

**Mthree Alumni Training**



## Module 2 SQL and Finance

- Additional useful SQL functions



# SQL

## Objectives

While we will spend more time exploring the finance database, we will introduce a few new concepts in this module as well.

### Trainer Notes

We will review the new methods here then go through demos and exercises to practice

## The regexp\_like command

- The regexp\_like() command is treated like a function in Oracle Database and an operator in SQL standard. It allows you to search databases using regular expressions:

```
REGEXP_LIKE(source_string, search_pattern [, match_parameter]);
```

- Let's look at an example of searching for all values that start with s (case insensitive) in the column:

```
SELECT column FROM table WHERE REGEXP_LIKE (column, '^s', 'i');
```

- As with other regular expressions, you can use other special characters such as \$ or |

### Trainer/Trainee Notes

this is a useful search tool within the database if you are wanting to find a subset of records.

# Upper and Lower

As we know with real life data, we will always expect inconsistencies in accuracy of it. This can become an issue if you are searching for specific values and are unsure if the data was entered consistently.

The two functions `upper()` and `lower()` helps with any ambiguity around case.

```
SELECT UPPER(column) FROM table;
```

```
SELECT column FROM table WHERE  
UPPER(column) LIKE "EXAMPLE";
```

As you can see, we can use the function in multiple ways to convert its argument to upper case

```
SELECT LOWER(column) FROM table;
```

```
SELECT column FROM table WHERE  
LOWER(column) LIKE "EXAMPLE";
```

Here we convert the value to all lower case and then search on the result

## Trainer/Trainee Notes

This function is often used in conjunction with other functions – and is a helpful way to ensure you catch all instances of the value you are trying to find. Obviously you would want to use a regular expression search if you are concerned about spelling being incorrect in the data.

## INSTR function

The INSTR() function is another useful SQL tool for searching for strings within strings.

**INSTR(string1, string2)**

- String1 – the first string is the one that will be searched
- String2 – the pattern we are searching for

The method will return the position at which the string starts.

```
SELECT INSTR(column, "example") FROM table;
```

### Trainer/Trainee Notes

Another extremely useful search function across the database.

# Substring

SQL provides us with a function to return a substring of a value.

**SUBSTR (String, start, length)**

The function takes the string you wish to manipulate and then you provide it two numerical values, where the substring you require starts and what length it needs to be

Here is an example of how you use it:

```
SELECT SUBSTR('example', 3, 3) AS SubstringOutput;
```

(This would return *amp* as the result)

## Trainer/Trainee Notes

Notice in this example we did not even have to run the query against a table – you can however use it against a column if you want to.

## Replace

You may want to replace a substring with different values. The replace function with pattern matching allows you to this.

```
REPLACE(string, original, newValue)
```

```
SELECT REPLACE('Example string', 'string', 'strong');
```

Here we are simply replacing the word *string* with *strong* in the string "Example string"

### Trainer/Trainee Notes

Remember this is not going to change the underlying value in the database, only the results from your query.

## Coalesce Function

We have seen in some of our data that we can have **null** values. The coalesce function when given a list will return the first non **null** value.

For example, say you had a table containing personal data such as home phone, work phone and cell – perhaps you wish to just return the first value that is not null for each person.

```
SELECT COALESCE(null, null, 'A', null, 'B');
```

This will return A as it is the first non-null value.

### Trainer/Trainee Notes

Another useful command – we will demo it later and how you may find it useful.



# SQL Case Statements

Like we see in most programming languages, SQL allows us to use case statements to achieve an outcome.

```
CASE
  WHEN condition1 THEN result1
  WHEN condition2 THEN result2
  WHEN condition_x THEN result_x
  ELSE result
END;
```

Conditions are evaluated in the order they are listed, and it is always important to ensure you have a catch all ELSE statement at the end.

## Trainer/Trainee Notes

You can use case statements in various places of your query including your order by clause and in the main part of the query.