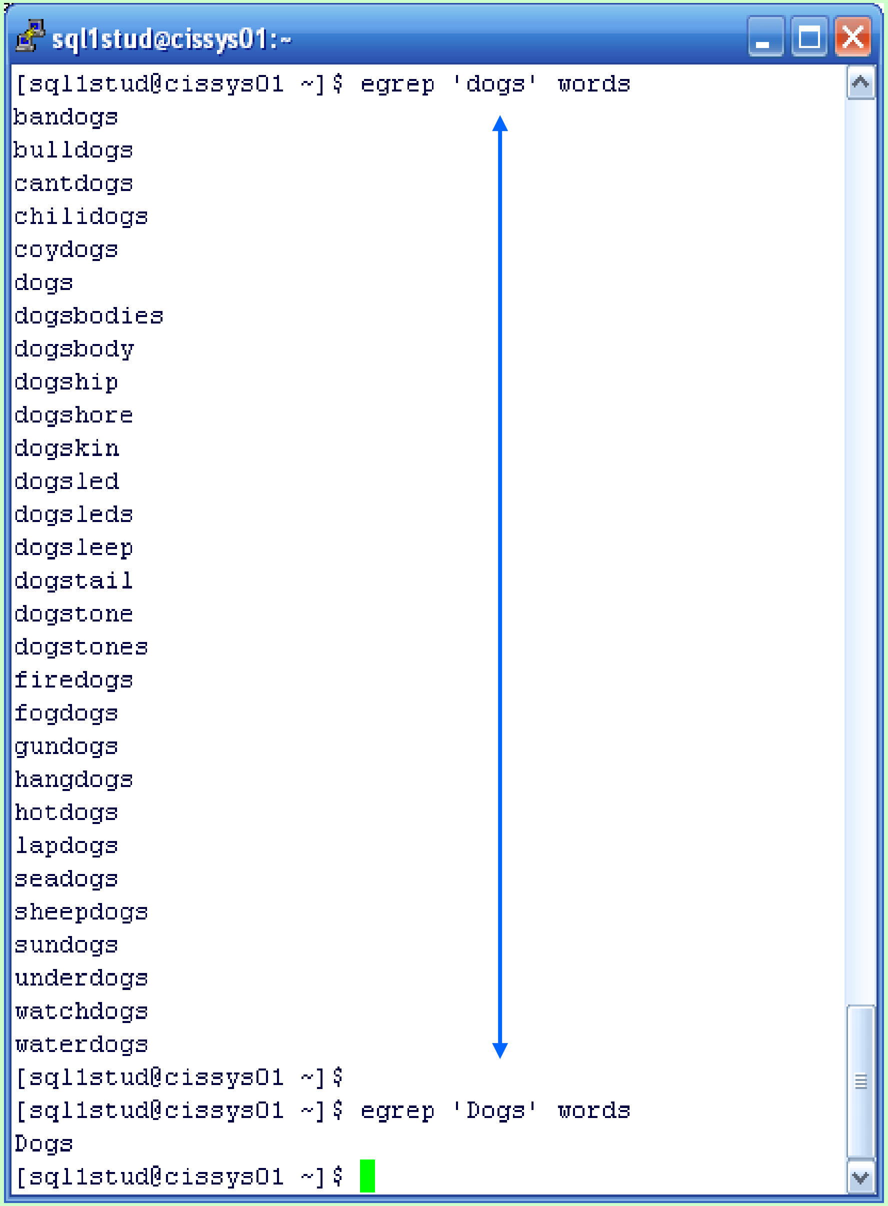
egrep: name of the utility program

‘dogs’: the regular expression (in quotes)

words: the file that will be examined – in this case, *words* is a file that contains words you would find in a dictionary

As you can see from the example, any line from the words file that has ‘dogs’ in it is identified. It doesn’t matter if ‘dogs’ occurs at the beginning of the line, or whether it appears in the middle or the end of the line. As long as that character sequence is present somewhere in that line of text, egrep will find it.

Page D-4: Regular Expressions

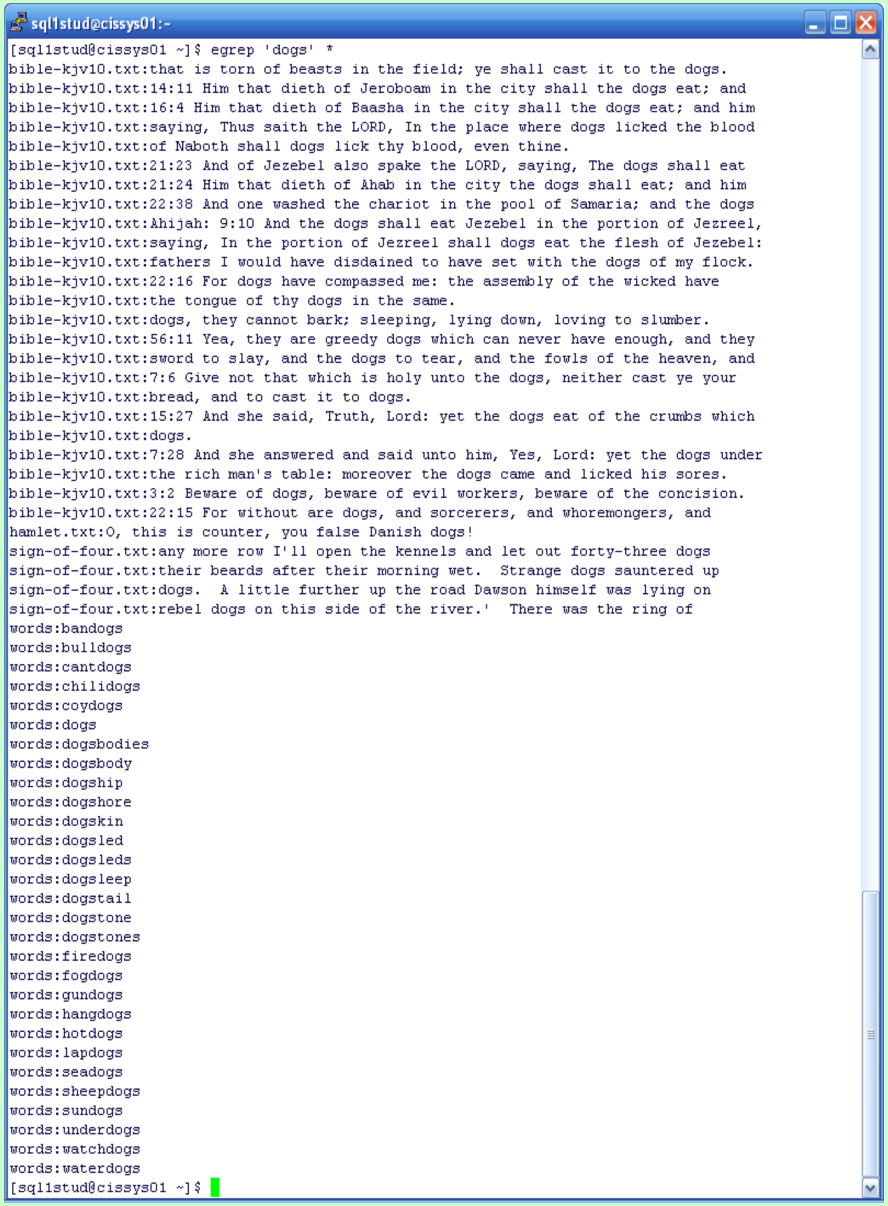


```
[sql1stud@cissys01 ~]$ egrep 'dogs' words
bandogs
bulldogs
cantdogs
chilidogs
coydogs
dogs
dogsbodies
dogsbody
dogship
dogshore
dogskin
dogsled
dogsleds
dogsleep
dogstail
dogstone
dogstones
firedogs
fogdogs
gundogs
hangdogs
hotdogs
lapdogs
seadogs
sheepdogs
sundogs
underdogs
watchdogs
waterdogs
[sql1stud@cissys01 ~]$
[sql1stud@cissys01 ~]$ egrep 'Dogs' words
Dogs
[sql1stud@cissys01 ~]$
```

dogs is an example of a “literal” regular expression, and it will match all character sequences that are literally (ie., exactly) the same as ‘dogs’.

In this regard, you should note that regular expressions are case-sensitive, hence ‘Dogs’ is not equivalent to ‘dogs’.

Page D-5: Regular Expressions



```
[sql1stud@cissys01 ~]$ egrep 'dogs' *
bible-kjv10.txt:that is torn of beasts in the field; ye shall cast it to the dogs.
bible-kjv10.txt:14:11 Him that dieth of Jeroboam in the city shall the dogs eat; and
bible-kjv10.txt:16:4 Him that dieth of Baasha in the city shall the dogs eat; and him
bible-kjv10.txt:saying, Thus saith the LORD, In the place where dogs licked the blood
bible-kjv10.txt:of Naboth shall dogs lick thy blood, even thine.
bible-kjv10.txt:21:23 And of Jezebel also spake the LORD, saying, The dogs shall eat
bible-kjv10.txt:21:24 Him that dieth of Ahab in the city the dogs shall eat; and him
bible-kjv10.txt:22:38 And one washed the chariot in the pool of Samaria; and the dogs
bible-kjv10.txt:Ahijah: 9:10 And the dogs shall eat Jezebel in the portion of Jezreel,
bible-kjv10.txt:saying, In the portion of Jezreel shall dogs eat the flesh of Jezebel:
bible-kjv10.txt:fathers I would have disdained to have set with the dogs of my flock.
bible-kjv10.txt:22:16 For dogs have compassed me: the assembly of the wicked have
bible-kjv10.txt:the tongue of thy dogs in the same.
bible-kjv10.txt:dogs, they cannot bark; sleeping, lying down, loving to slumber.
bible-kjv10.txt:56:11 Yea, they are greedy dogs which can never have enough, and they
bible-kjv10.txt:sword to slay, and the dogs to tear, and the fowls of the heaven, and
bible-kjv10.txt:7:6 Give not that which is holy unto the dogs, neither cast ye your
bible-kjv10.txt:bread, and to cast it to dogs.
bible-kjv10.txt:15:27 And she said, Truth, Lord: yet the dogs eat of the crumbs which
bible-kjv10.txt:dogs.
bible-kjv10.txt:7:28 And she answered and said unto him, Yes, Lord: yet the dogs under
bible-kjv10.txt:the rich man's table: moreover the dogs came and licked his sores.
bible-kjv10.txt:3:2 Beware of dogs, beware of evil workers, beware of the concision.
bible-kjv10.txt:22:15 For without are dogs, and sorcerers, and whoremongers, and
hamlet.txt:O, this is counter, you false Danish dogs!
sign-of-four.txt:any more row I'll open the kennels and let out forty-three dogs
sign-of-four.txt:their beards after their morning wet.  Strange dogs sauntered up
sign-of-four.txt:dogs.  A little further up the road Dawson himself was lying on
sign-of-four.txt:rebel dogs on this side of the river.'  There was the ring of
words:bandogs
words:bulldogs
words:cantdogs
words:chilidogs
words:coydogs
words:dogs
words:dogsbodies
words:dogsbody
words:dogship
words:dogshore
words:dogskin
words:dogsled
words:dogsleds
words:dogsleep
words:dogstail
words:dogstone
words:dogstones
words:firedogs
words:fogdogs
words:gundogs
words:hangdogs
words:hotdogs
words:lapdogs
words:seadogs
words:sheepdogs
words:sundogs
words:underdogs
words:watchdogs
words:waterdogs
[sql1stud@cissys01 ~]$
```

Here's another example that uses that same literal text pattern, but this time we're telling egrep to examine all of the files that are available in the local directory (in this file folder).

Since a number of different files are being examined, egrep modifies its behavior somewhat, and displays the file name at the start of each line of output.

Do you have your x-ray glasses? Let me try to explain the output from this example.

The file, bible-kjv10 (a King James version of the bible), contains about 24 lines that include the string 'dogs'. This means that in the entire file (the whole of the Bible) then, there are only 24 occurrences of 'dogs'.

The string pattern 'dogs' appears on only one line in the text of Shakespeare's Hamlet.

And this same string pattern occurs on only four lines of text from the Sherlock Holmes story: The Sign of Four.

Page D-6: Regular Expressions

These past few examples have been somewhat simple in that they used simple regex patterns to match *literal* text.

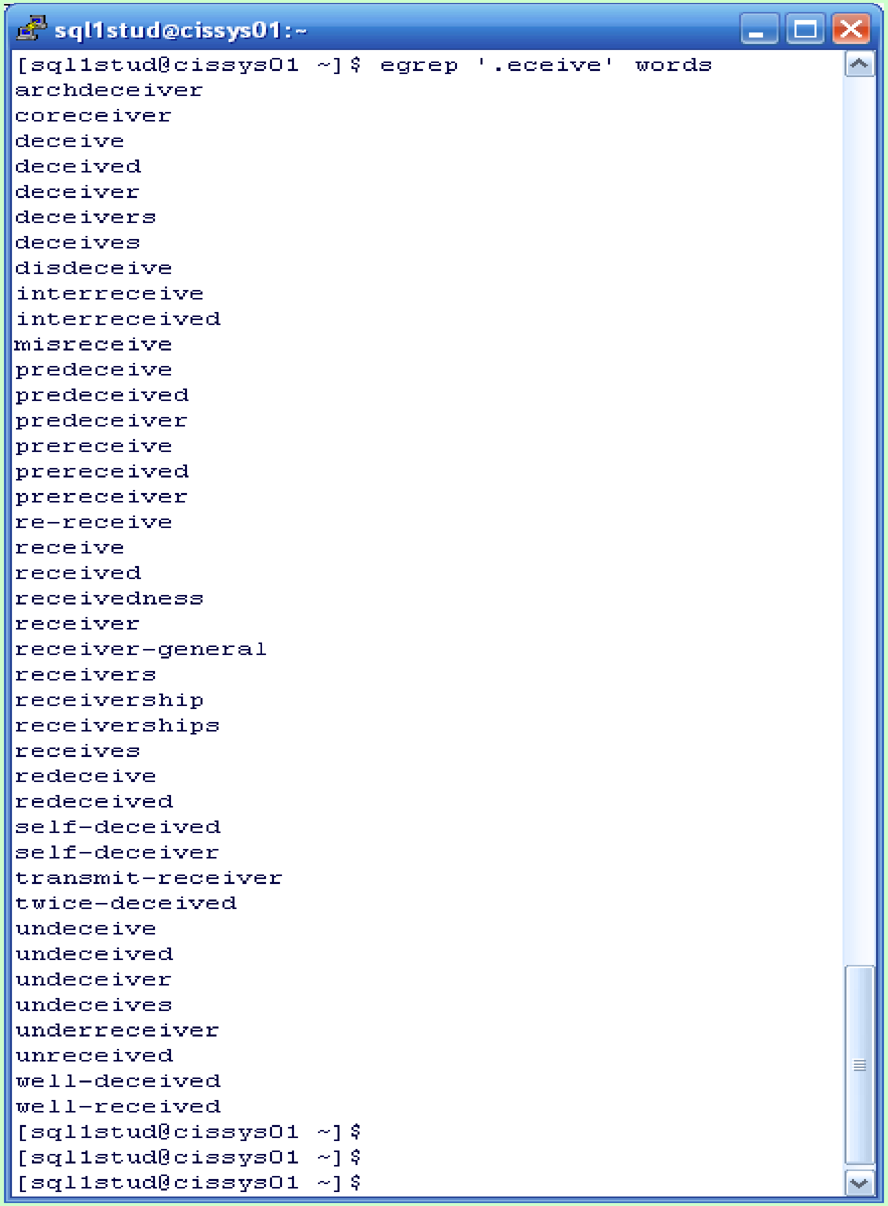
Their simplicity lies in the fact that they are ‘static’ and use unchanging string patterns.

But the real power of regex lies in its ability to use non-static patterns to search for string values.

To accomplish these more sophisticated searches, regex uses a set of special characters to build very complex, and very flexible pattern templates.

Module 06: Character Data

Page D-7: dot (.)



```
[sql1stud@cissys01 ~]$ egrep '.eceive' words
archdeceiver
corereceiver
deceive
deceived
deceiver
deceivers
deceives
disdeceive
interreceive
interreceived
misreceive
predeceive
predeceived
predeceiver
prereceive
prereceived
prereceiver
re-receive
receive
received
receivedness
receiver
receiver-general
receivers
receivership
receiverships
receives
redeceive
redeceived
self-deceived
self-deceiver
transmit-receiver
twice-deceived
undeceive
undeceived
undeceiver
undeceives
underreceiver
unreceived
well-deceived
well-received
[sql1stud@cissys01 ~]$
[sql1stud@cissys01 ~]$
[sql1stud@cissys01 ~]$
```

The dot (.) special character is used to denote any character. In this sense, the dot (.) is analogous to the underscore (_) that we saw earlier in SQL's like-expressions.

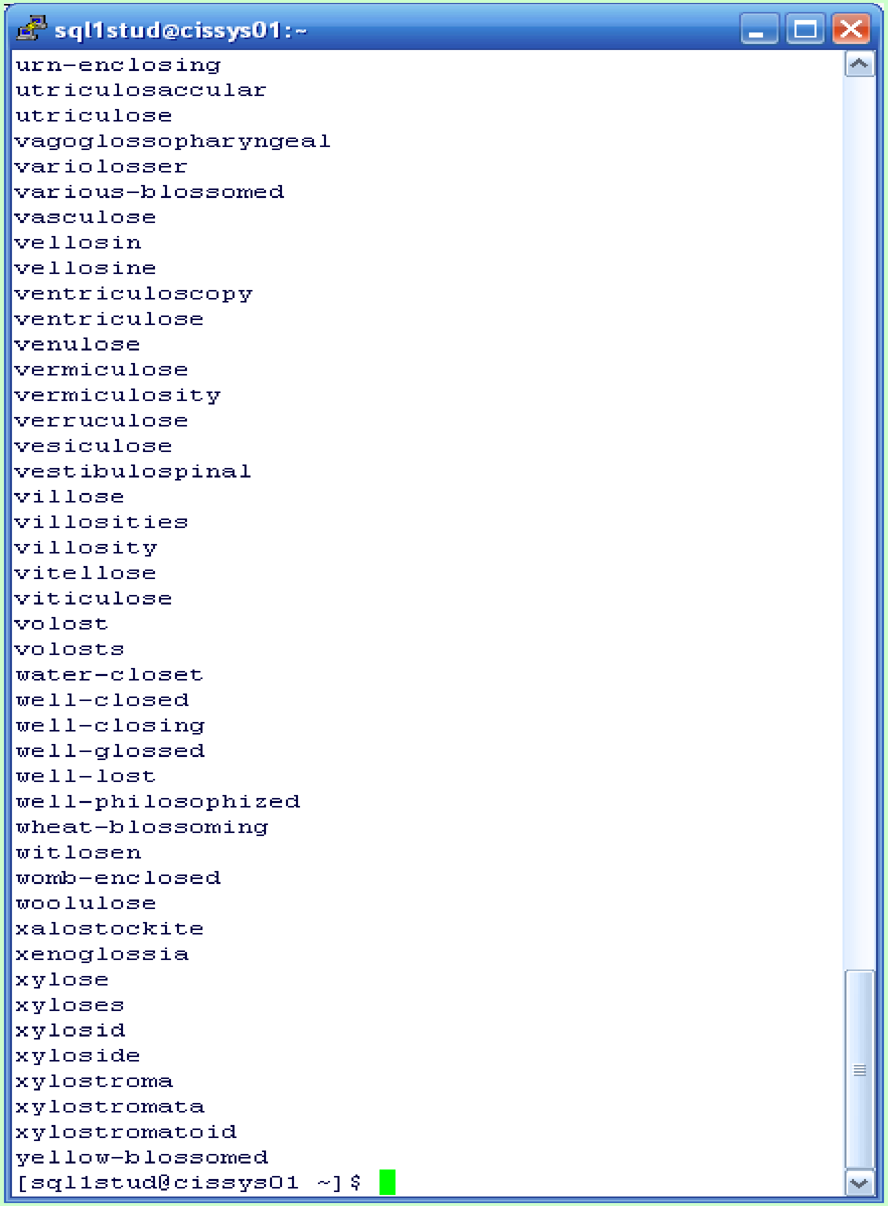
egrep '.eceive' words

This will find lines in the words file (ie. the file of dictionary words) that include any character, immediately followed by the literal pattern 'eceive'.

You can expect to see 'deceive' and 'receive' in the output. And you can also expect to see 'receivership', as well as any other line that includes this character pattern.

Module 06: Character Data

Page D-8: dot (.)



```
urn-enclosing
utriculosaccular
utriculose
vagoglossopharyngeal
variolosser
various-blossomed
vasculose
vellosin
vellosine
ventriculoscopy
ventriculose
venulose
vermiculose
vermiculosity
verruculose
vesiculose
vestibulospinal
villose
villosities
villosity
vitellose
viticulose
volost
volosts
water-closet
well-closed
well-closing
well-glossed
well-lost
well-philosophized
wheat-blossoming
witlosen
womb-enclosed
woolulose
xalostockite
xenoglossia
xylose
xyloses
xylosid
xyloside
xylostroma
xylostromata
xylostromatoid
yellow-blossomed
[sql1stud@cissys01 ~]$
```

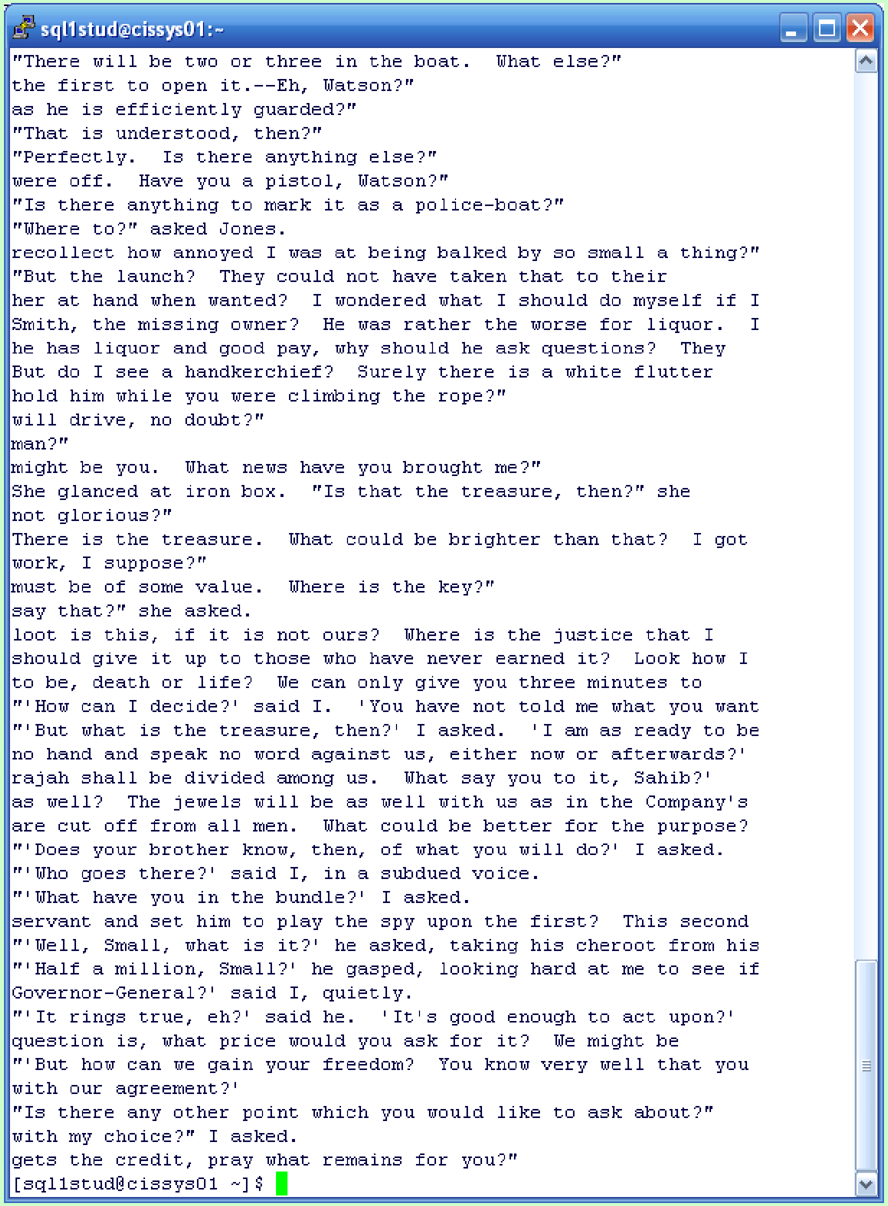
egrep 'los.' words

This will find lines in the words file that include the literal text pattern 'los' followed immediately by any other character.

You can expect to see items like 'lose' and 'lost', as well as 'floss', 'los angeles' (spaces count as characters), 'closed', and any other line with the pattern 'l' immediately followed by 'o' immediately followed by 's' immediately followed by some other character.

You would NOT see lines from the dictionary file for 'Carlos' or 'silos' because these words end in 'los' and the pattern is looking for 'los' PLUS some other character.

Page D-9: escape (\)



```
"There will be two or three in the boat.  What else?"
the first to open it.--Eh, Watson?"
as he is efficiently guarded?"
"That is understood, then?"
"Perfectly.  Is there anything else?"
were off.  Have you a pistol, Watson?"
"Is there anything to mark it as a police-boat?"
"Where to?" asked Jones.
recollect how annoyed I was at being balked by so small a thing?"
"But the launch?  They could not have taken that to their
her at hand when wanted?  I wondered what I should do myself if I
Smith, the missing owner?  He was rather the worse for liquor.  I
he has liquor and good pay, why should he ask questions?  They
But do I see a handkerchief?  Surely there is a white flutter
hold him while you were climbing the rope?"
will drive, no doubt?"
man?"
might be you.  What news have you brought me?"
She glanced at iron box.  "Is that the treasure, then?" she
not glorious?"
There is the treasure.  What could be brighter than that?  I got
work, I suppose?"
must be of some value.  Where is the key?"
say that?" she asked.
loot is this, if it is not ours?  Where is the justice that I
should give it up to those who have never earned it?  Look how I
to be, death or life?  We can only give you three minutes to
"'How can I decide?' said I.  'You have not told me what you want
"'But what is the treasure, then?' I asked.  'I am as ready to be
no hand and speak no word against us, either now or afterwards?'
rajah shall be divided among us.  What say you to it, Sahib?'
as well?  The jewels will be as well with us as in the Company's
are cut off from all men.  What could be better for the purpose?
"'Does your brother know, then, of what you will do?' I asked.
"'Who goes there?' said I, in a subdued voice.
"'What have you in the bundle?' I asked.
servant and set him to play the spy upon the first?  This second
"'Well, Small, what is it?' he asked, taking his cheroot from his
"'Half a million, Small?' he gasped, looking hard at me to see if
Governor-General?' said I, quietly.
"'It rings true, eh?' said he.  'It's good enough to act upon?'
question is, what price would you ask for it?  We might be
"'But how can we gain your freedom?  You know very well that you
with our agreement?'
"Is there any other point which you would like to ask about?"
with my choice?" I asked.
gets the credit, pray what remains for you?"
[sql1stud@cissys01 ~]$ 
```

Using dots (.) is all well and good, but what happens when I want to look thru a text file to find the lines with periods in them?

The backslash character (\) is the escape character, and it's the regex signal that 'something special is about to happen'.

We can use the escape character as a signal that the following character is to be treated as the character it is, and is not to be treated as one of the magical regex metacharacters.

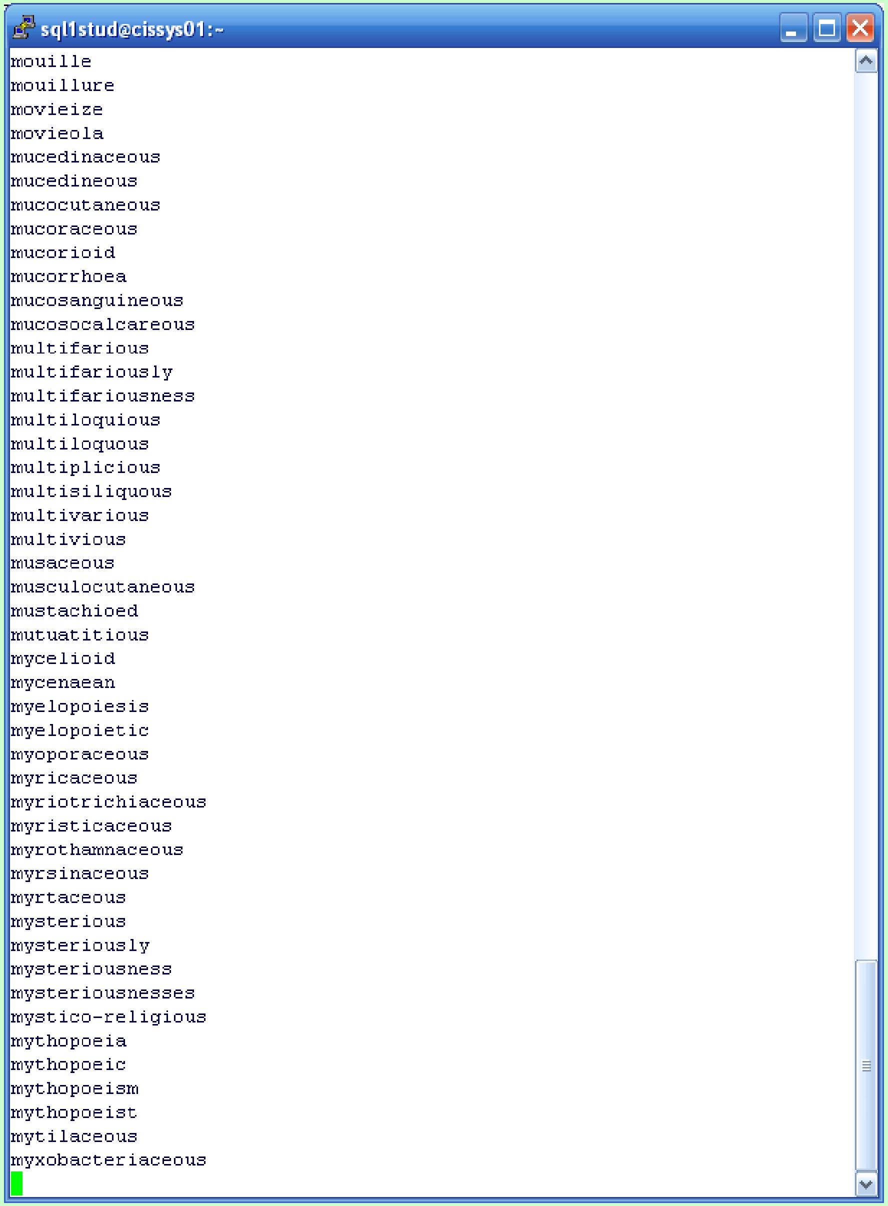
egrep '\.' sign-of-four.txt

will find all lines in this Sherlock Holmes story that have a period (.) in them.

egrep '\?' sign-of-four.txt

will find all lines in this Sherlock Holmes story that have a question-mark (?) in them.

Page D-10: character sets ([])



```
mouille
mouillure
movieize
movieola
mucedinaceous
mucedineous
mucocutaneous
mucoraceous
mucorioid
mucorrhoea
mucosanguineous
mucosocalcareous
multifarious
multifariously
multifariousness
multiloquious
multiloquous
multiplicious
multisiliquous
multivarious
multivious
musaceous
musculocutaneous
mustachioed
mutuatitious
mycelioid
mycenaean
myelopoiesis
myelopoietic
myoporaceous
myricaceous
myriotrichiaceous
myristicaceous
myrothamnaceous
myrsinaceous
myrtaceous
mysterious
mysteriously
mysteriousness
mysteriousnesses
mystico-religious
mythopoeia
mythopoeic
mythopoeism
mythopoeist
mytilaceous
myxobacteriaceous
```

And now we come to one of my favorite features of regex – the ability for us to define our own ‘character sets’.

We just saw how the period (.) can be used to represent ‘any character’. But sometimes we’re looking for a pattern that’s a bit more specific than just any old character.

Using brackets ([]) we can define a regular expression that describes “any of *these* characters”.

egrep ‘[aeiou]’ words

will find any lines that have a vowel in them

egrep ‘[aeiou][aeiou]’ words

will find any lines that have a double vowel sequence in them.

egrep ‘[aeiou][aeiou][aeiou]’ words

will find any lines that have a triple vowel sequence (lower case) in them.

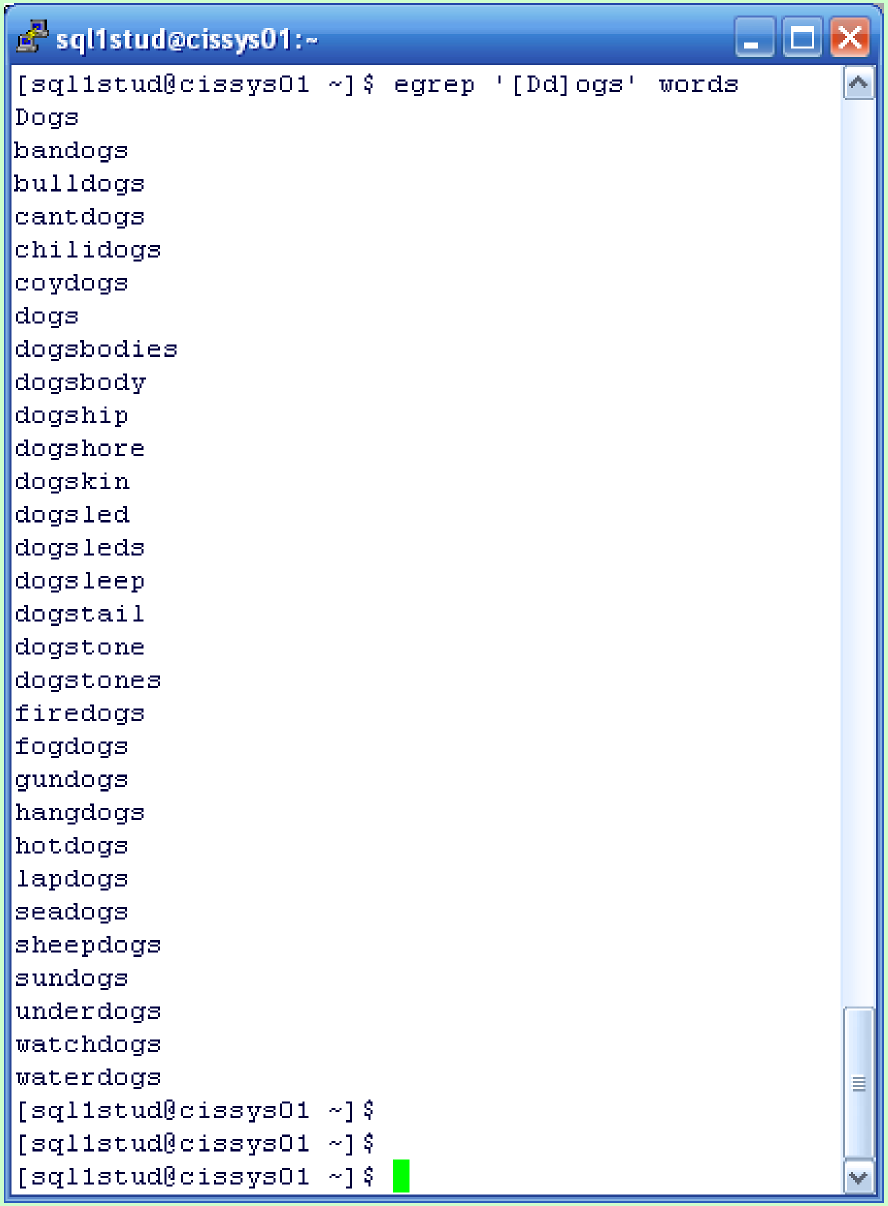
Page D-11: character sets ([])

Are there any words in the dictionary that have four vowels, all strung together?

Are there any words in the dictionary that have five vowels all strung together?

(You’ll see this on one of the projects, so if you’re following along with me, save your work as you figure it out, and you can cut and paste it in later)

Page D-12: character sets ([])



```
[sql1stud@cissys01 ~]$ egrep '[Dd]ogs' words
Dogs
bandogs
bulldogs
cantdogs
chilidogs
coydogs
dogs
dogsbodies
dogsbody
dogship
dogshore
dogskin
dogsled
dogsleds
dogsleep
dogstail
dogstone
dogstones
firedogs
fogdogs
gundogs
hangdogs
hotdogs
lapdogs
seadogs
sheepdogs
sundogs
underdogs
watchdogs
waterdogs
[sql1stud@cissys01 ~]$
[sql1stud@cissys01 ~]$
[sql1stud@cissys01 ~]$
```

```
egrep '[Dd]ogs' words
```

This will locate any lines in the words file that contain the character sequence that starts with an upper-case D or lower-case d, followed immediately by ‘o’ followed immediately by ‘g’ followed immediately by ‘s’.

Page D-13: character sets (ranges)

Regex gives you a shortcut way to define some of these character ranges.

As long as the range of characters all lie together in the character set the system is using, you can use a dash (-) to specify a *range expression* that defines a range/set of characters.

For example:
[0-9] ~ [0123456789]

[A-Z] ~
[ABCDEFGHIJKLMONPQRSTUVWXYZ]

[a-z] ~
[abcdefghijklmnopqrstuvwxyz]

When the dash (-) occurs inside the brackets, it's is treated as a range indicator, when it occurs outside the brackets it's treated simply as the dash (-) character.

Page D-14: character sets (ranges)

How would you find a pattern that matches a typical seven-digit phone number?

```
>
```

Page D-15: character sets (ranges)

How would you find a pattern that matches a typical seven-digit phone number?

This phone number pattern is usually three digits, a dash, and then four digits.

So what would the regex expression look like

>

Page D-16: character sets (ranges)

How would you find a pattern that matches a typical seven-digit phone number?

This phone number pattern is usually three digits, a dash, and then four digits.

So what would the regex expression look like

>

‘[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]’

Page D-17: character sets (ranges)

Remember that the brackets are used to define a character set , that is, the collection of characters can stand in this *single* position.

The consonants in the English language include the whole of the alphabet, except for: a,e,i,o,u (and let’s not consider ‘y’ and ‘w’ – let’s just keep things simple for now).

What regular expression defines the consonant letters of English (only worry about the lower-case versions)?

>

Page D-18: character sets (ranges)

Remember that the brackets are used to define a character set , that is, the collection of characters can stand in this position.

The consonants in the English language include the whole of the alphabet, except for: a,e,i,o,u (and let’s not consider ‘y’ and ‘w’ – let’s just keep things simple for now).

What regular expression defines the consonant letters of English (only worry about the lower-case versions)?

‘[bcdfghjklmnpqrstvwxyz]’

>

Page D-19: character sets (ranges)

Remember that the brackets are used to define a character set , that is, the collection of characters can stand in this position.

The consonants in the English language include the whole of the alphabet, except for: a,e,i,o,u (and let’s not consider ‘y’ and ‘w’ – let’s just keep things simple for now).

What regular expression defines the consonant letters of English (only worry about the lower-case versions)?

‘[bcdfghjklmnpqrstvwxyz]’

Or

[b-df-hj-np-tv-z]

Or

>

Page D-20: character sets (ranges)

Remember that the brackets are used to define a character set , that is, the collection of characters can stand in this position.

The consonants in the English language include the whole of the alphabet, except for: a,e,i,o,u (and let's not consider 'y' and 'w' – let's just keep things simple for now).

What regular expression defines the consonant letters of English (only worry about the lower-case versions)?

'[bcdfghjklmnpqrstvwxyz]

Or

[b-df-hj-np-tv-z]

Or

[bcdfghj-np-tvwxyz]

Page D-21: character sets (ranges)

Regex permits us to define character sets in a very flexible and easy to use fashion.

As a programmer all you have to do is list each of the characters that may occur in this ‘slot’ in the target character string.

If it helps to use ranges, then go ahead and use ranges. And as I demonstrated on the previous slide, each of these expressions is equivalent to one another.

```
[bcdfghjklmnpqrstvwxyz]
[b-df-hj-np-tv-z]
[bcdfghj-np-tvwxyz]
```

What words in the dictionary contain a string of six consonants? Are there any words in the dictionary that have an eight-character consonant cluster?

Page D-22: character classes

Some implementations of regex provide yet another short cut when referring to character sets. Character *classes* have been predefined for some of the more popular character sets, and we can use these character class names in lieu of having to specify each of the characters in the set.

For example:
[0123456789] ~ [0-9] ~ [[:digit:]]
- - -

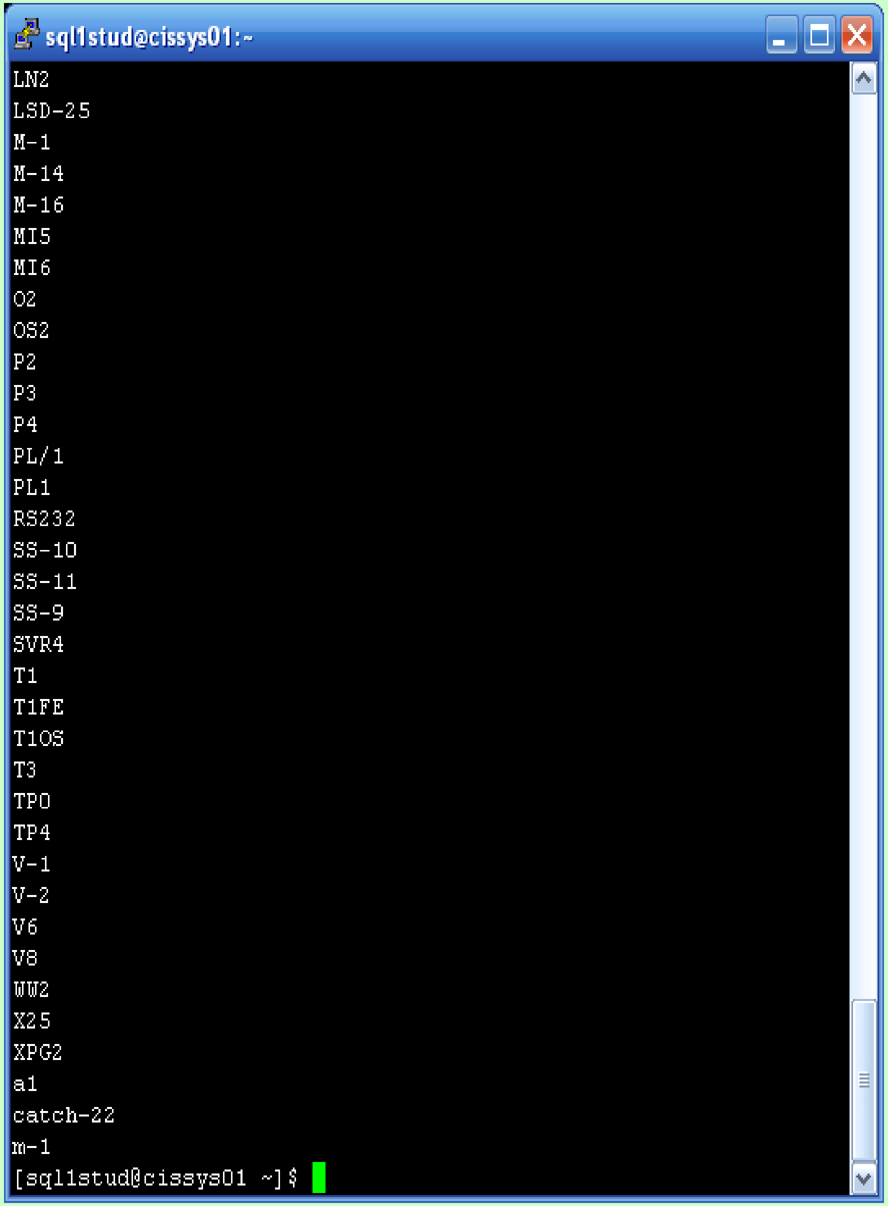
[[:alpha:]]	any letter
[[:lower:]]	any lower-case letter
[[:upper:]]	any upper-case letter
[[:digit:]]	any digit
[[:xdigit:]]	any hexadecimal digit ie. [0-9][a-f][A-F]
[[:alnum:]]	any letter or digit
[[:punct:]]	punctuation-like characters
[[:space:]]	any whitespace character

Page D-23: character classes

Using character classes, let’s check the dictionary for any ‘words’ that have digits in them:

```
>
```

Page D-24: character classes

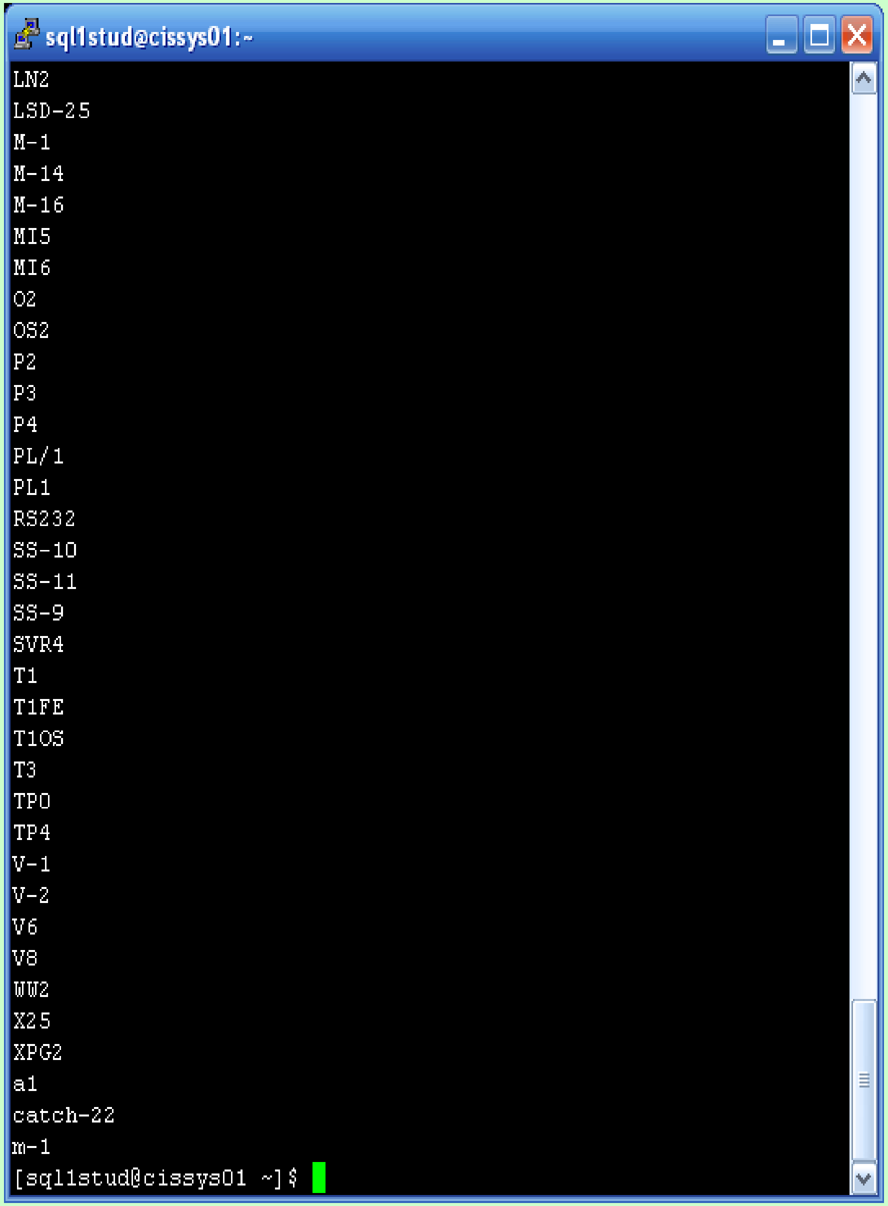


```
LN2
LSD-25
M-1
M-14
M-16
MI5
MI6
O2
OS2
P2
P3
P4
PL/1
PL1
RS232
SS-10
SS-11
SS-9
SVR4
T1
T1FE
T1OS
T3
TP0
TP4
V-1
V-2
V6
V8
WW2
X25
XPG2
a1
catch-22
m-1
[sql1stud@cissys01 ~]$
```

Using character classes, let’s check the dictionary for any ‘words’ that have digits in them:

egrep ‘[[:digit:]]’ words

Page D-25: character classes



```
LN2
LSD-25
M-1
M-14
M-16
MI5
MI6
O2
OS2
P2
P3
P4
PL/1
PL1
RS232
SS-10
SS-11
SS-9
SVR4
T1
T1FE
T1OS
T3
TP0
TP4
V-1
V-2
V6
V8
WW2
X25
XPG2
a1
catch-22
m-1
[sql1stud@cissys01 ~]$
```

Using character classes, let’s check the dictionary for any ‘words’ that have digits in them:

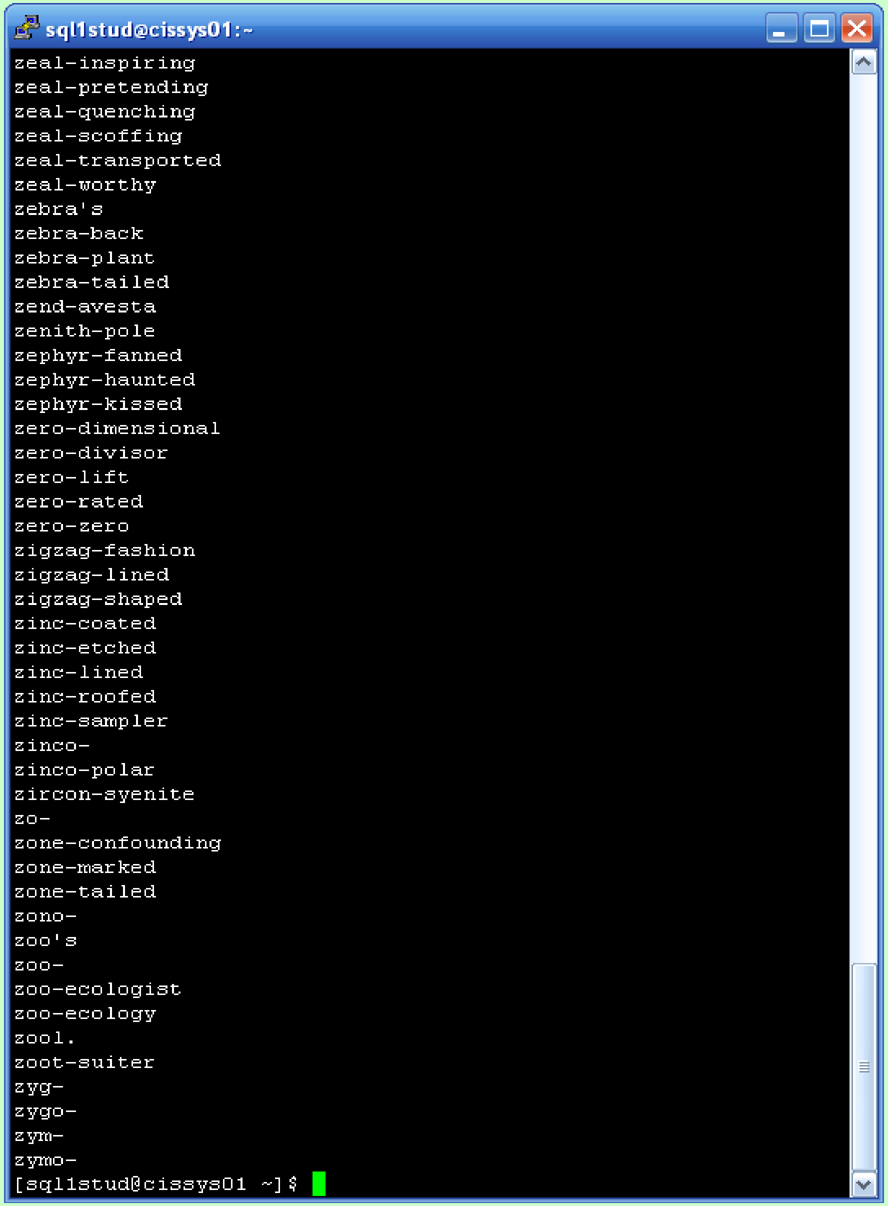
egrep ‘[[:digit:]]’ words

Now let’s look for any lines in the dictionary that include punctuation-like marks

>

Module 06: Character Data

Page D-26: character classes



```
zeal-inspiring
zeal-pretending
zeal-quenching
zeal-scoffing
zeal-transported
zeal-worthy
zebra's
zebra-back
zebra-plant
zebra-tailed
zend-avesta
zenith-pole
zephyr-fanned
zephyr-haunted
zephyr-kissed
zero-dimensional
zero-divisor
zero-lift
zero-rated
zero-zero
zigzag-fashion
zigzag-lined
zigzag-shaped
zinc-coated
zinc-etched
zinc-lined
zinc-roofed
zinc-sampler
zinco-
zinco-polar
zircon-syenite
zo-
zone-confounding
zone-marked
zone-tailed
zono-
zoo's
zoo-
zoo-ecologist
zoo-ecology
zool.
zoot-suiter
zyg-
zygo-
zym-
zymo-
[sql1stud@cissys01 ~]$
```

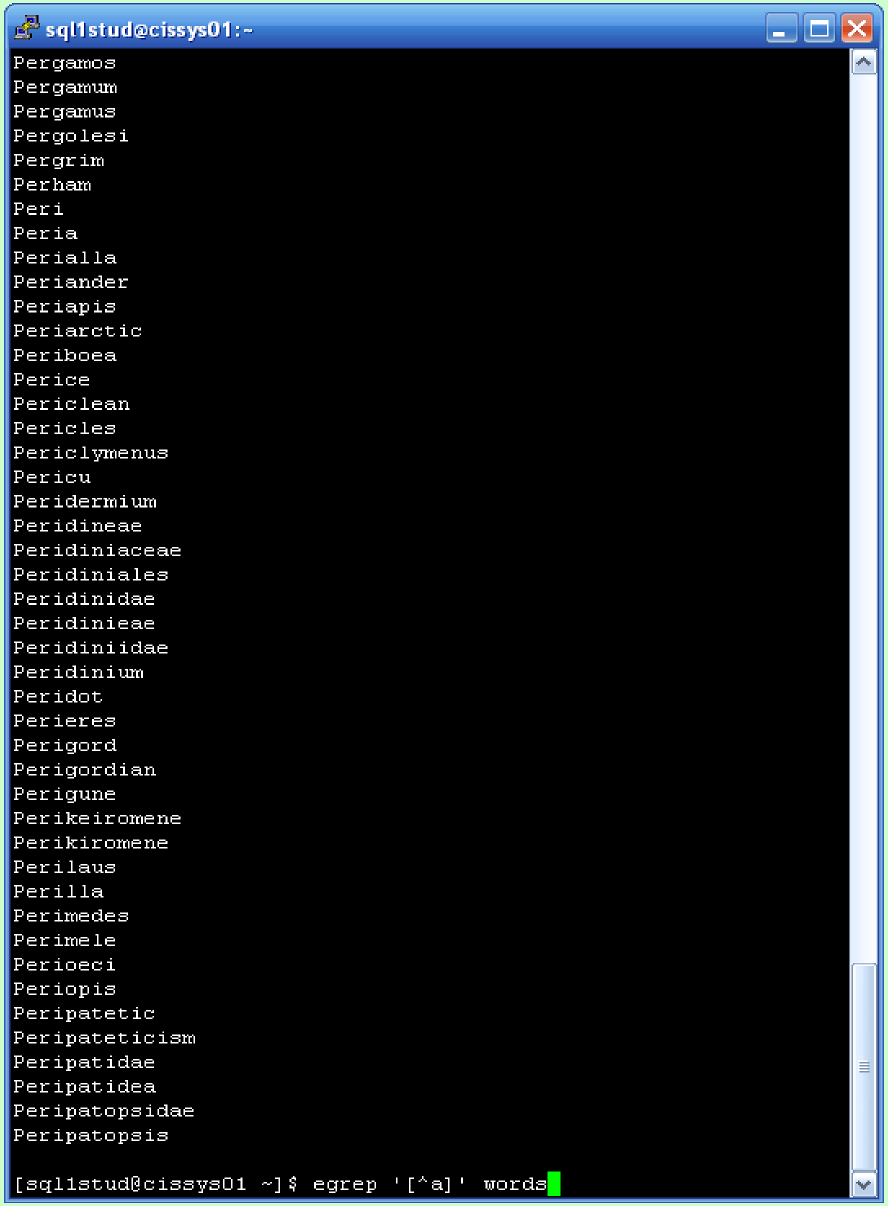
Using character classes, let’s check the dictionary for any ‘words’ that have digits in them:

egrep ‘[[:digit:]]’ words

Now let’s look for any lines in the dictionary that include punctuation-like marks

egrep ‘[[:punct:]]’ words

Page D-27: Boolean - NOT



```
Pergamos
Pergamum
Pergamus
Pergolesi
Pergrim
Perham
Peri
Peria
Perialla
Periander
Periapis
Periarctic
Periboea
Perice
Periclean
Pericles
Periclymenus
Pericu
Peridermium
Peridineae
Peridiniaceae
Peridiniales
Peridinidae
Peridinieae
Peridiniidae
Peridinium
Peridot
Perieres
Perigord
Perigordian
Perigune
Perikeiromene
Perikiromene
Perilaus
Perilla
Perimedes
Perimele
Perioeci
Periopis
Peripatetic
Peripateticism
Peripatidae
Peripatidea
Peripatopsidae
Peripatopsis

[sql1stud@cissys01 ~]$ egrep '[^a]' words
```

As we saw in one of our earlier modules, sometimes when we’re constructing a comparison expression, it might just be easier to describe what we’re not looking for as opposed to what we are looking for.

Regular expressions provide us with this same degree of flexibility, and we can use the not operator (^) to accomplish this ‘negation’.

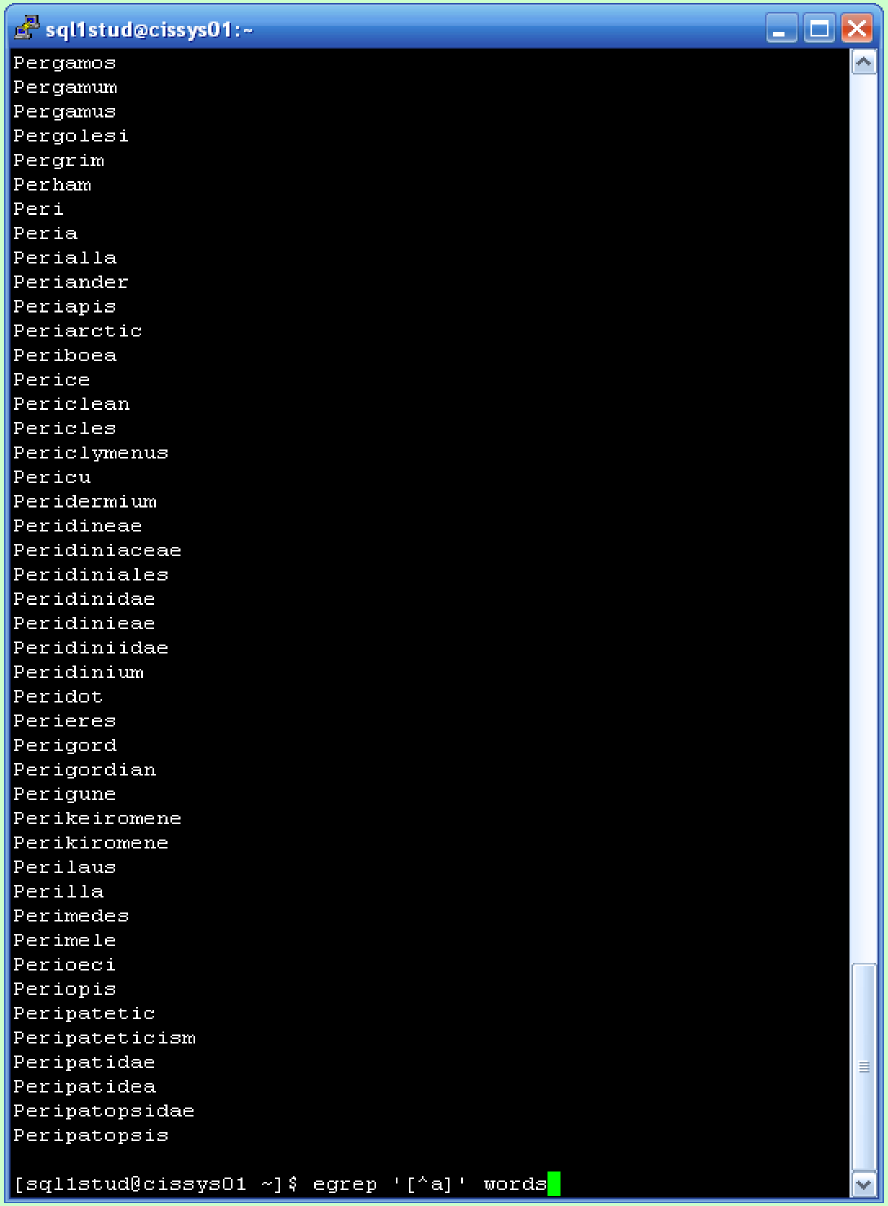
For example, let’s look for all of the words in the dictionary that include some character other than the letter ‘a’.

egrep ‘ [^a]’ words

But wait a sec! This doesn’t look quite ‘kosher’. That first line in the screen shot has the letter ‘a’ in it.

What’s up with that?

Page D-28: Boolean - NOT



```
Pergamos
Pergamum
Pergamus
Pergolesi
Pergrim
Perham
Peri
Peria
Perialla
Periander
Periapis
Periarctic
Periboea
Perice
Periclean
Pericles
Periclymenus
Pericu
Peridermium
Peridineae
Peridiniaceae
Peridiniales
Peridinidae
Peridinieae
Peridiniidae
Peridinium
Peridot
Perieres
Perigord
Perigordian
Perigune
Perikeiromene
Perikiromene
Perilaus
Perilla
Perimedes
Perimele
Perioeci
Periopis
Peripatetic
Peripateticism
Peripatidae
Peripatidea
Peripatopsidae
Peripatopsis

[sql1stud@cissys01 ~]$ egrep '[^a]' words
```

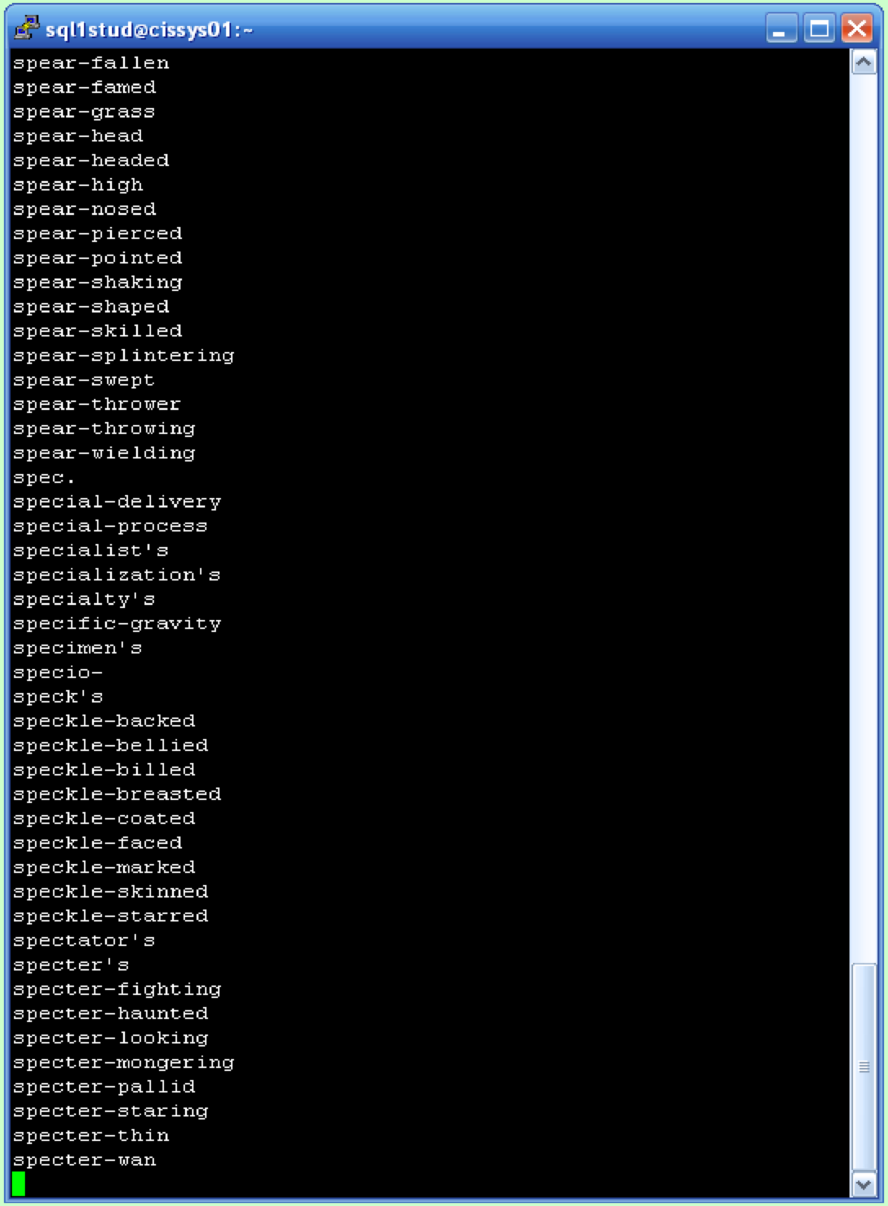
Egrep checks thru each line of the file, and prints out all lines that match the regular expression.

Take a moment to recall how egrep evaluates a regular expression.

It checks each line on a character-by-character basis, and as soon as it finds a character sequence that matches the pattern, it prints it out.

In that first line of output, egrep discovered a match as soon as it encountered any character that wasn't an ‘a’. ‘P’ is not ‘a’, hence that line was printed out.

Page D-29: Boolean - NOT



```
spear-fallen
spear-famed
spear-grass
spear-head
spear-headed
spear-high
spear-nosed
spear-pierced
spear-pointed
spear-shaking
spear-shaped
spear-skilled
spear-splintering
spear-swept
spear-thrower
spear-throwing
spear-wielding
spec.
special-delivery
special-process
specialist's
specialization's
specialty's
specific-gravity
specimen's
specio-
speck's
speckle-backed
speckle-bellied
speckle-billed
speckle-breasted
speckle-coated
speckle-faced
speckle-marked
speckle-skinned
speckle-starred
spectator's
specter's
specter-fighting
specter-haunted
specter-looking
specter-mongering
specter-pallid
specter-staring
specter-thin
specter-wan
```

Note also, that the not operator (^) was included inside the brackets. This is important, but we haven't covered enough ground yet for me to explain why – give me a few seconds to get there.

But let's try another example using the regular expression not operator, but this time with character classes.

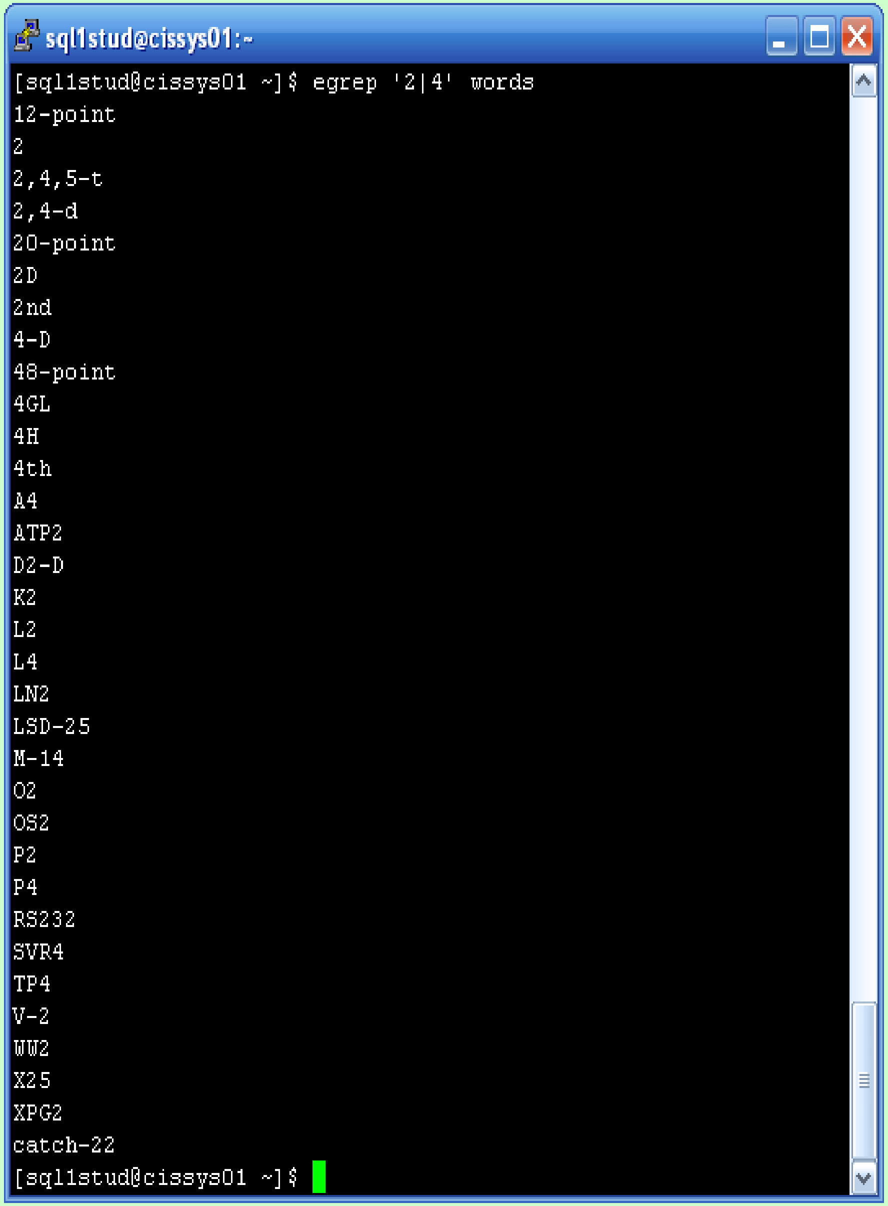
Let's look for all of the words in the dictionary that don't have a letter in them.

egrep ' [^[:letter:]]' words

Notice that this expression doesn't locate words that don't have 'any letters', so much as words that include something 'other than a letter'...

- - -

Page D-30: Boolean - OR



```
[sql1stud@cissys01 ~]$ egrep '2|4' words
12-point
2
2,4,5-t
2,4-d
20-point
2D
2nd
4-D
48-point
4GL
4H
4th
A4
ATP2
D2-D
K2
L2
L4
LN2
LSD-25
M-14
O2
OS2
P2
P4
RS232
SVR4
TP4
V-2
WW2
X25
XPG2
catch-22
[sql1stud@cissys01 ~]$
```

The next Boolean operator that we should examine is the ‘or’ operator (|).

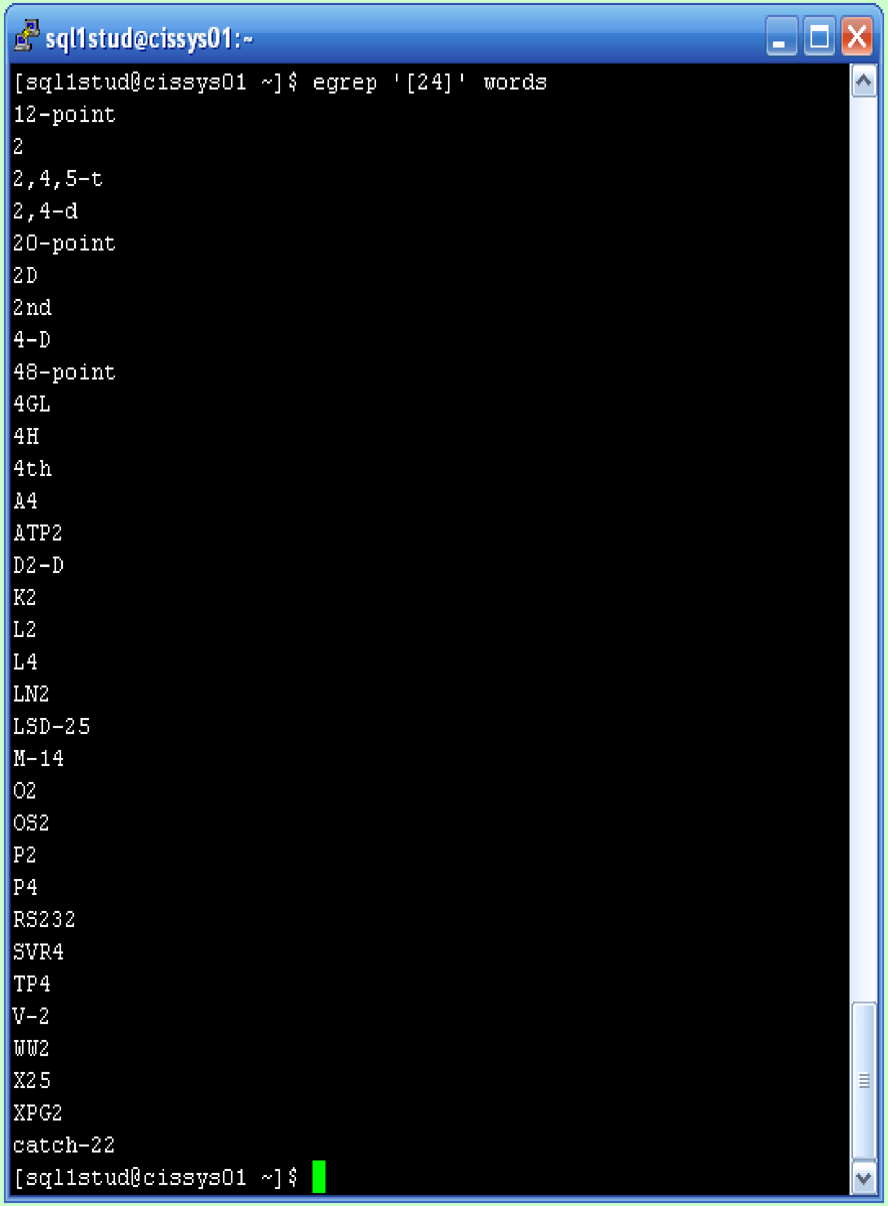
Let’s look for all of the words in the dictionary that include either the digit 2, or the digit 4.

egrep ‘2|4’ words

Pretty straightforward?

>

Page D-31: Boolean - OR



```
[sql1stud@cissys01 ~]$ egrep '[24]' words
12-point
2
2,4,5-t
2,4-d
20-point
2D
2nd
4-D
48-point
4GL
4H
4th
A4
ATP2
D2-D
K2
L2
L4
LN2
LSD-25
M-14
O2
OS2
P2
P4
RS232
SVR4
TP4
V-2
WW2
X25
XPG2
catch-22
[sql1stud@cissys01 ~]$
```

The next Boolean operator that we should examine is the ‘or’ operator (|).

Let’s look for all of the words in the dictionary that include either the digit 2, or the digit 4.

egrep ‘2|4’ words

Pretty straightforward?

We could have defined a character set [24], and gotten the same results:

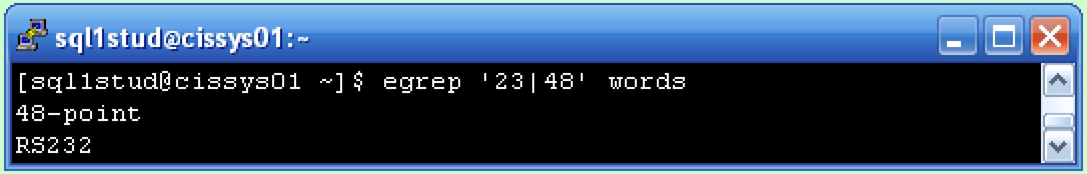
egrep ‘[24]’ words

Page D-32: Boolean - OR

But how about this now.

Let’s say we want to find either 23 or 48. How might we do that…

egrep ’23|48’ words



```
[sql1stud@cissys01 ~]$ egrep '23|48' words
48-point
RS232
```

And if we were looking for either 22, 23, or 48 – how might we do that?

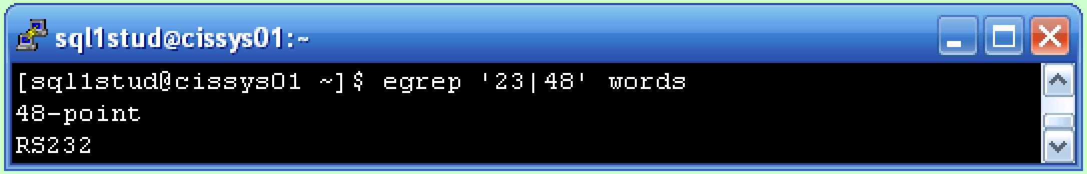
>

Page D-33: Boolean - OR

But how about this now.

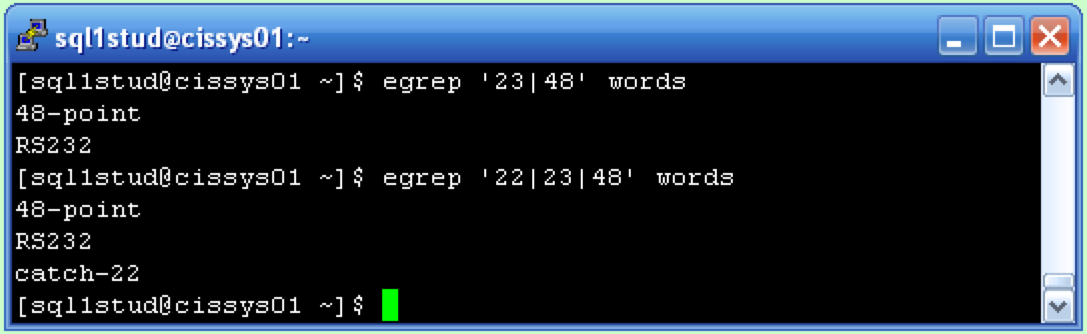
Let’s say we want to find either 23 or 48. How might we do that…

egrep ’23|48’ words



```
[sql1stud@cissys01 ~]$ egrep '23|48' words
48-point
RS232
```

And if we were looking for either 22, 23, or 48 – how might we do that?



```
[sql1stud@cissys01 ~]$ egrep '23|48' words
48-point
RS232
[sql1stud@cissys01 ~]$ egrep '22|23|48' words
48-point
RS232
catch-22
[sql1stud@cissys01 ~]$
```

Page D-34: Boolean - OR

This brings up the notion of ‘scope’, that is what is the scope, or range of these operators?

The ‘or’ mark (|) splits the expression up into chunks, and *everything* on the left hand side of the bar (|) is compared with *everything* on the right hand side of the bar.

Page D-35: Parens

We can use parentheses to delimit the scope of these regular expression operators.

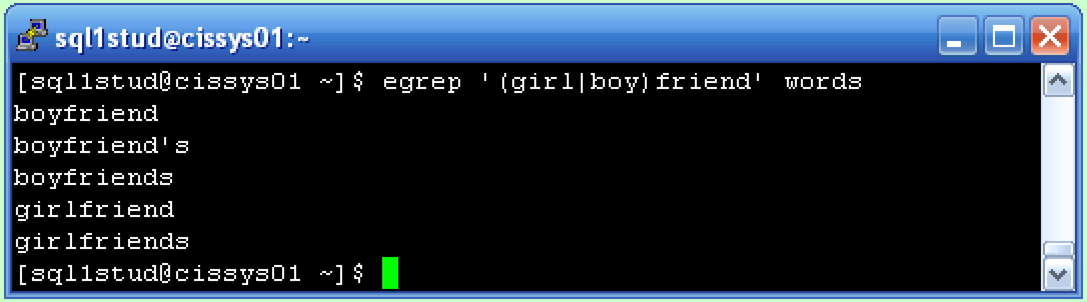
Find all occurrences of ‘girlfriend’ or ‘boyfriend’ in the file.

One solution would be:

egrep ‘girlfriend|boyfriend’ words

Another would be:

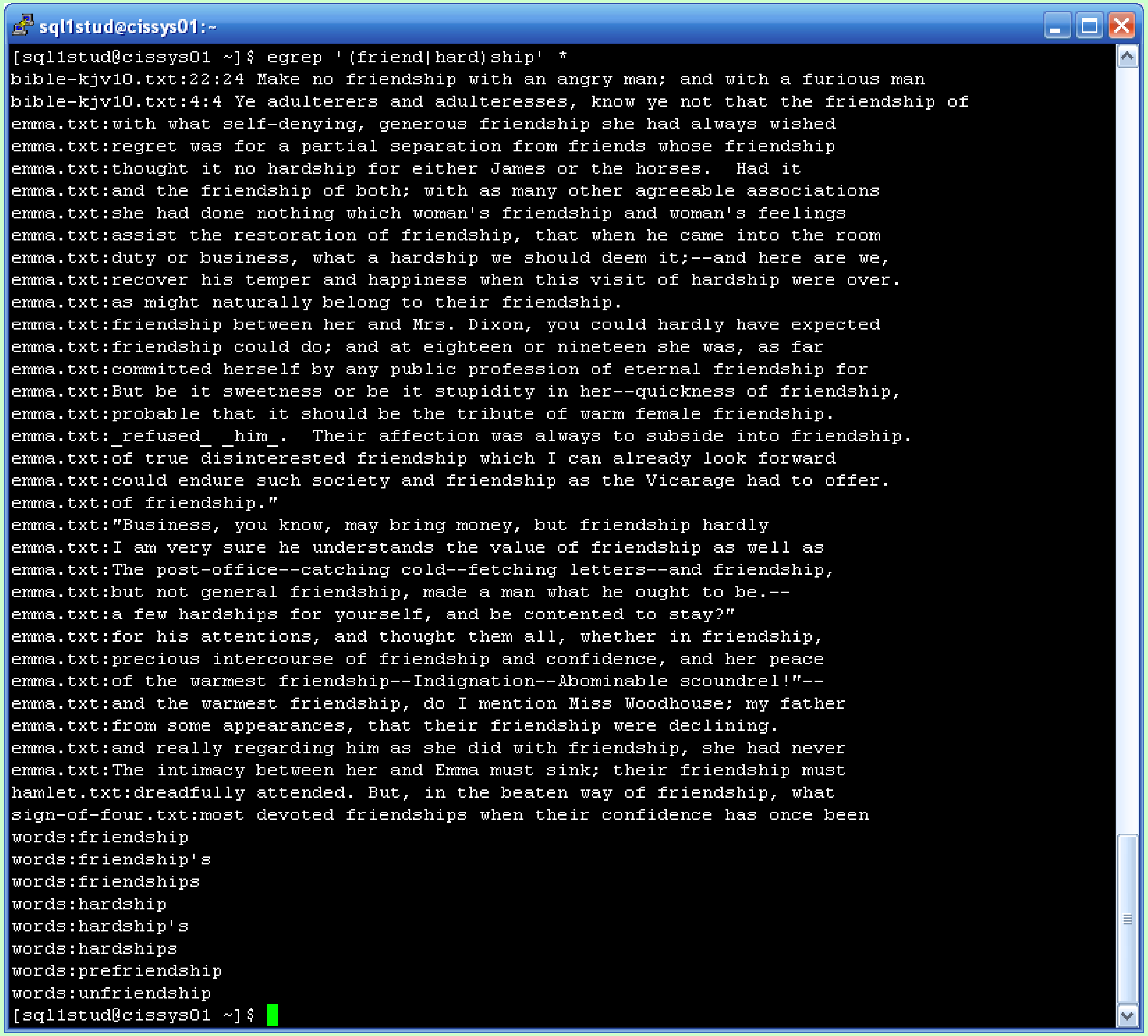
egrep ‘(girl|boy)friend’ words



```
[sql1stud@cissys01 ~]$ egrep '(girl|boy)friend' words
boyfriend
boyfriend's
boyfriends
girlfriend
girlfriends
[sql1stud@cissys01 ~]$
```

Page D-36: Parens

Let’s try looking for either of the strings: friendship or hardship in any of our files.



```
[sql1stud@cissys01 ~]$ egrep '(friend|hard)ship' *
bible-kjv10.txt:22:24 Make no friendship with an angry man; and with a furious man
bible-kjv10.txt:4:4 Ye adulterers and adulteresses, know ye not that the friendship of
emma.txt:with what self-denying, generous friendship she had always wished
emma.txt:regret was for a partial separation from friends whose friendship
emma.txt:thought it no hardship for either James or the horses.  Had it
emma.txt:and the friendship of both; with as many other agreeable associations
emma.txt:she had done nothing which woman's friendship and woman's feelings
emma.txt:assist the restoration of friendship, that when he came into the room
emma.txt:duty or business, what a hardship we should deem it;--and here are we,
emma.txt:recover his temper and happiness when this visit of hardship were over.
emma.txt:as might naturally belong to their friendship.
emma.txt:friendship between her and Mrs. Dixon, you could hardly have expected
emma.txt:friendship could do; and at eighteen or nineteen she was, as far
emma.txt:committed herself by any public profession of eternal friendship for
emma.txt:But be it sweetness or be it stupidity in her--quickness of friendship,
emma.txt:probable that it should be the tribute of warm female friendship.
emma.txt:_refused_ _him_.  Their affection was always to subside into friendship.
emma.txt:of true disinterested friendship which I can already look forward
emma.txt:could endure such society and friendship as the Vicarage had to offer.
emma.txt:of friendship."
emma.txt:"Business, you know, may bring money, but friendship hardly
emma.txt:I am very sure he understands the value of friendship as well as
emma.txt:The post-office--catching cold--fetching letters--and friendship,
emma.txt:but not general friendship, made a man what he ought to be.--
emma.txt:a few hardships for yourself, and be contented to stay?"
emma.txt:for his attentions, and thought them all, whether in friendship,
emma.txt:precious intercourse of friendship and confidence, and her peace
emma.txt:of the warmest friendship--Indignation--Abominable scoundrel!"--
emma.txt:and the warmest friendship, do I mention Miss Woodhouse; my father
emma.txt:from some appearances, that their friendship were declining.
emma.txt:and really regarding him as she did with friendship, she had never
emma.txt:The intimacy between her and Emma must sink; their friendship must
hamlet.txt:dreadfully attended. But, in the beaten way of friendship, what
sign-of-four.txt:most devoted friendships when their confidence has once been
words:friendship
words:friendship's
words:friendships
words:hardship
words:hardship's
words:hardships
words:prefriendship
words:unfriendship
[sql1stud@cissys01 ~]$
```

Page D-37: boundaries

Carrying on, the next concept that I want to explore is the notion of *boundaries*.

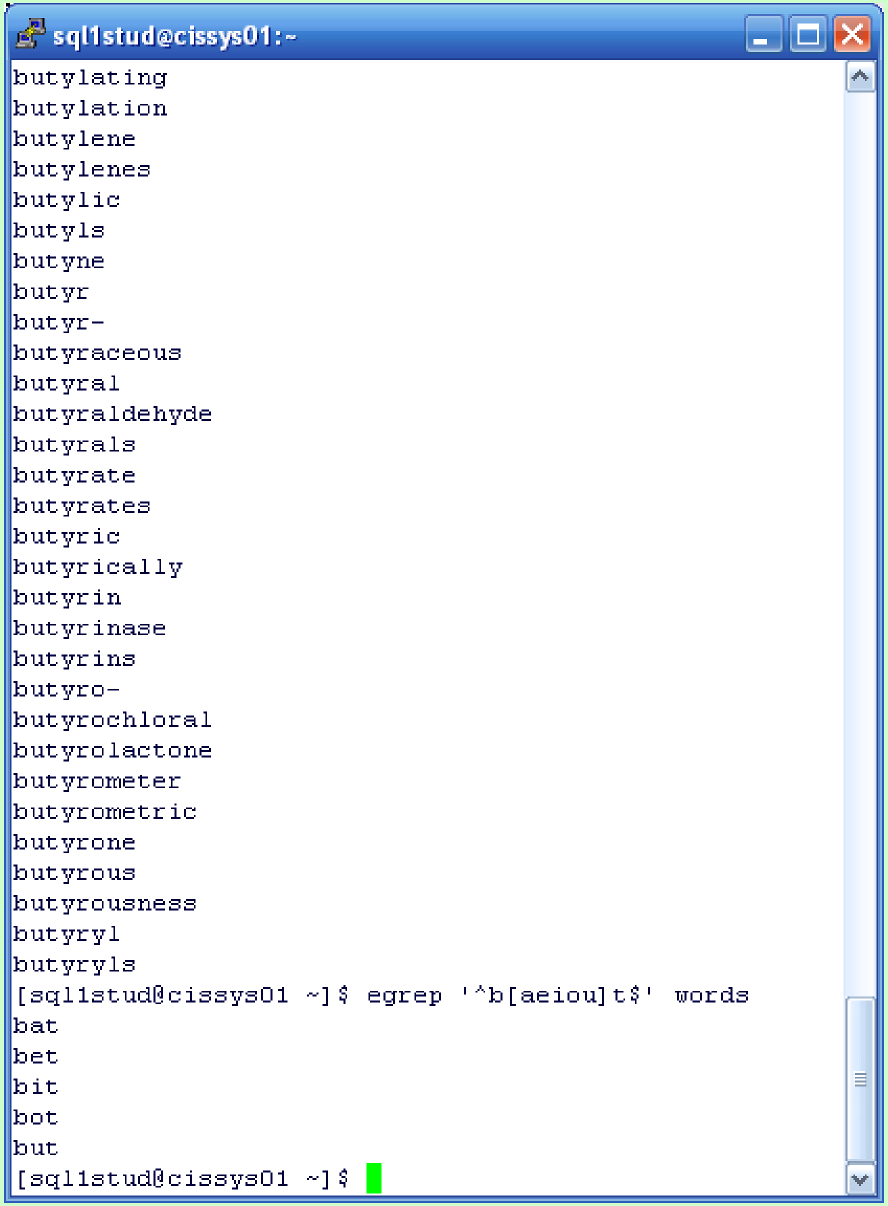
Every line in a file has a beginning and an end, as do the words in each of those lines.

There are occasions when we're interested in knowing what's happening at the beginning of the line, or the end of the line , or ….

And we can use these boundaries as delimiters in our regular expressions. Here are some metacharacters that will come in handy:

^ - start of the line
$ - end of the line

Page D-38: boundaries



```
butylating
butylation
butylene
butylenes
butylic
butyls
butyne
butyr
butyr-
butyraceous
butyral
butyraldehyde
butyrals
butyrate
butyrates
butyric
butyrically
butyrin
butyrinase
butyrins
butyro-
butyrochloral
butyrolactone
butyrometer
butyrometric
butyrone
butyrous
butyrousness
butyryl
butyryls
[sql1stud@cissys01 ~]$ egrep '^b[aeiou]t$' words
bat
bet
bit
bot
but
[sql1stud@cissys01 ~]$
```

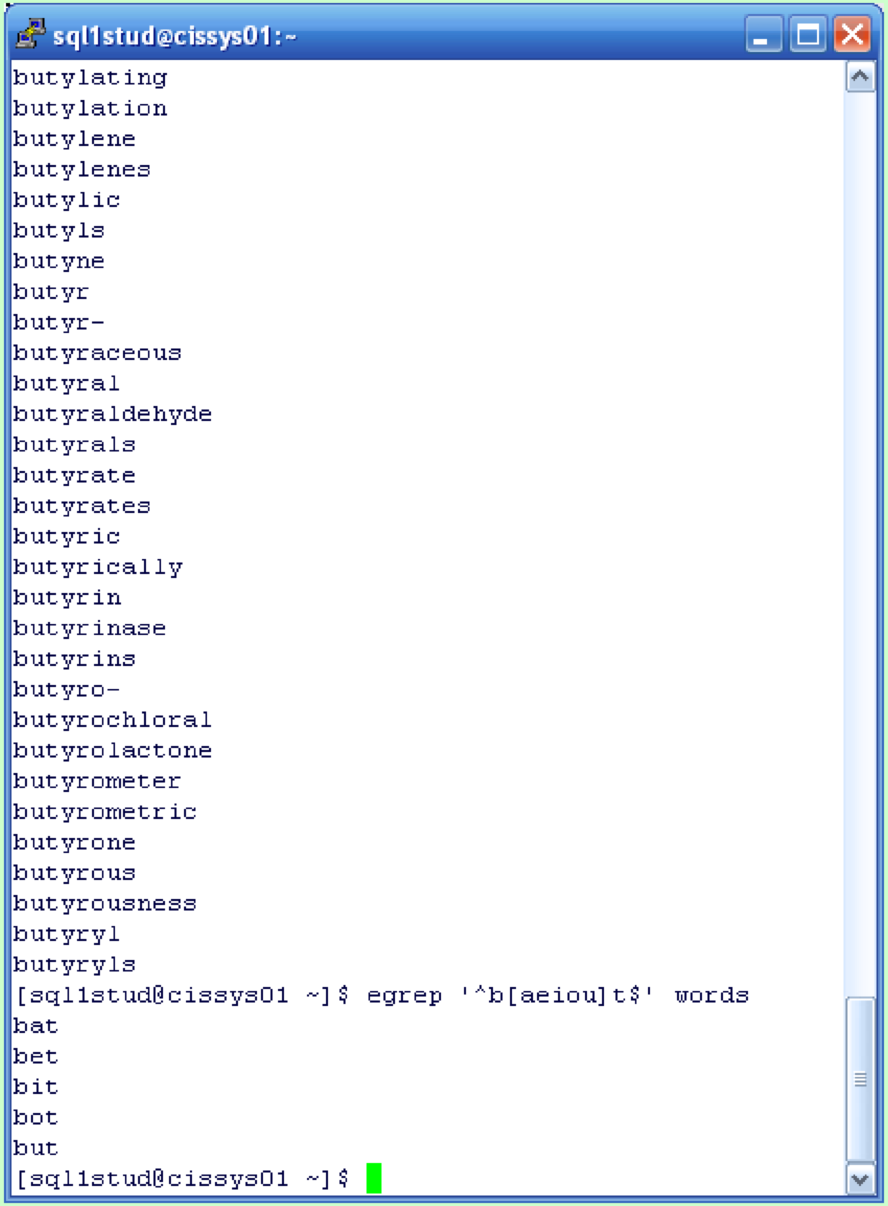
egrep ‘^b[aeiou]t’ words

Will locate all of the words in the dictionary that start with ‘b’, are followed immediately by an English vowel, which is followed immediately by a ‘t’.

egrep ‘^b[aeiou]t$’ words

Will locate all of the words in the dictionary that start with ‘b’, are followed immediately by an English vowel, which is followed immediately by a ‘t’, which is followed immediately by the end of the line.

Page D-39: boundaries



```
butylating
butylation
butylene
butylenes
butylic
butyls
butyne
butyr
butyr-
butyraceous
butyral
butyraldehyde
butyrals
butyrate
butyrates
butyric
butyrically
butyrin
butyrinase
butyrins
butyro-
butyrochloral
butyrolactone
butyrometer
butyrometric
butyrone
butyrous
butyrousness
butyryl
butyryls
[sql1stud@cissys01 ~]$ egrep '^b[aeiou]t$' words
bat
bet
bit
bot
but
[sql1stud@cissys01 ~]$
```

The delimiters for word boundaries vary from product to product, and in this version of egrep, (\<) marks the beginning of a word, and (\>) marks the ending of a word.

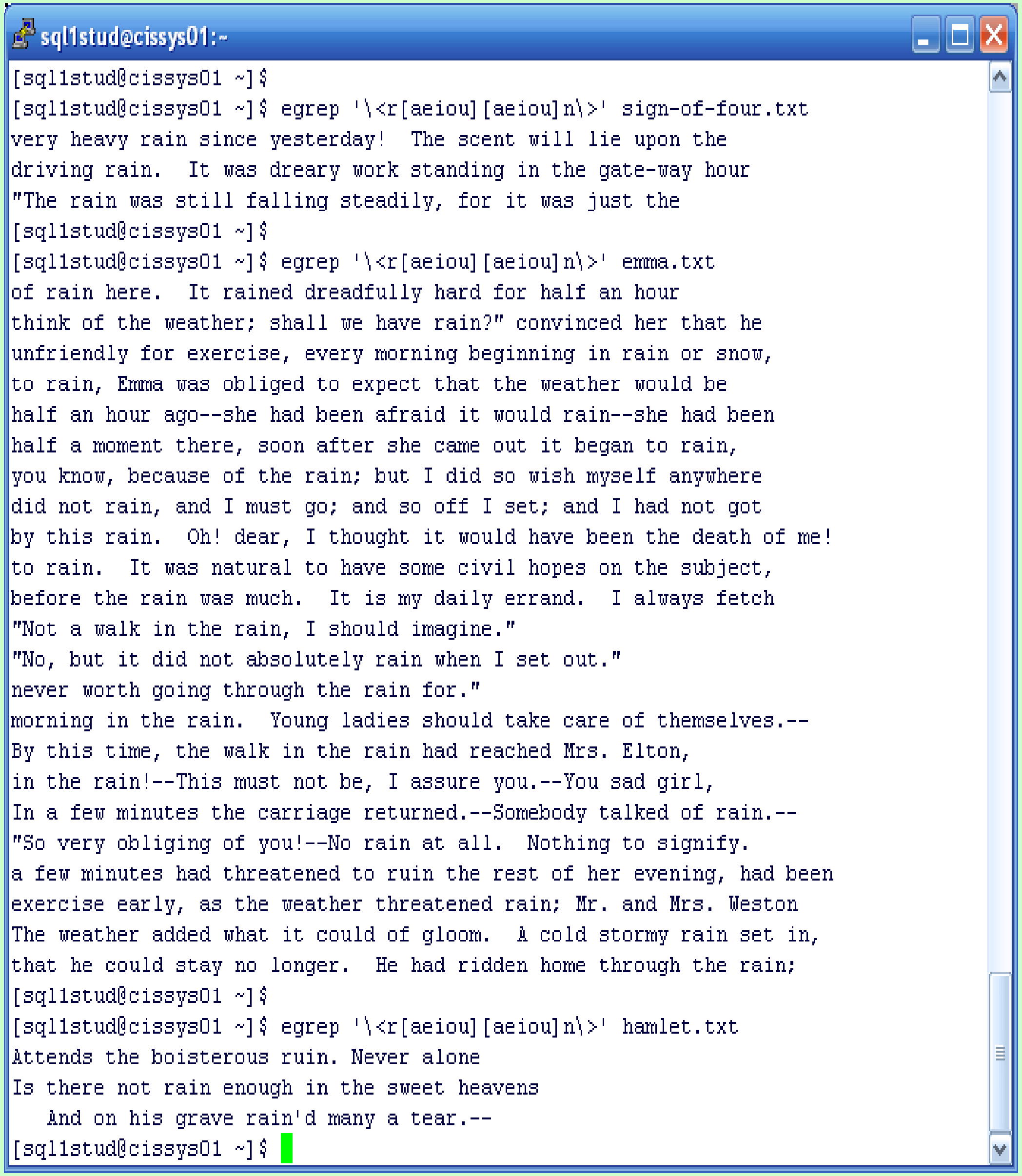
egrep '^b[aeiou]t' words

This egrep invocation will locate all of the words in the dictionary that start with 'b', followed immediately by an English vowel, followed immediately by a 't'.

egrep '^b[aeiou]t$' words

This one will locate all of the words in the dictionary that start with 'b', followed immediately by an English vowel, followed immediately by a 't', followed immediately by the end of the line.

Module 06: Character Data Page D-40: boundaries



```
[sql1stud@cissys01 ~]$
[sql1stud@cissys01 ~]$ egrep '\<r[aeiou][aeiou]n\>' sign-of-four.txt
very heavy rain since yesterday!  The scent will lie upon the
driving rain.  It was dreary work standing in the gate-way hour
"The rain was still falling steadily, for it was just the
[sql1stud@cissys01 ~]$
[sql1stud@cissys01 ~]$ egrep '\<r[aeiou][aeiou]n\>' emma.txt
of rain here.  It rained dreadfully hard for half an hour
think of the weather; shall we have rain?" convinced her that he
unfriendly for exercise, every morning beginning in rain or snow,
to rain, Emma was obliged to expect that the weather would be
half an hour ago--she had been afraid it would rain--she had been
half a moment there, soon after she came out it began to rain,
you know, because of the rain; but I did so wish myself anywhere
did not rain, and I must go; and so off I set; and I had not got
by this rain.  Oh! dear, I thought it would have been the death of me!
to rain.  It was natural to have some civil hopes on the subject,
before the rain was much.  It is my daily errand.  I always fetch
"Not a walk in the rain, I should imagine."
"No, but it did not absolutely rain when I set out."
never worth going through the rain for."
morning in the rain.  Young ladies should take care of themselves.--
By this time, the walk in the rain had reached Mrs. Elton,
in the rain!--This must not be, I assure you.--You sad girl,
In a few minutes the carriage returned.--Somebody talked of rain.--
"So very obliging of you!--No rain at all.  Nothing to signify.
a few minutes had threatened to ruin the rest of her evening, had been
exercise early, as the weather threatened rain; Mr. and Mrs. Weston
The weather added what it could of gloom.  A cold stormy rain set in,
that he could stay no longer.  He had ridden home through the rain;
[sql1stud@cissys01 ~]$
[sql1stud@cissys01 ~]$ egrep '\<r[aeiou][aeiou]n\>' hamlet.txt
Attends the boisterous ruin. Never alone
Is there not rain enough in the sweet heavens
   And on his grave rain'd many a tear.--
[sql1stud@cissys01 ~]$
```

Can you see how the word boundary anchors are ‘smartly’ processed by regex?

Regex is not looking simply for white space to define word boundaries, it also ‘knows’ about punctuation marks.

Take a moment to examine this sample. The target string, ie the search pattern is all four-character words that start with ‘r’, end with ‘n’, and have two vowels in between.

Page D-41: repeating characters

Another one of the more important features of regex is the easy way it allows us to specify ‘repeating characters’.

C*	Matches zero or more occurrences of this character
C+	Matches one or more occurrences of this character
C?	Matches zero or one occurrences of this character
C{m,n}	Matches at least m, but no more than n, occurrences of this character

Page D-42: repeating characters

C*	Matches zero or more occurrences of this character
C+	Matches one or more occurrences of this character
C?	Matches zero or one occurrences of this character
C{m,n}	Matches at least m, but no more than n, occurrences of this character

To find all words in the dictionary that have a two or more ‘o’s in sequence, (oo…), we could try this regular expression:

```
egrep ‘oo+’ sign-of-four.txt
```

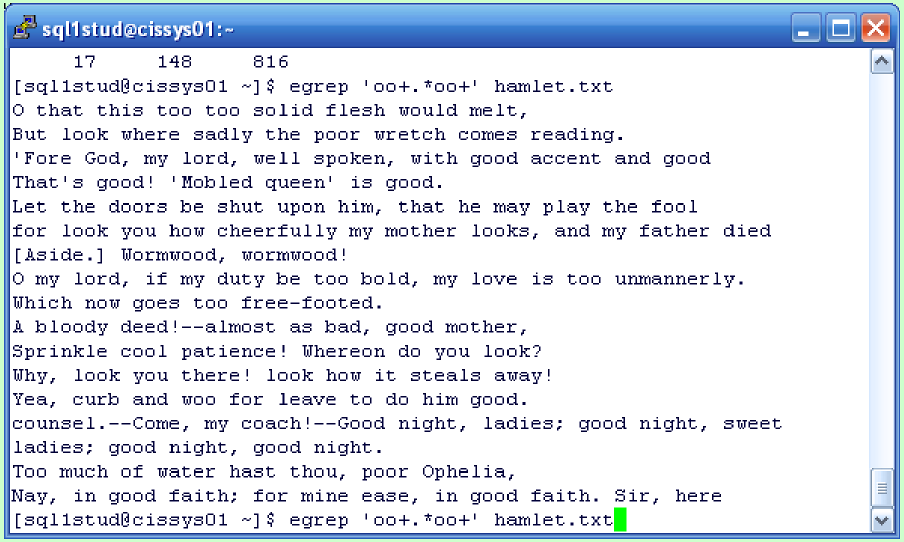
or

```
egrep ‘ooo*’ sign-of-four.txt
```

Module 06: Character Data

Page D-43: repeating characters

C*	Matches zero or more occurrences of this character
C+	Matches one or more occurrences of this character
C?	Matches zero or one occurrences of this character
C{m,n}	Matches at least m, but no more than n, occurrences of this character



```
     17      148      816
[sql1stud@cissys01 ~]$ egrep 'oo+.*oo+' hamlet.txt
O that this too too solid flesh would melt,
But look where sadly the poor wretch comes reading.
'Fore God, my lord, well spoken, with good accent and good
That's good! 'Mobled queen' is good.
Let the doors be shut upon him, that he may play the fool
for look you how cheerfully my mother looks, and my father died
[Aside.] Wormwood, wormwood!
O my lord, if my duty be too bold, my love is too unmannerly.
Which now goes too free-footed.
A bloody deed!--almost as bad, good mother,
Sprinkle cool patience! Whereon do you look?
Why, look you there! look how it steals away!
Yea, curb and woo for leave to do him good.
counsel.--Come, my coach!--Good night, ladies; good night, sweet
ladies; good night, good night.
Too much of water hast thou, poor Ophelia,
Nay, in good faith; for mine ease, in good faith. Sir, here
[sql1stud@cissys01 ~]$ egrep 'oo+.*oo+' hamlet.txt
```

To find all words in the story that have a two or more ‘o’s in sequence, (oo…), we could try this regular expression:

```
egrep ‘oo+’ sign-of-four.txt
```

or

```
egrep ‘ooo*’ sign-of-four.txt
```

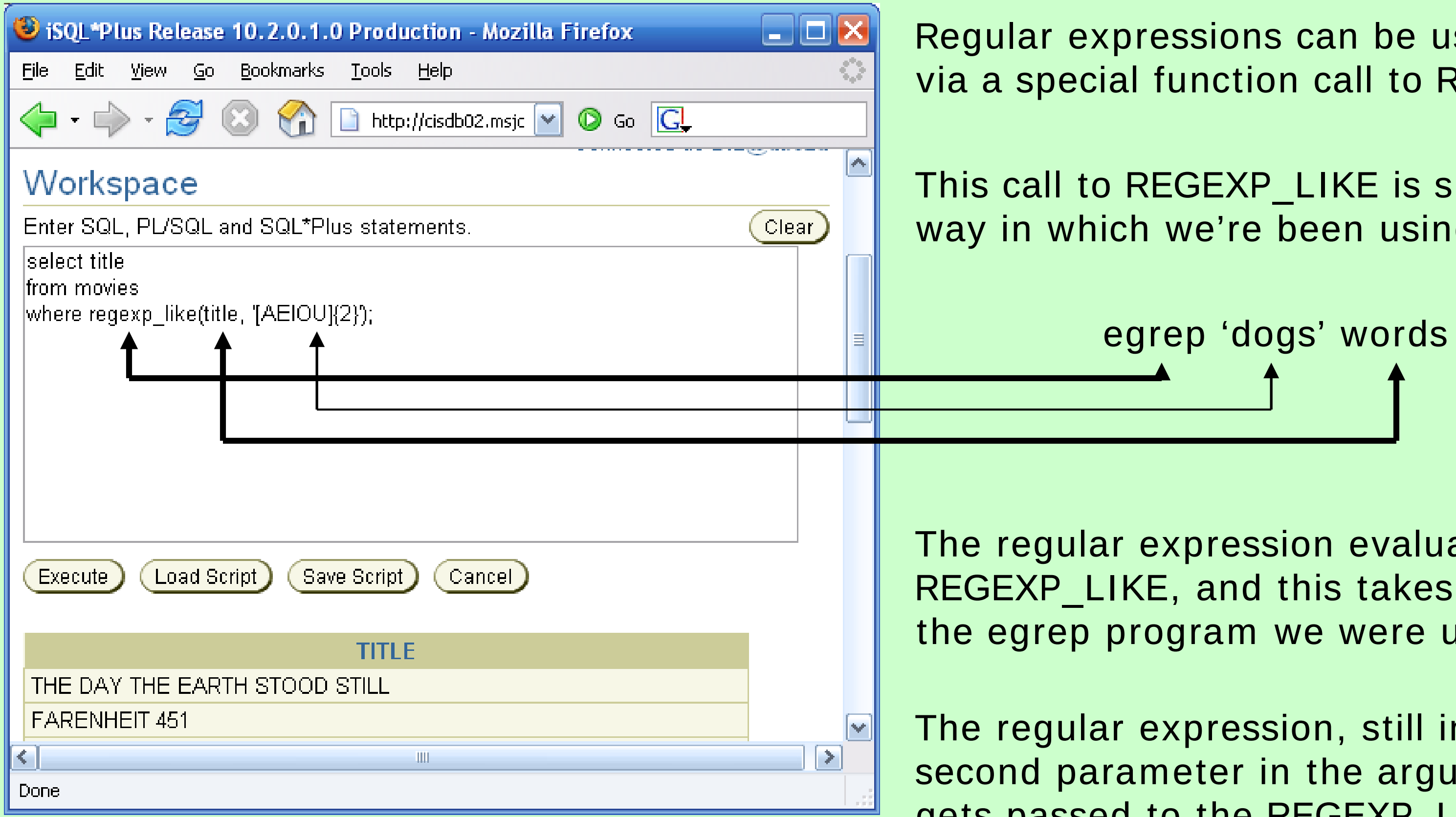
Let’s find all the lines in Hamlet, where two or more o’s are followed in the same line by two or more o’s.

```
egrep ‘oo+.*oo+’ hamlet.txt
```

This expression first looks for a pair of o’s followed by any number of characters, followed by a pair of o’s

Module 06: Character Data

Page E-01: Oracle RegEx


iSQL*Plus Release 10.2.0.1.0 Production - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://cisdb02.msjc
Go
Workspace
Clear
Execute
Load Script
Save Script
Cancel
Done
egrep ‘dogs’ words

Enter SQL, PL/SQL and SQL*Plus statements.

```
select title
from movies
where regexp_like(title, '[AEIOU]{2}');
```

TITLE
THE DAY THE EARTH STOOD STILL
FARENHEIT 451

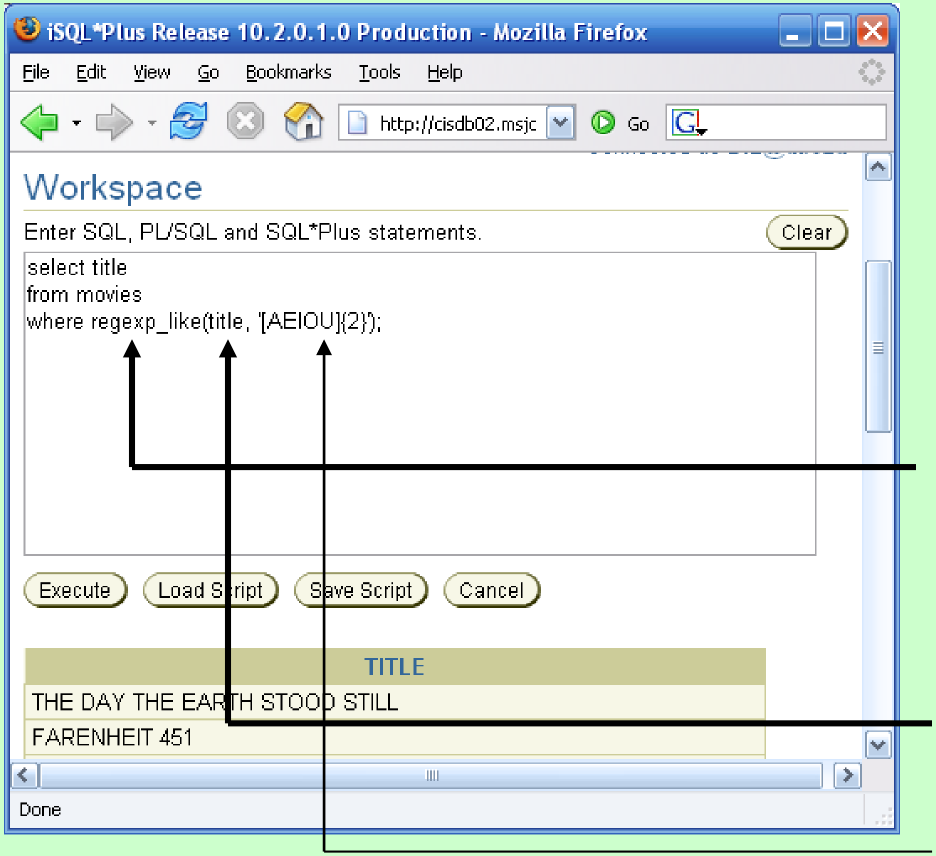
Regular expressions can be used in Oracle, via a special function call to REGEXP_LIKE.

This call to REGEXP_LIKE is similar to the way in which we’re been using egrep.

The regular expression evaluator in Oracle is REGEXP_LIKE, and this takes the place of the egrep program we were using earlier.

The regular expression, still in quotes, is the second parameter in the argument list that gets passed to the REGEXP_LIKE function, and the column to be checked is the first parameter in the argument list.

Page E-02: Oracle RegEx



iSQL*Plus Release 10.2.0.1.0 Production - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://cisdb02.msjc
Go
Workspace
Enter SQL, PL/SQL and SQL*Plus statements.
Clear
Execute
Load Script
Save Script
Cancel
Done

```
select title
from movies
where regexp_like(title, '[AEIOU]{2}');
```

TITLE
THE DAY THE EARTH STOOD STILL
FARENHEIT 451

Now I just mentioned a few technical terms that I haven't really briefed you on. I'll cover those in more detail in a later module.

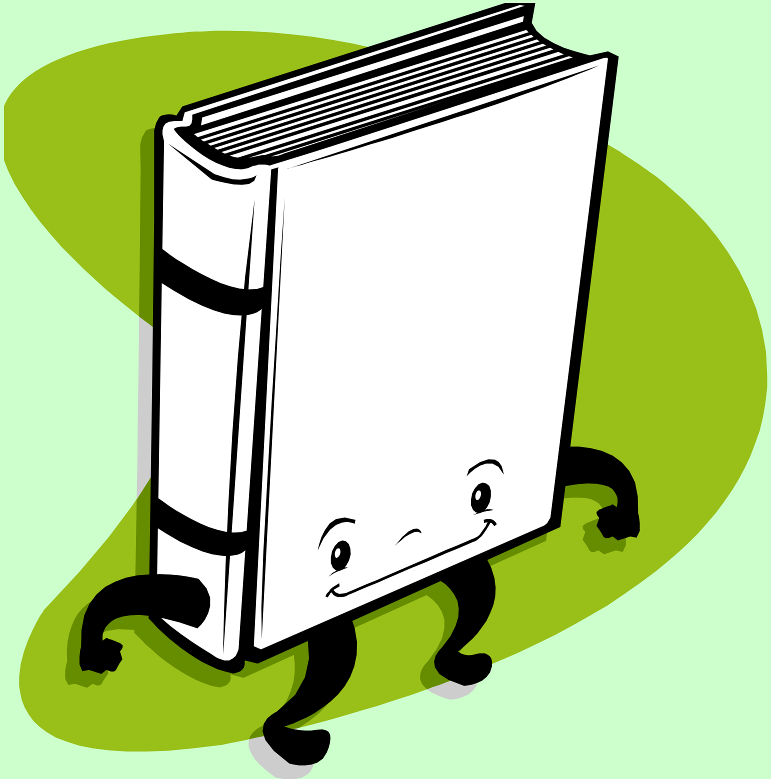
But for now, you can use regular expressions in Oracle, if you follow this 'recipe':

In the predicate expression, start with: REGEXP_LIKE( ). This is a function call to a *function* in Oracle that knows how to deal with regular expressions.

Then fill in the parentheses with two things. The name of the column you're checking (this should be first), then the regular expression that you want to use (and this should be second. Use a comma to separate the two.

And remember to use quote marks around the regular expression.

Page T-1: Terminology



LIKE
Wildcard characters, _, %

Special case: NOT LIKE

Regular expressions, regex
Literal pattern, static pattern

Range expression
Character set, character class

Boolean operators (^, |)

Page X-1: End Notes

The title for this module: ‘Like is, Like, Like, ya Know’ was inspired by the paper ‘Like is, Like, Focus’ written by Robert Underhill, San Diego State University and appearing in the Journal: American Speech 63.3 (1988).

Did I mention that I have a degree in Linguistics? The short story is that I’ve always been fascinated by the structure of language, and would you believe it, Linguistics is a Science, the science of language. (Although the way I write, no one would ever believe that I’ve studied anything at all about language ☺)

If you enjoy problem solving, and mental exercises (*like* most programmers do) you might be interested in taking a Linguistics class or two. Especially if you’ve got some general ed courses left to get out of the way.

Mind you, I don’t get a commission on student registrations, but I found Linguistics fun, and on the off chance that you too might enjoy it, I’m bringing it to your attention.

Page X-2: End Notes

And speaking of END notes, it turns out that the database administrator has been recording all SQL queries for a database tuning project he’s working on.

Your query about shortest last names caught his eye and he forwarded a report to your manager.

Your manager has written a formal reprimand into your personnel file, and you must attend a 4hr Personnel Training session on the Acceptable Use Policy (without pay) scheduled for this weekend ☺.

Note: in all likelihood, this kind of stuff happens all the time, you should figure that ANYTHING you do on a wrk, or school, workstation will be recorded and held against you.

Page Y-1: Sources

The data files that were used to demonstrate some of the regular expression concepts were downloaded from the Project Gutenberg .

You can find out more about this project from their website:
http://www.gutenberg.org/wiki/Main_Page

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