8

Regular Expression Support

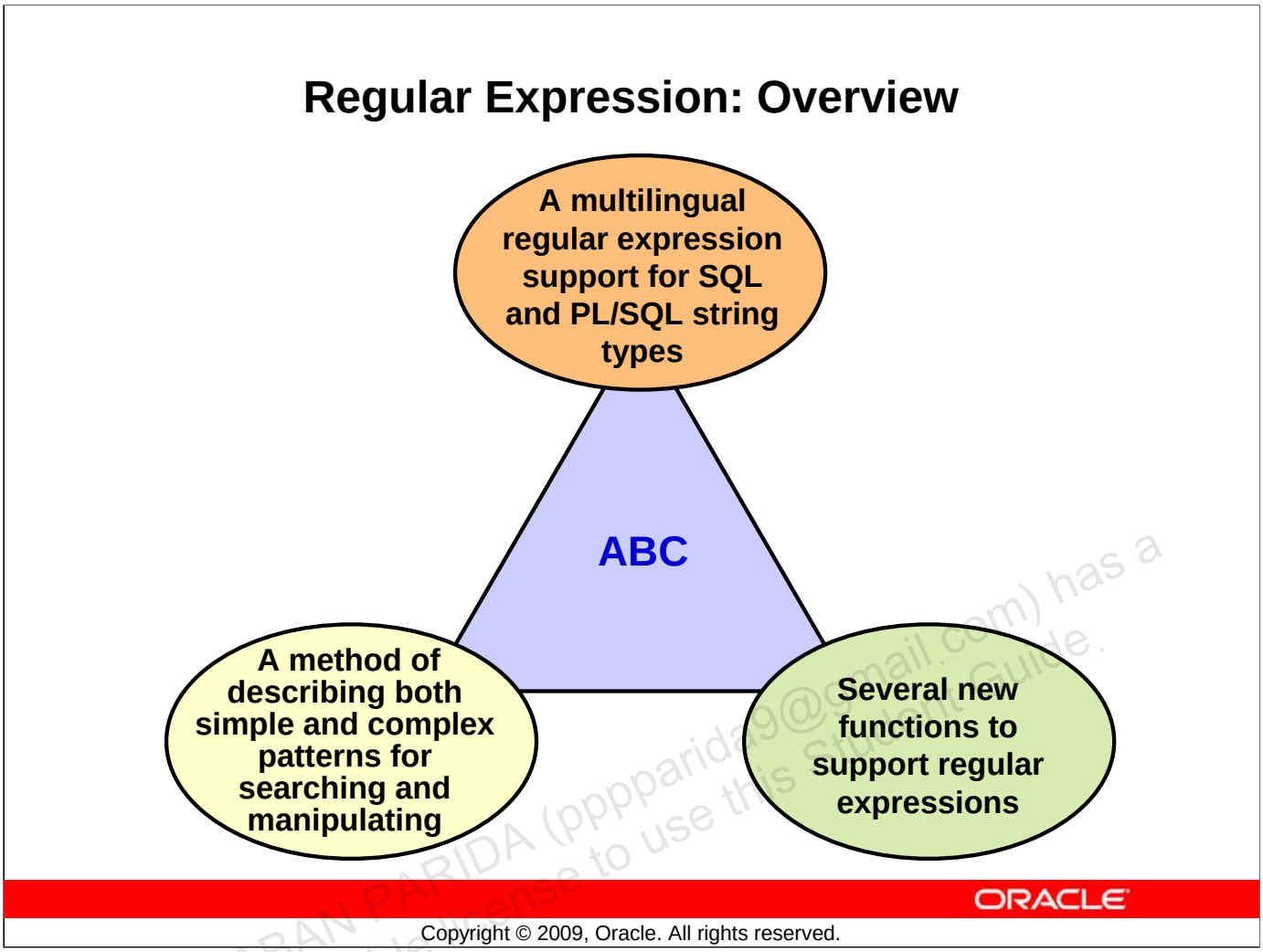
ORACLE

Objectives

After completing this lesson, you should be able to use regular expression support in SQL to search, match, and replace strings in terms of regular expressions.

Objectives

In this lesson, you learn to use the regular expression support feature that has been introduced in Oracle Database 10*g*.



Regular Expression: Overview

Oracle Database 10*g* introduces support for regular expressions. The implementation complies with the Portable Operating System for UNIX (POSIX) standard, controlled by the Institute of Electrical and Electronics Engineers (IEEE), for ASCII data matching semantics and syntax. Oracle's multilingual capabilities extend the matching capabilities of the operators beyond the POSIX standard. Regular expressions are a method of describing both simple and complex patterns for searching and manipulating.

String manipulation and searching contribute to a large percentage of the logic within a Web-based application. Usage ranges from the simple (for example, find the word "San Francisco" in a specified text) to the complex (for example, extract all URLs from the text) to the more complex (for instance, find all words whose every second character is a vowel).

When coupled with native SQL, the use of regular expressions allows for very powerful search and manipulation operations on any data stored in an Oracle Database. You can use this feature to easily solve problems that would otherwise be very complex to program.

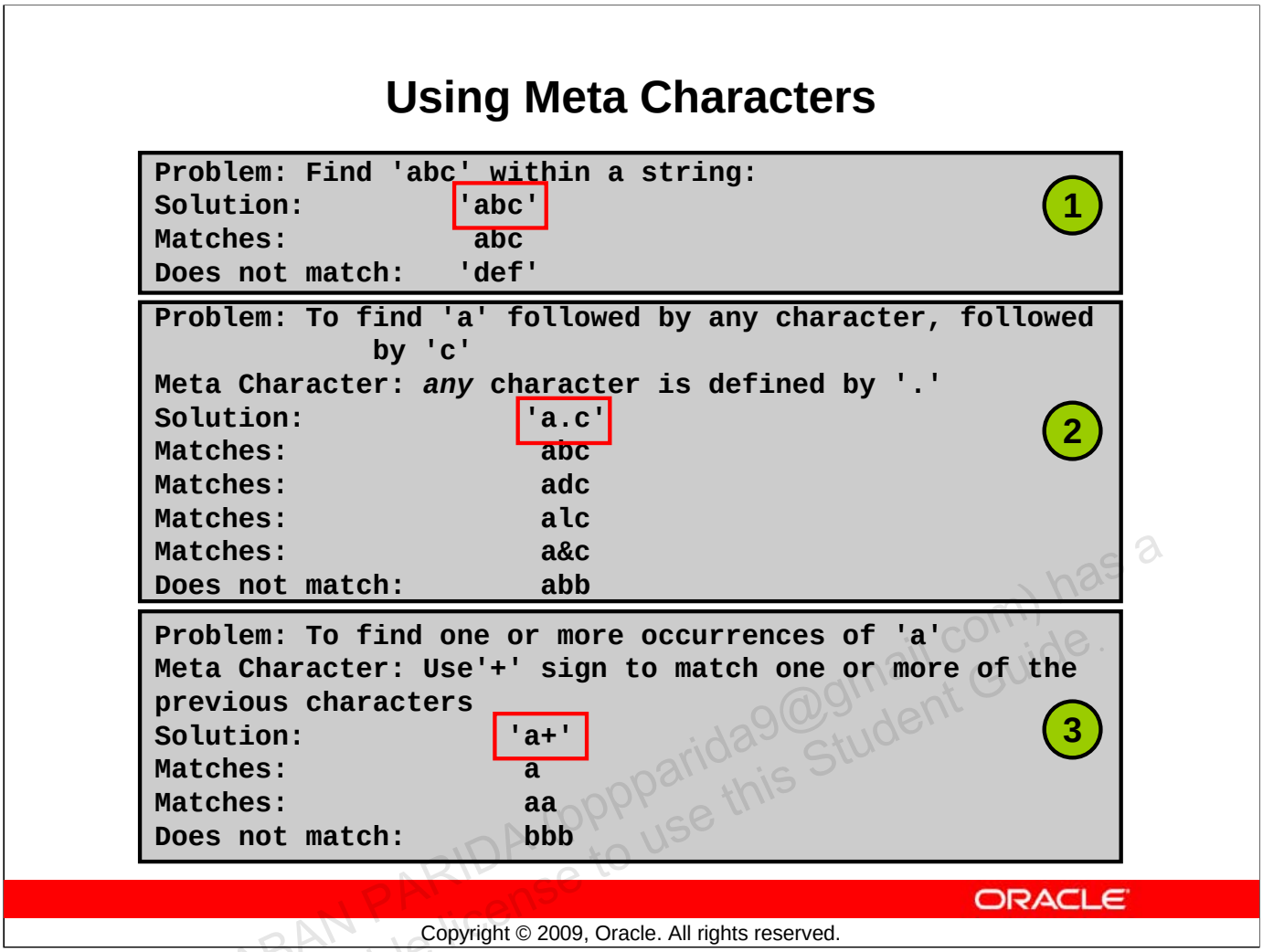
Meta Characters

Symbol	Description
*	Matches zero or more occurrences
|	Alteration operator for specifying alternative matches
^/$	Matches the start-of-line/end-of-line
[ ]	Bracket expression for a matching list matching any one of the expressions represented in the list
{m}	Matches exactly *m* times
{m,n}	Matches at least *m* times but no more than *n* times
[: :]	Specifies a character class and matches any character in that class
\	Can have 4 different meanings: 1. Stand for itself. 2. Quote the next character. 3. Introduce an operator. 4. Do nothing.
+	Matches one or more occurrences
?	Matches zero or one occurrence
.	Matches any character in the supported character set, except NULL
()	Grouping expression, treated as a single subexpression
[==]	Specifies equivalence classes
\n	Back-reference expression
[..]	Specifies one collation element, such as a multicharacter element

ORACLE

Meta Characters

Meta characters are special characters that have a special meaning, such as a wildcard character, a repeating character, a nonmatching character, or a range of characters. You can use several predefined meta character symbols in the pattern matching.



Using Meta Characters

1. In the first example, a simple match is performed.
2. In the second example, the any character is defined as a '.'. This example searches for the character "a" followed by any character, followed by the character "c".
3. The third example searches for one or more occurrences of the letter "a." The "+" character is used here to indicate a match of one or more of the previous characters.

You can search for nonmatching character lists too. A nonmatching character list allows you to define a set of characters for which a match is invalid. For example, to find anything but the characters "a," "b," or "c," you can define the "^" to indicate a nonmatch.

```
Expression:        [^abc]
  Matches:         abcdef
  Matches:         ghi
  Does not match:  abc
```

To match any letter not between "a" and "i," you can use:

```
Expression:        [^a-i]
  Matches:         hijk
  Matches:         lmn
  Does not match:  abcdefghi
```

Using Meta Characters (continued)

Meta Character Syntax	Operator Name	Description
.	Any Character – Dot	Match any character.
+	One or More – Plus Quantifier	Match one or more occurrences of the preceding subexpression.
?	Zero or One – Question Mark Quantifier	Match zero or one occurrence of the preceding subexpression.
*****	Zero or More – Star Quantifier	Match zero or more occurrences of the preceding subexpression.
{m} **{m,}** **{m,n}**	Interval – Exact Count	Match • exactly *m* occurrences • at least *m* occurrences • at least *m*, but not more than *n* occurrences of the preceding subexpression
[…]	Matching Character List	Match any character in list ...
[^…]	Non-Matching Character List	Match any character not in list ...
|	Or	'a|b' matches character 'a' or 'b'.
(…)	Subexpression or Grouping	Treat expression ... as a unit.
\n	Back reference	Match the n^{th} preceding subexpression, where *n* is an integer from 1 to 9.
\	Escape Character	Treat the subsequent meta character in the expression as a literal.
^	Beginning of Line Anchor	Match the subsequent expression when it occurs at the beginning of a line.
$	End of Line Anchor	Match the preceding expression only when it occurs at the end of a line.
[:class:]	POSIX Character Class	Match any character belonging to the specified character *class*.

Regular Expression Functions

Function Name	Description
REGEXP_LIKE	Similar to the LIKE operator, but performs regular expression matching instead of simple pattern matching
REGEXP_REPLACE	Searches for a regular expression pattern and replaces it with a replacement string
REGEXP_INSTR	Searches for a given string for a regular expression pattern and returns the position where the match is found
REGEXP_SUBSTR	Searches for a regular expression pattern within a given string and returns the matched substring

Regular Expression Functions

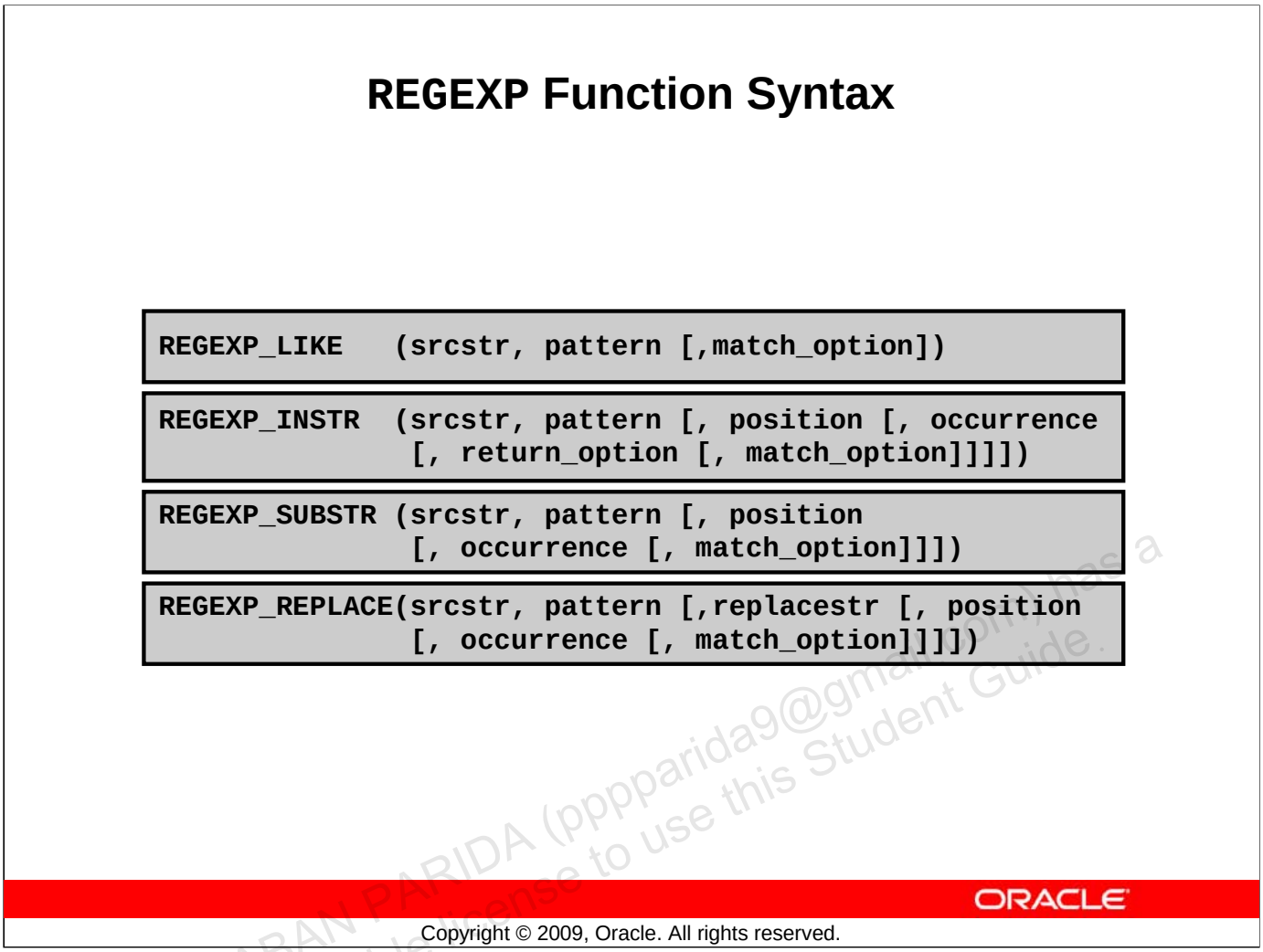
Oracle Database 10*g* provides a set of SQL functions that you can use to search and manipulate strings using regular expressions. You can use these functions on any data type that holds character data such as CHAR, NCHAR, CLOB, NCLOB, NVARCHAR2, and VARCHAR2. A regular expression must be enclosed or wrapped between single quotation marks. Doing so ensures that the entire expression is interpreted by the SQL function and can improve the readability of your code.

REGEXP_LIKE: This function searches a character column for a pattern. Use this function in the WHERE clause of a query to return rows matching the regular expression you specify.

REGEXP_REPLACE: This function searches for a pattern in a character column and replaces each occurrence of that pattern with the pattern you specify.

REGEXP_INSTR: This function searches a string for a given occurrence of a regular expression pattern. You specify which occurrence you want to find and the start position to search from. This function returns an integer indicating the position in the string where the match is found.

REGEXP_SUBSTR: This function returns the actual substring matching the regular expression pattern you specify.



REGEXP Function Syntax

The following table contains descriptions of the terms shown in the syntax in the slide:

`srcstr`	Search value
`pattern`	Regular expression
`occurrence`	Occurrence to search for
`position`	Search starting position
`return_option`	Start or end position of occurrence
`replacestr`	Character string replacing pattern
`match_option`	Option to change default matching; it can include one or more of the following values: “`c`” uses case-sensitive matching (default). “i” uses non-case-sensitive matching. “`n`” allows match-any-character operator. “`m`” treats source string as multiple line.

Performing Basic Searches

```
SELECT first_name, last_name
FROM employees
WHERE REGEXP_LIKE (first_name, '^Ste(v|ph)en$');
```

	FIRST_NAME	LAST_NAME
1	Steven	King
2	Steven	Markle
3	Stephen	Stiles

ORACLE

Example of REGEXP_LIKE

In this query, against the EMPLOYEES table, all employees with first names containing either Steven or Stephen are displayed. In the expression used,

'^Ste(v|ph)en$' :

- ^ indicates the beginning of the sentence
- $ indicates the end of the sentence
- | indicates either/or

Checking the Presence of a Pattern

```
SELECT street_address,
   REGEXP_INSTR(street_address,'[^[:alpha:]]')
FROM   locations
WHERE
   REGEXP_INSTR(street_address,'[^[:alpha:]]')> 1;
```

	STREET_ADDRESS	REGEXP_INSTR(STREET_ADDRESS,'[^[:ALPHA:]]')
1	Magdalen Centre, The Oxford Science Park	9
2	Schwanthalerstr. 7031	16
3	Rua Frei Caneca 1360	4
4	Murtenstrasse 921	14
5	Pieter Breughelstraat 837	7
6	Mariano Escobedo 9991	8

ORACLE

Checking the Presence of a Pattern

In this example, the `REGEXP_INSTR` function is used to search the street address to find the location of the first nonalphabetic character, regardless of whether it is in uppercase or lowercase. The search is performed only on those addresses that do not start with a number. Note that `[:<class>:]` implies a character class and matches any character from within that class, and `[:alpha:]` matches with any alphabetic character. The results are displayed.

In the expression used in the query **`'[^[:alpha:]]'`**:

- [starts the expression
- ^ indicates NOT
- [:alpha:] indicates alpha character class
-] ends the expression

Note: The POSIX character class operator enables you to search for an expression within a character list that is a member of a specific POSIX character class. You can use this operator to search for specific formatting, such as uppercase characters, or you can search for special characters such as digits or punctuation characters. The full set of POSIX character classes is supported. Use the syntax [:*class*:] where *class* is the name of the POSIX character class to search for. The following regular expression searches for one or more consecutive uppercase characters: `[[:upper:]]+` .

Example of Extracting Substrings

```
SELECT REGEXP_SUBSTR(street_address , ' [^ ]+ ')
"Road" FROM locations;
```

	Road
1	Via
2	Calle
3	(null)
4	(null)
5	Jabberwocky
6	Interiors
7	Zagora
8	Charade
9	Spadina
10	Boxwood

...

Example of Extracting a Substring

In this example, the road names are extracted from the `LOCATIONS` table. To do this, the contents in the `STREET_ADDRESS` column that are before the first space are returned using the `REGEXP_SUBSTR` function. In the expression used in the query `'[^ ]+ '`:

- [starts the expression
- ^ indicates NOT
- indicates space
-] ends the expression
- + indicates 1 or more
- indicates space

Replacing Patterns

```
SELECT REGEXP_REPLACE( country_name, '(.)',
                       '\1 ') "REGEXP_REPLACE"
FROM countries;
```

	REGEXP_REPLACE
1	Argentina
2	Australia
3	Belgium
4	Brazil
5	Canada
6	Switzerland
7	China
8	Germany
9	Denmark
10	Egypt

...

Replacing Patterns

This example examines `COUNTRY_NAME`. The Oracle Database reformats this pattern with a space after each non-null character in the string. The results are shown.

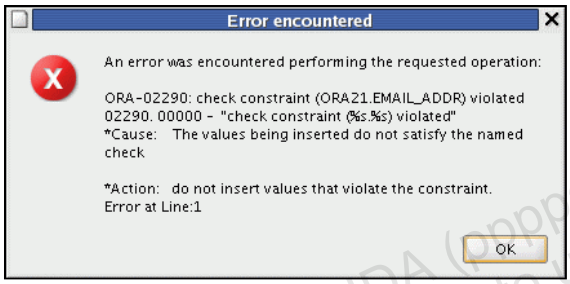
Regular Expressions and Check Constraints

```
ALTER TABLE emp8
  ADD CONSTRAINT email_addr
  CHECK(REGEXP_LIKE(email,'@'))NOVALIDATE ;
```

1

```
INSERT INTO emp8 VALUES
  (500,'Christian','Patel',
  'ChrisP2creme.com', 1234567890,
   '12-Jan-2004', 'HR_REP', 2000, null, 102, 40) ;
```

2



ORACLE

Regular Expressions and Check Constraints

Regular expressions can also be used in check constraints. In this example, a check constraint is added on the `EMAIL` column of the `EMPLOYEES` table. This ensures that only strings containing an "@" symbol are accepted. The constraint is tested. The check constraint is violated because the e-mail address does not contain the required symbol. The `NOVALIDATE` clause ensures that the existing data is not checked.

Summary

In this lesson, you should have learned how to use regular expression support in SQL and PL/SQL to search, match, and replace strings in terms of regular expressions.

Summary

In this lesson, you have learned to use the regular expression support features that have been introduced in Oracle Database 10*g*.

Practice 8: Overview

This practice covers using regular expressions.

Practice 8: Overview

This practices covers searching and replacing data using regular expressions.

Practice 8

1. Write a query to search the EMPLOYEES table for all employees whose first names start with "Ne" or "Na."

	FIRST_NAME	LAST_NAME
1	Nanette	Cambrault
2	Nancy	Greenberg
3	Neena	Kochhar
4	Nandita	Sarchand

2. Create a query that removes the spaces in the STREET_ADDRESS column of the LOCATIONS table in the display.

	REGEXP_REPLACE(STREET_ADDRESS,'','')
1	1297ViaColadiRie
2	93091CalledellaTesta
3	2017Shinjuku-ku
4	9450Kamiya-cho
5	2014JabberwockyRd
6	2011InteriorsBlvd
7	2007ZagoraSt
8	2004CharadeRd
9	147SpadinaAve
10	6092BoxwoodSt
11	40-5-12Laogianggen
12	1298Vileparle(E)
13	12-98VictoriaStreet
14	198ClementiNorth

...

18	Schwanthalerstr.7031
19	RuaFreiCaneca1360
20	20RuedesCorps-Saints
21	Murtenstrasse921
22	PieterBreughelstraat837
23	MarianoEscobedo9991

Practice 8 (continued)

3. Create a query that displays “St” replaced by “Street” in the STREET_ADDRESS column of the LOCATIONS table. Be careful that you do not affect any rows that already have “Street” in them. Display only those rows that are affected.

	REGEXP_REPLACE(STREET_ADDRESS,'ST$','STREET')
1	2007 Zagora Street
2	6092 Boxwood Street
3	12-98 Victoria Street
4	8204 Arthur Street