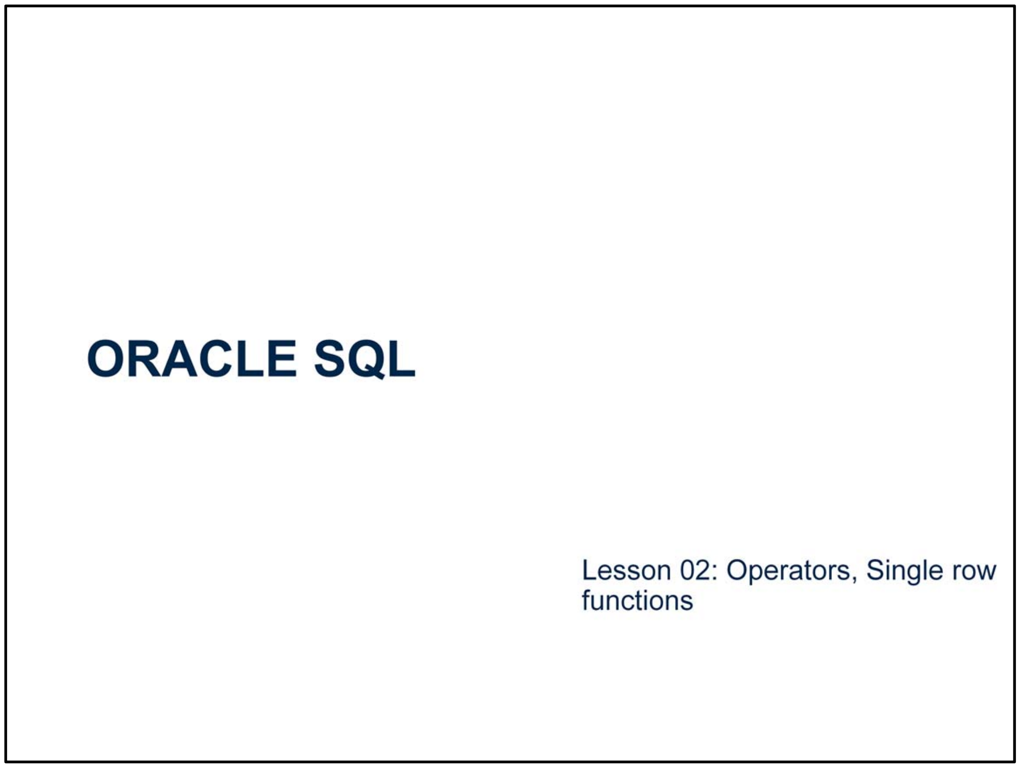
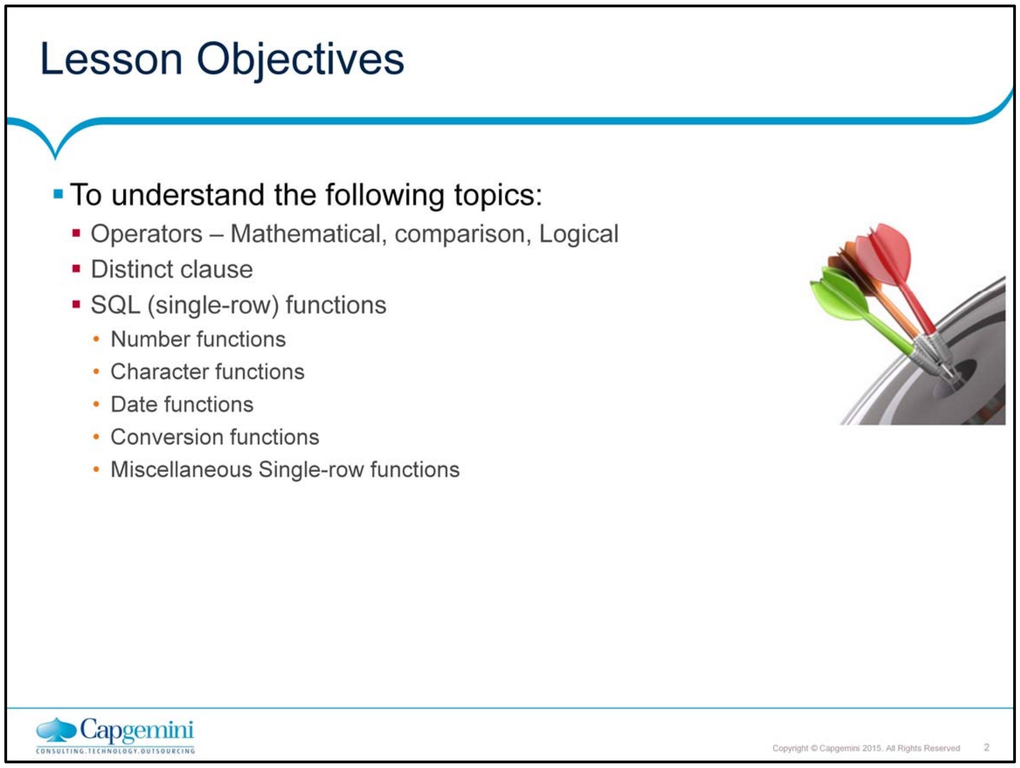


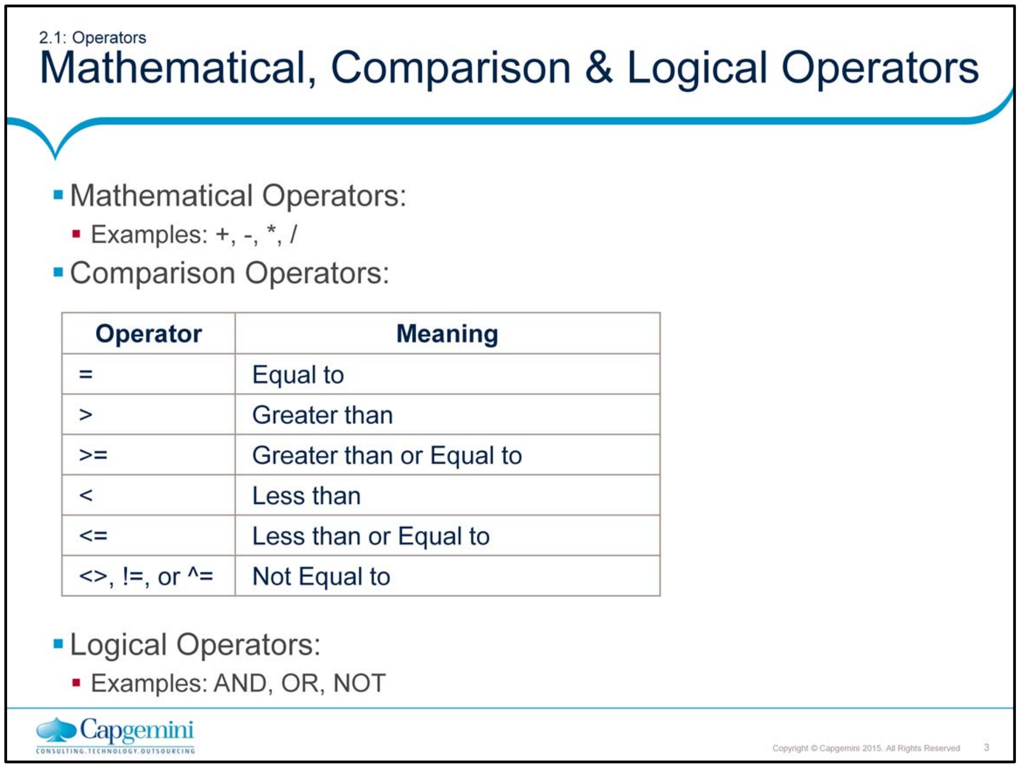
ORACLE SQL

Lesson 02: Operators, Single row functions



Lesson Objectives

- To understand the following topics:
  - Operators – Mathematical, comparison, Logical
  - Distinct clause
  - SQL (single-row) functions
    - Number functions
    - Character functions
    - Date functions
    - Conversion functions
    - Miscellaneous Single-row functions


2.1: Operators

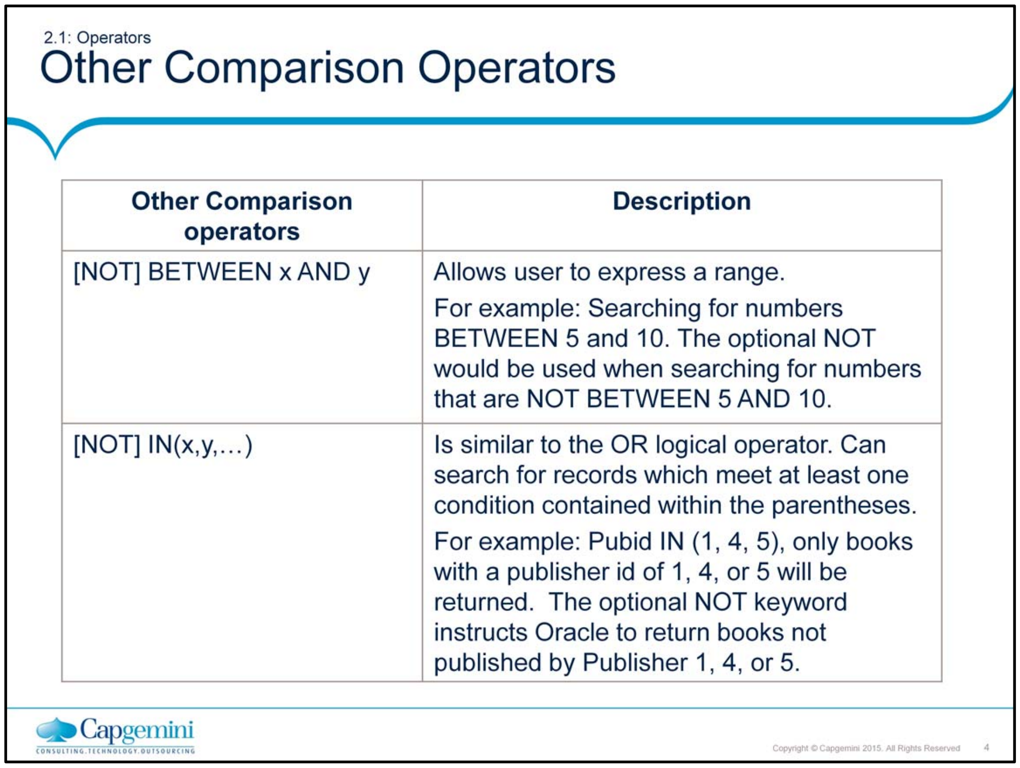
Mathematical, Comparison & Logical Operators

- Mathematical Operators:
  - Examples: +, -, *, /
- Comparison Operators:

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or Equal to
<	Less than
<=	Less than or Equal to
<>, !=, or ^=	Not Equal to

- Logical Operators:
  - Examples: AND, OR, NOT

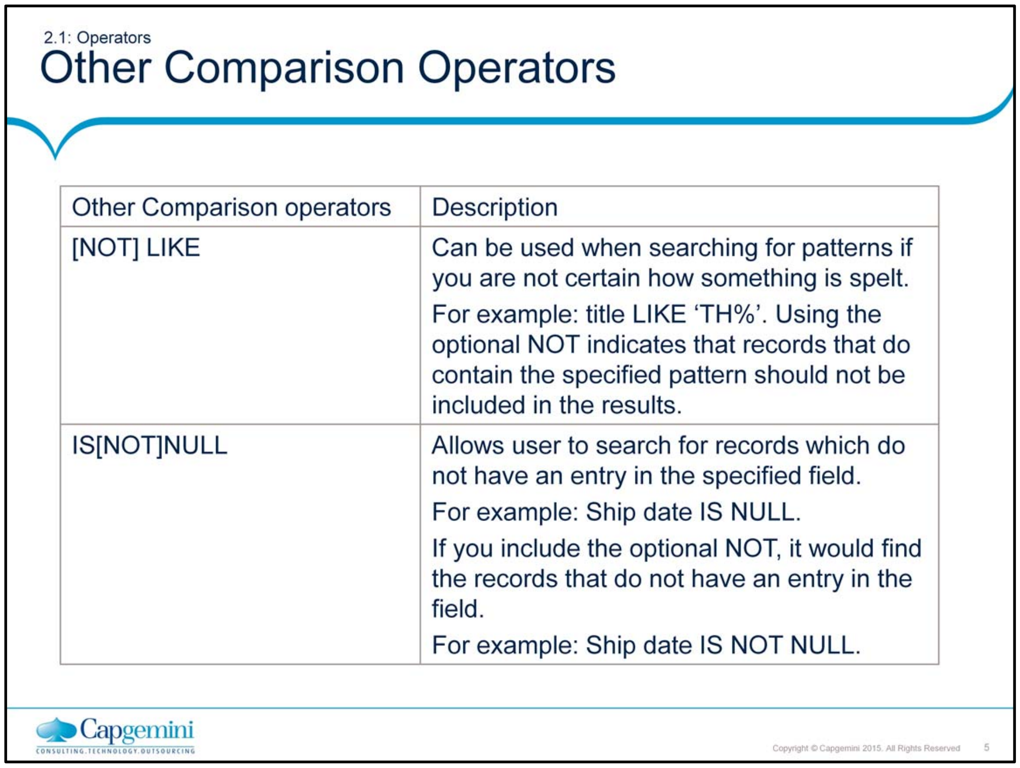
Operators:
Operators are used in “expressions” or “conditional statements”. They show equality, inequality, or a combination of both.
Operators are of three types:
mathematical
logical
range (comparison)
These operators are mainly used in the WHERE clause, HAVING clause in order to filter the data to be selected.
Mathematical operators:
These operators add, subtract, multiply, divide, and compare equality of numbers and strings. They are +, -, *, /
Comparison Operators:
These operators are used to compare the column data with specific values in a condition. “Comparison Operators” are also used along with the “SELECT statement” to filter data based on specific conditions. The table in the slide describes each Comparison operator. Comparison operators indicate how the data should relate to the given search value.
Logical Operators:
There are three Logical Operators namely AND, OR and NOT. These operators compare two conditions at a time to determine whether a row can be selected for the output or not. When retrieving data by using a SELECT statement, you can use logical operators in the WHERE clause. This allows you to combine more than one condition.



2.1: Operators

Other Comparison Operators

Other Comparison operators	Description
[NOT] BETWEEN x AND y	Allows user to express a range. For example: Searching for numbers BETWEEN 5 and 10. The optional NOT would be used when searching for numbers that are NOT BETWEEN 5 AND 10.
[NOT] IN(x,y,…)	Is similar to the OR logical operator. Can search for records which meet at least one condition contained within the parentheses. For example: Pubid IN (1, 4, 5), only books with a publisher id of 1, 4, or 5 will be returned. The optional NOT keyword instructs Oracle to return books not published by Publisher 1, 4, or 5.



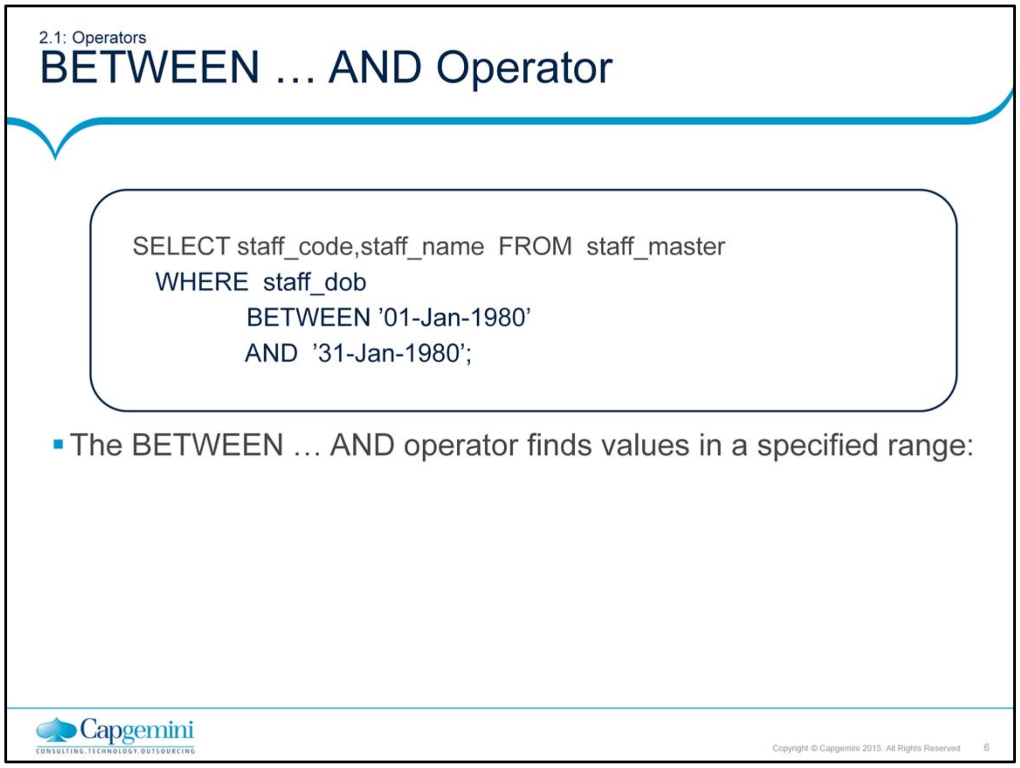
2.1: Operators

Other Comparison Operators

Other Comparison operators	Description
[NOT] LIKE	Can be used when searching for patterns if you are not certain how something is spelt. For example: title LIKE 'TH%'. Using the optional NOT indicates that records that do contain the specified pattern should not be included in the results.
IS[NOT]NULL	Allows user to search for records which do not have an entry in the specified field. For example: Ship date IS NULL. If you include the optional NOT, it would find the records that do not have an entry in the field. For example: Ship date IS NOT NULL.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

5

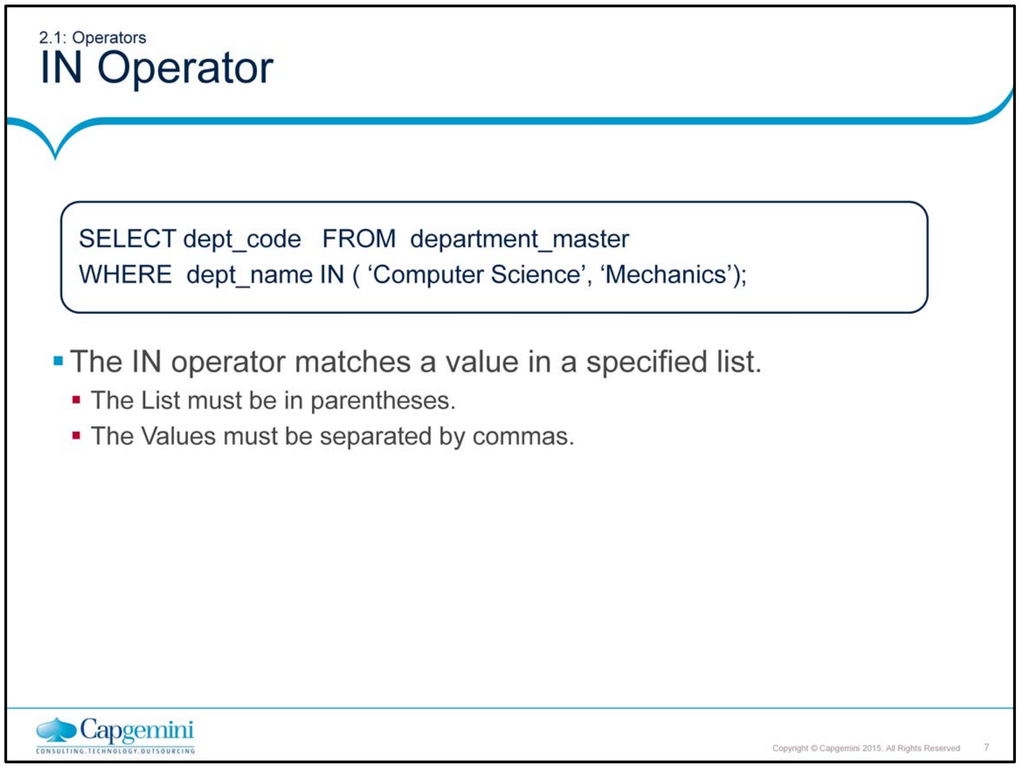


2.1: Operators

BETWEEN … AND Operator

```
SELECT staff_code,staff_name  FROM  staff_master
  WHERE  staff_dob
           BETWEEN ’01-Jan-1980’
           AND  ’31-Jan-1980’;
```

- The BETWEEN … AND operator finds values in a specified range:



2.1: Operators

IN Operator

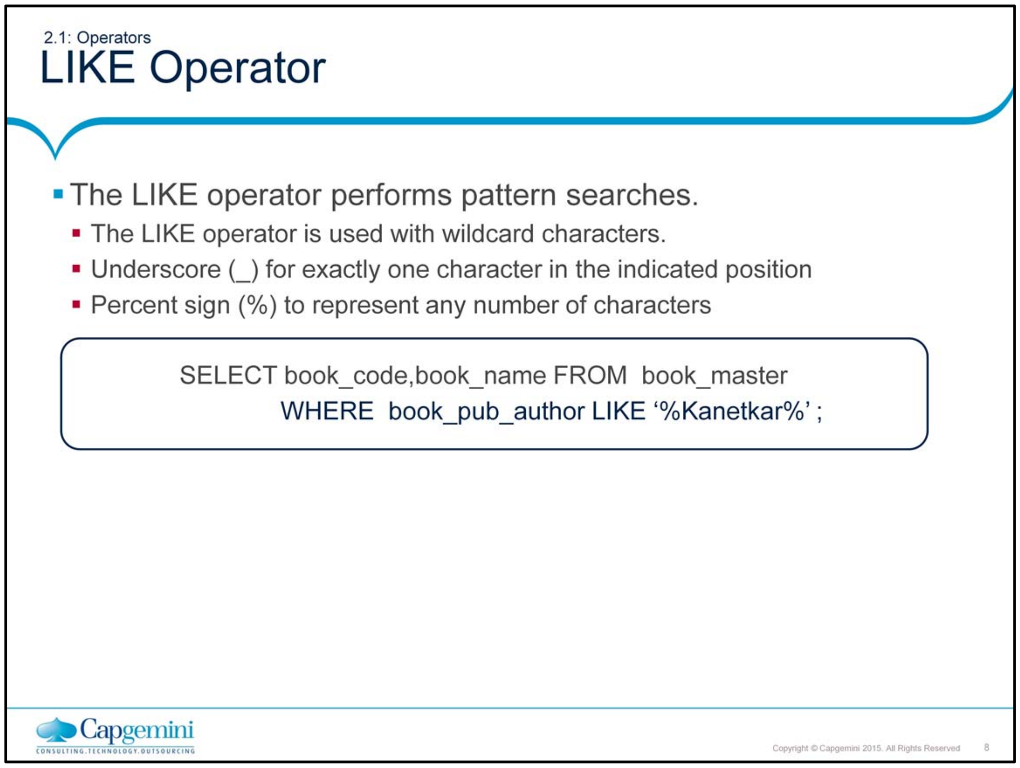
```
SELECT dept_code   FROM  department_master
WHERE  dept_name IN ( 'Computer Science', 'Mechanics');
```

- The IN operator matches a value in a specified list.
  - The List must be in parentheses.
  - The Values must be separated by commas.

IN predicate:
It is of the form:

```
<Expression> IN <LIST>
<Expression> IN <SUBQUERY>
```

The data types should match.



2.1: Operators

LIKE Operator

- The LIKE operator performs pattern searches.
  - The LIKE operator is used with wildcard characters.
  - Underscore (_) for exactly one character in the indicated position
  - Percent sign (%) to represent any number of characters

```
SELECT book_code,book_name FROM  book_master
        WHERE  book_pub_author LIKE '%Kanetkar%' ;
```

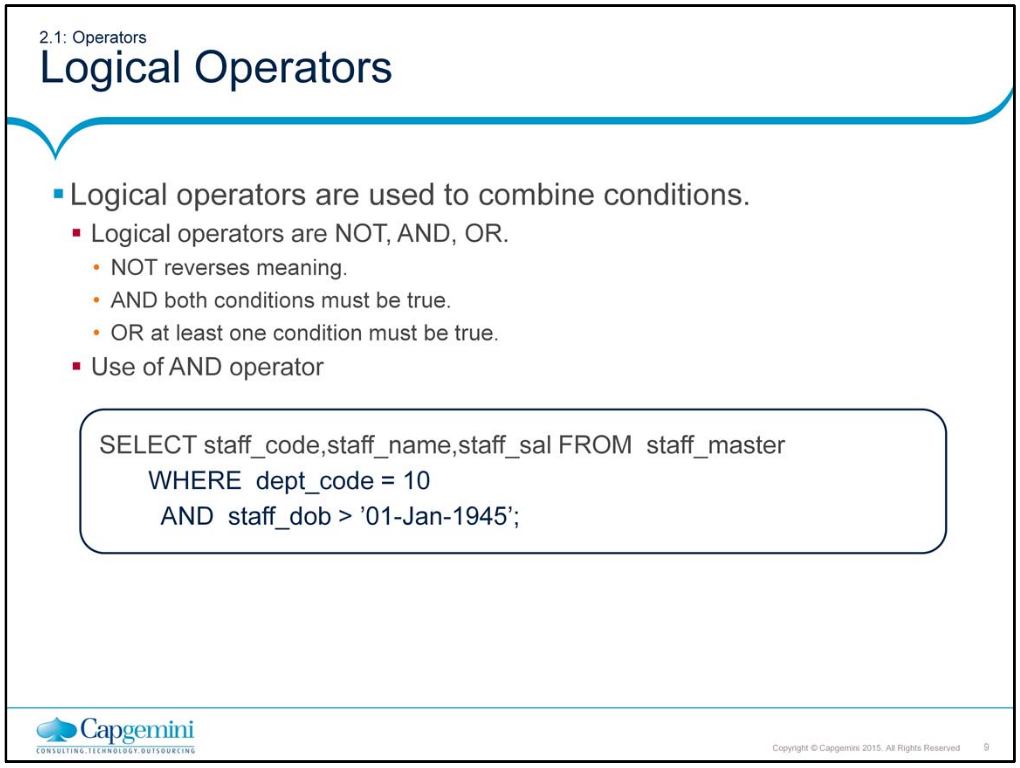
Copyright © Capgemini 2015. All Rights Reserved 8

LIKE predicate:
It is of the form:

<COLUMN > LIKE < PATTERN>

The pattern contains a search string along with other special characters % and _. The % character represents a string of any length where as _ (underscore) represents exactly one character.
A pattern %XYZ% means search has to be made for string XYZ in any position. A pattern '_XYZ%' means search has to be made for string XYZ in position 2 to 4.
To search for characters % and _ in the string itself we have to use an "escape" character.

For example: To search for string NOT_APP in column status, we have to use the form Status like 'NOT_APP' ESCAPE '\'
The use of \ as escape character is purely arbitrary.



2.1: Operators

Logical Operators

- Logical operators are used to combine conditions.
  - Logical operators are NOT, AND, OR.
    - NOT reverses meaning.
    - AND both conditions must be true.
    - OR at least one condition must be true.
  - Use of AND operator

```
SELECT staff_code,staff_name,staff_sal FROM  staff_master
      WHERE  dept_code = 10
       AND  staff_dob > '01-Jan-1945';
```

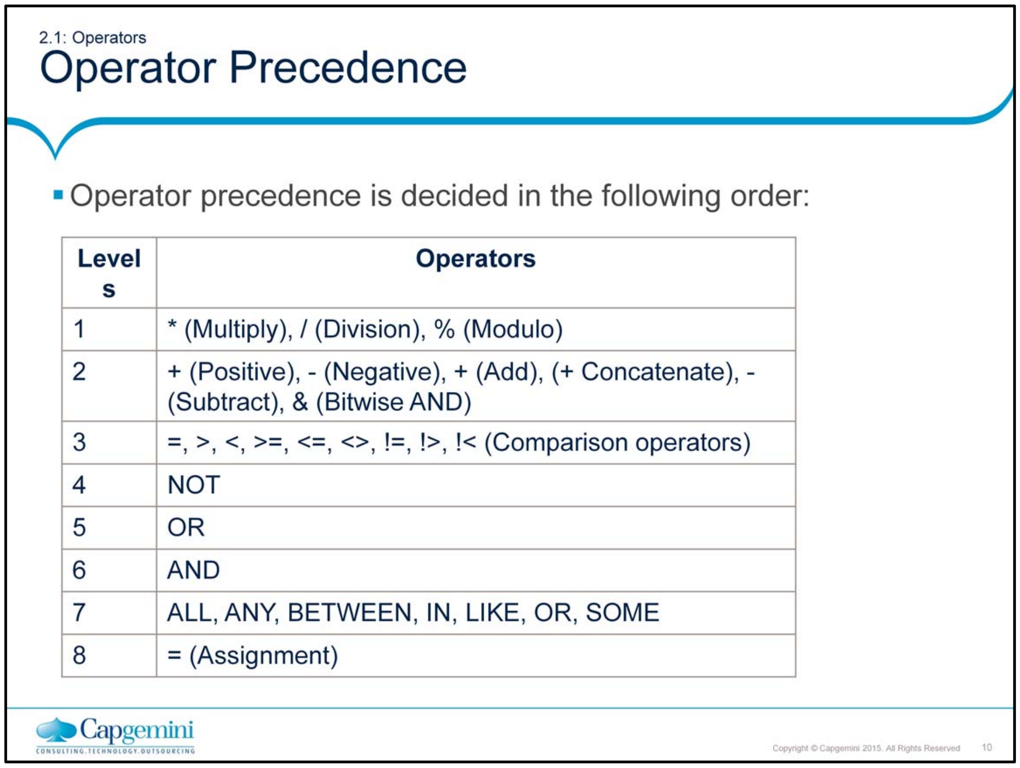
Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

The AND operator displays a record if both the first condition and the second condition is true.
One More Example:

```
SQL>   SELECT title, pubid, category
   2   FROM books
   3   WHERE pubid = 3
   4   AND category = 'COMPUTER';
```

Combining Predicates by using Logical Operators:
The predicates can be combined by using logical operators like AND, OR, NOT.
The evaluation proceeds from left to right and order of evaluation is:

* Enclosed in parenthesis
AND
OR



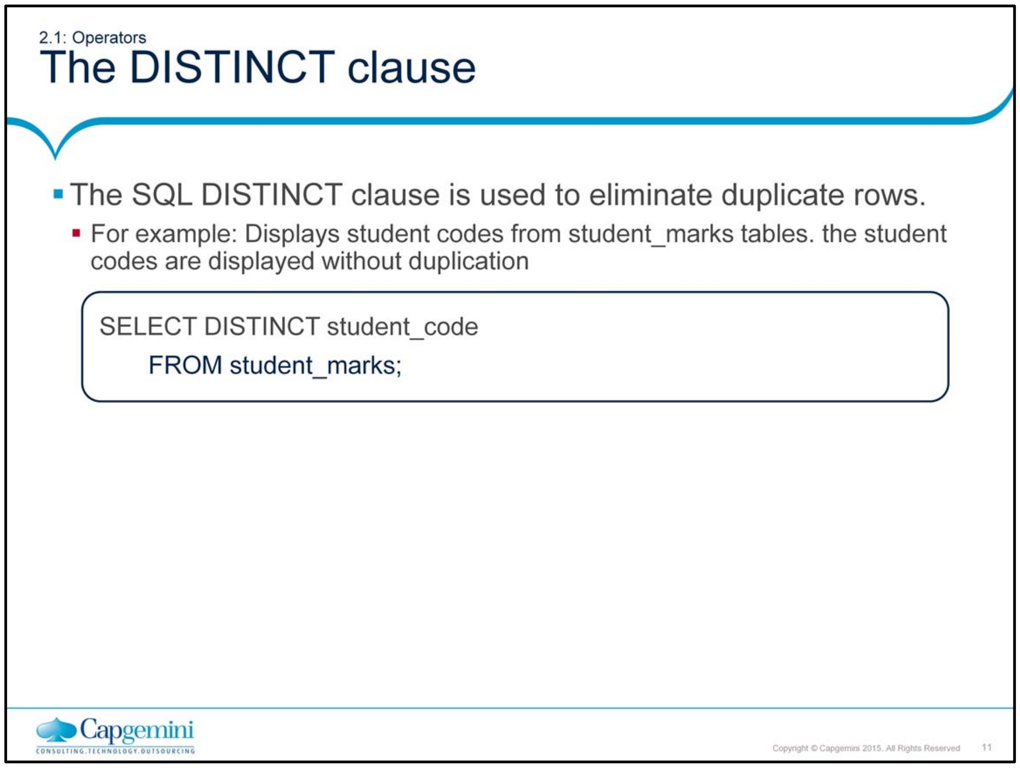
2.1: Operators

Operator Precedence

- Operator precedence is decided in the following order:

Level s	Operators
1	* (Multiply), / (Division), % (Modulo)
2	+ (Positive), - (Negative), + (Add), (+ Concatenate), - (Subtract), & (Bitwise AND)
3	=, >, <, >=, <=, <>, !=, !>, !< (Comparison operators)
4	NOT
5	OR
6	AND
7	ALL, ANY, BETWEEN, IN, LIKE, OR, SOME
8	= (Assignment)

Operator Precedence:
When a complex expression has multiple operators, the operator precedence (or order of execution of operators) determines the sequence in which the operations are performed.
The order of execution can significantly affect the resulting value.
The operators have the precedence levels as shown in the table given in the slide.
An operator on higher levels is evaluated before an operator on lower level.



2.1: Operators

The DISTINCT clause

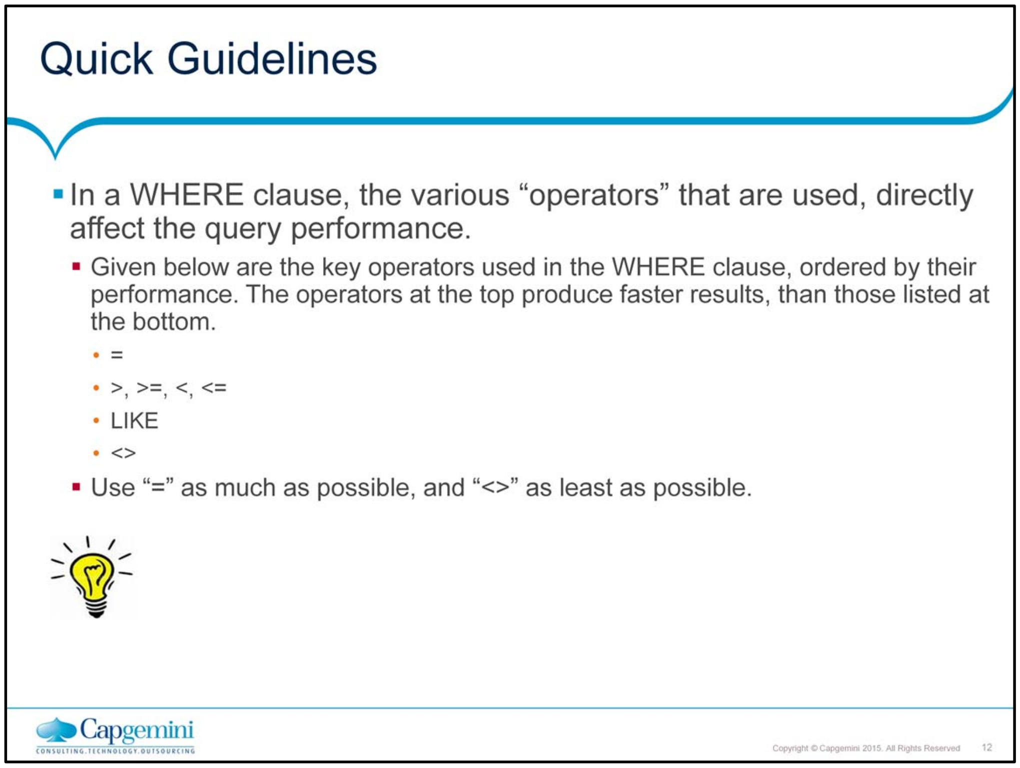
- The SQL DISTINCT clause is used to eliminate duplicate rows.
  - For example: Displays student codes from student_marks tables. the student codes are displayed without duplication

```
SELECT DISTINCT student_code
      FROM student_marks;
```

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

The DISTINCT clause:
In the examples discussed so far, some of the values have been repeated. However, by default, all values are retrieved. If you wish to remove duplicate values, then use the query as shown in the slide above.

Retrieval of Constant values by using Dual Table
A “dual” is a table, which is created by Oracle along with the data dictionary. It consists of exactly one column, whose name is dummy, and one record. The value of that record is X.



Quick Guidelines

- In a WHERE clause, the various “operators” that are used, directly affect the query performance.
  - Given below are the key operators used in the WHERE clause, ordered by their performance. The operators at the top produce faster results, than those listed at the bottom.
    - =
    - >, >=, <, <=
    - LIKE
    - <>
  - Use “=” as much as possible, and “<>” as least as possible.

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

Tips and Tricks in SELECT Statements (contd.):

Use simple operands

Some operators tend to produced speedy results than other operators. Of course, you may not have choice of using an operator in your WHERE clauses, but sometimes you do have a choice.

- Using simpler operands, and exact numbers, provides the best overall performance.
- If a WHERE clause includes multiple expressions, there is generally no performance benefit gained by ordering the various expressions in any particular order.
  - This is because the Query Optimizer does this for you, saving you the effort. There are a few exceptions to this, which are discussed further in the lesson.

contd.

Tips and Tricks in SELECT Statements (contd.):
Don’t include code that does not do anything
This may sound obvious. However, this scenario is seen in some off-the-shelf applications.

For example, you may see code which is given below:

```
SELECT column_name FROM table_name  WHERE 1 = 0
```

When this query is run, no rows will be returned. It is just wasting SQL Server resources.

By default, some developers routinely include code, which is similar to the one given above, in their WHERE clauses when they make string comparisons.

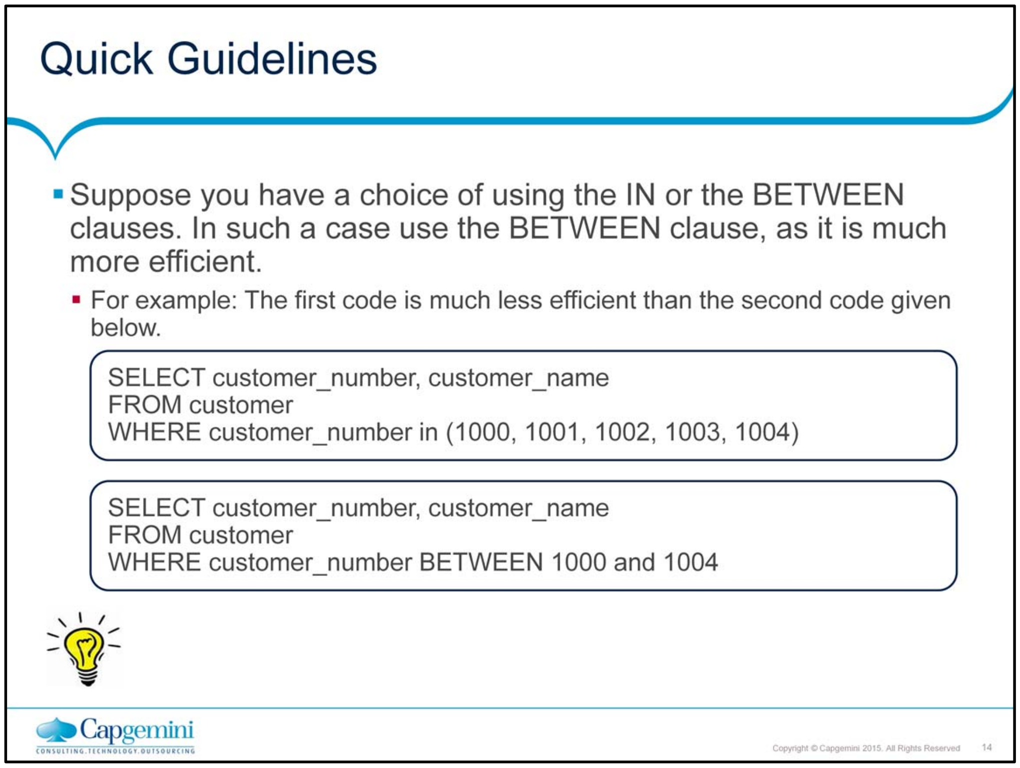
For example:

```
SELECT column_name FROM table_name
WHERE LOWER(column_name) = ‘name’
```

Any use of text functions in a WHERE clause decreases performance.
If your database has been configured to be case-sensitive, then using text functions in the WHERE clause does decrease performance. However, in such a case, use the technique described below, along with appropriate indexes on the column in question:

```
SELECT column_name FROM table_name
WHERE column_name = ‘NAME’ or column_name = ‘name’
```

This code will run much faster than the first example.



Quick Guidelines

- Suppose you have a choice of using the IN or the BETWEEN clauses. In such a case use the BETWEEN clause, as it is much more efficient.
  - For example: The first code is much less efficient than the second code given below.

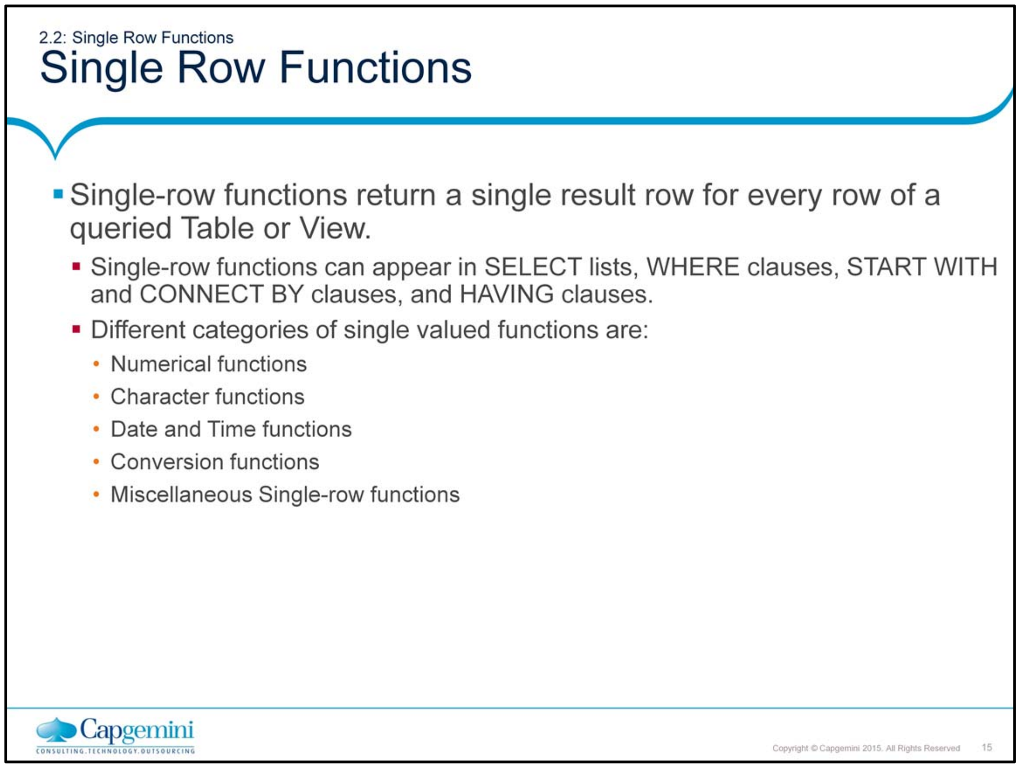
```
SELECT customer_number, customer_name
FROM customer
WHERE customer_number in (1000, 1001, 1002, 1003, 1004)
```

```
SELECT customer_number, customer_name
FROM customer
WHERE customer_number BETWEEN 1000 and 1004
```

Tips and Tricks in SELECT Statements (contd.):
Assuming there is a useful Index on customer_number, the Query Optimizer can locate a range of numbers much faster by using BETWEEN clause.

This is much faster than it can find a series of numbers by using the IN clause (which is really just another form of the OR clause).

Using Efficient Non-index WHERE clause sequencing:
Oracle evaluates un-indexed equations, linked by the AND verb in a bottom-up fashion. This means that the first clause (last in the AND list) is evaluated, and if it is found TRUE, then the second clause is tested.
Always try to position the most expensive clause first in the WHERE clause sequencing.
Oracle evaluates un-indexed equations, linked by the OR verb in a top-down fashion. This means that the first clause (first in the OR list) is evaluated, and if it is found FALSE, then the second clause is tested.
Always try to position the most expensive OR clause last in the WHERE clause sequencing.



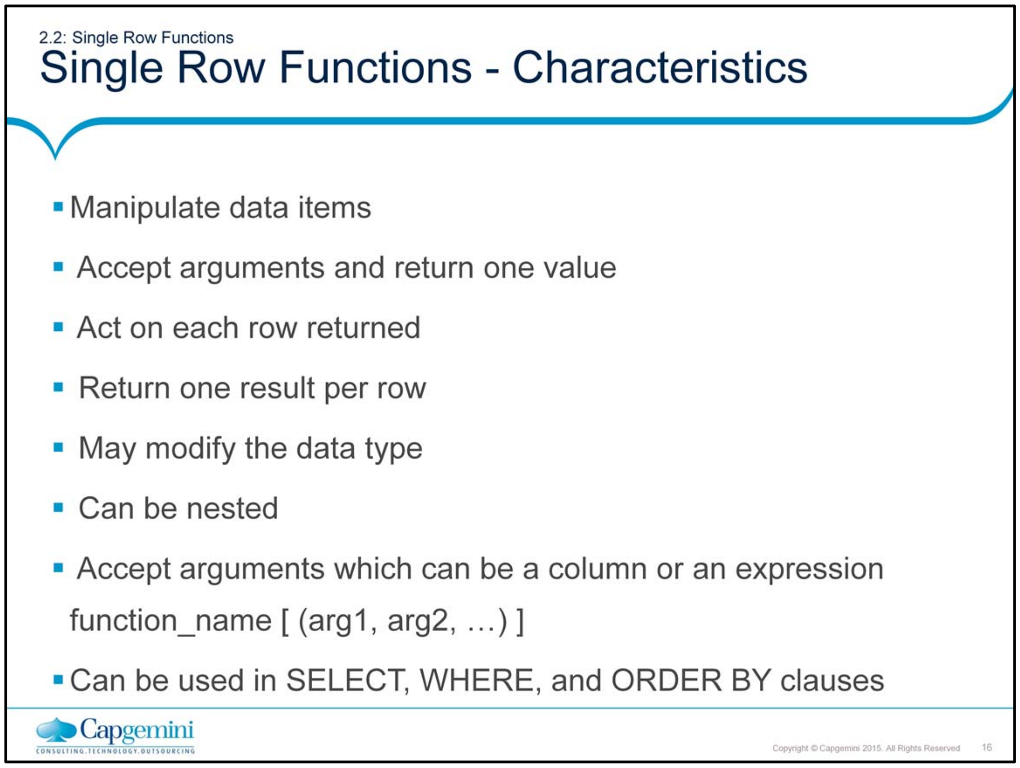
2.2: Single Row Functions

Single Row Functions

- Single-row functions return a single result row for every row of a queried Table or View.
  - Single-row functions can appear in SELECT lists, WHERE clauses, START WITH and CONNECT BY clauses, and HAVING clauses.
  - Different categories of single valued functions are:
    - Numerical functions
    - Character functions
    - Date and Time functions
    - Conversion functions
    - Miscellaneous Single-row functions

SQL Functions:
So far, we have seen “Aggregate functions”, which operate against a “collection of values”, however return a “single value”.
Now we shall see “scalar functions” which operate against a “single value”, and return a “single value” based on the input value.
The functions can be broadly classified into:

Numerical functions- Accept numeric input and return numeric values
For example: ABS, CEIL, TRUNC, ROUND, POWER, etc.
Character functions - Accept character input and can return both characters and number values
For example: CONCAT, LPAD, RPAD, TRIM, SUBSTR, etc.
Date and Time functions- Operate on values of the DATE data type
For example: SYSDATE, ADD_MONTHS, LAST_DAY, etc.
Conversion functions- Convert a value from one data type to another
For example: CAST, ASCIISTR, ROWIDTOCHAR, etc.
Miscellaneous Single-row functions
For example: BFILENAME, DECODE, NVL, NULLIF, USERENV, etc.



2.2: Single Row Functions

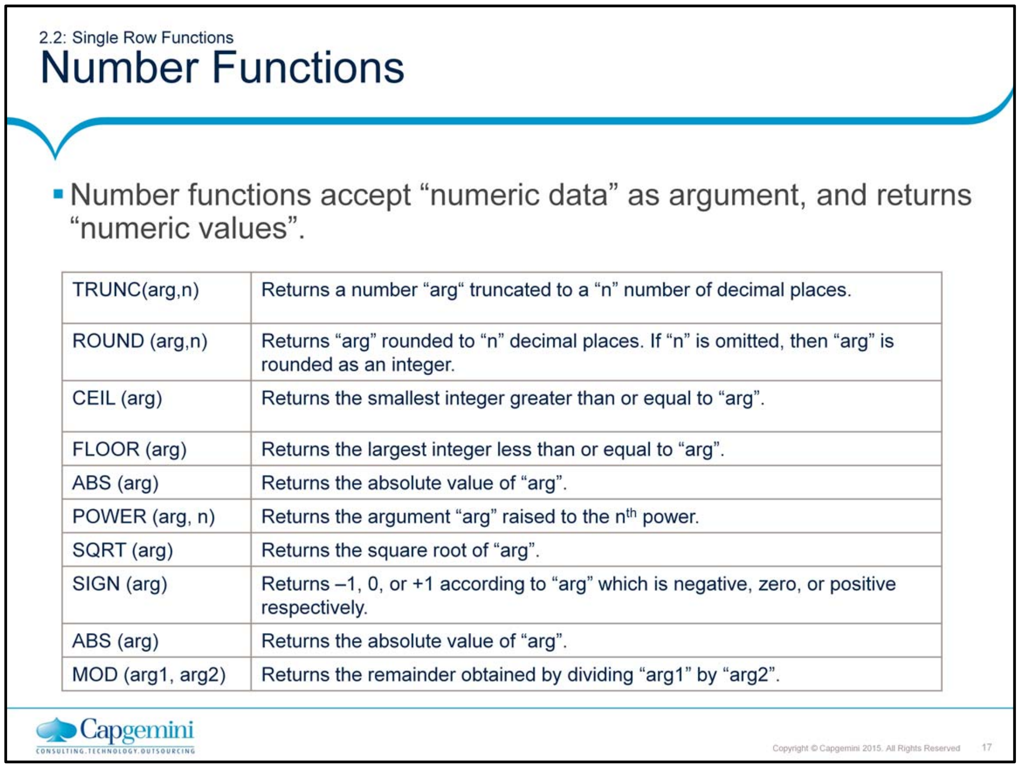
Single Row Functions - Characteristics

- Manipulate data items
- Accept arguments and return one value
- Act on each row returned
- Return one result per row
- May modify the data type
- Can be nested
- Accept arguments which can be a column or an expression

  function_name [(arg1, arg2, …)]
- Can be used in SELECT, WHERE, and ORDER BY clauses

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Let us now have a look at each of these function categories



2.2: Single Row Functions

Number Functions

- Number functions accept “numeric data” as argument, and returns “numeric values”.

TRUNC(arg,n)	Returns a number “arg“ truncated to a “n” number of decimal places.
ROUND (arg,n)	Returns “arg” rounded to “n” decimal places. If “n” is omitted, then “arg” is rounded as an integer.
CEIL (arg)	Returns the smallest integer greater than or equal to “arg”.
FLOOR (arg)	Returns the largest integer less than or equal to “arg”.
ABS (arg)	Returns the absolute value of “arg”.
POWER (arg, n)	Returns the argument “arg” raised to the nth power.
SQRT (arg)	Returns the square root of “arg”.
SIGN (arg)	Returns –1, 0, or +1 according to “arg” which is negative, zero, or positive respectively.
ABS (arg)	Returns the absolute value of “arg”.
MOD (arg1, arg2)	Returns the remainder obtained by dividing “arg1” by “arg2”.

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

Numeric Functions:
TRUNC(n,m)
The trunc function returns a number truncated to a certain number of decimal places. The syntax for the trunc function is:

```
trunc( number, [ decimal_places ] )
```

where:
number = the number to truncate.
decimal_places = the number of decimal places to truncate to. This value must be an integer. If this parameter is omitted, the trunc function will truncate the number to 0 decimal places.
For example:
trunc(125.815) would return 125
trunc(125.815, 1) would return 125.8
trunc(125.815, -1) would return 120
trunc(125.815, -2) would return 100
ROUND(n,m)
The round function returns a number rounded to a certain number of decimal places. The syntax for the round function is:

```
round( number, [ decimal_places ] )
```

Numeric Functions (contd.):
where:
number = the number to round.
decimal_places = the number of decimal places rounded to. This value must be an integer. If this parameter is omitted, the round function will round the number to 0 decimal places.
For example:
round(125.315) would return 125
round(125.315, 0) would return 125
round(125.315, 1) would return 125.3
round(125.315, 2) would return 125.32
round(-125.315, 2) would return -125.32
CEIL(n)
It returns smallest integer greater than or equal to n.

```
SELECT CEIL(15.7) "Ceiling" FROM DUAL;
```

```
Ceiling
16
```

FLOOR(n)
•It returns largest integer equal to or less than n.

```
SELECT FLOOR(15.7) "Floor" FROM DUAL;
```

```
Floor
15
```

ABS(n)
•It returns the absolute value of n.

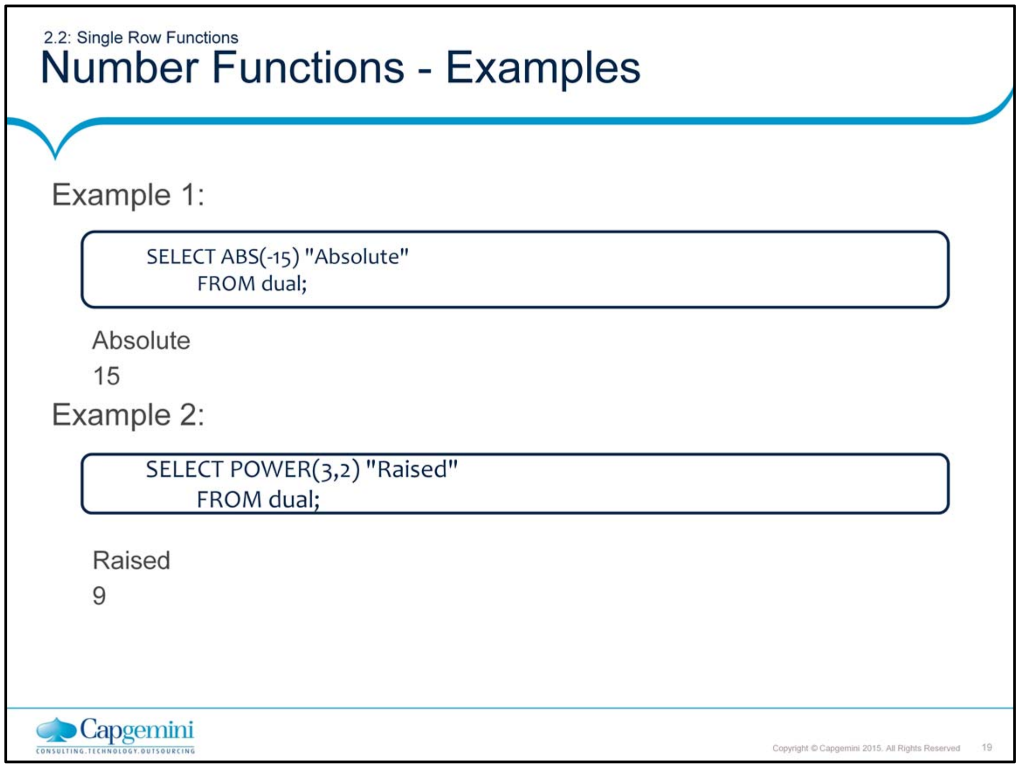
```
SELECT ABS(20) "Absolute" FROM DUAL;
```

```
Absolute
20
```

POWER function
•It returns m raised to nth power .It is of the form: Power(m,n)

```
SELECT POWER (3,3) “Raised” FROM DUAL;
```

```
Raised
27
```



2.2: Single Row Functions

Number Functions - Examples

Example 1:

```
SELECT ABS(-15) "Absolute"
       FROM dual;
```

```
Absolute
15
```

Example 2:

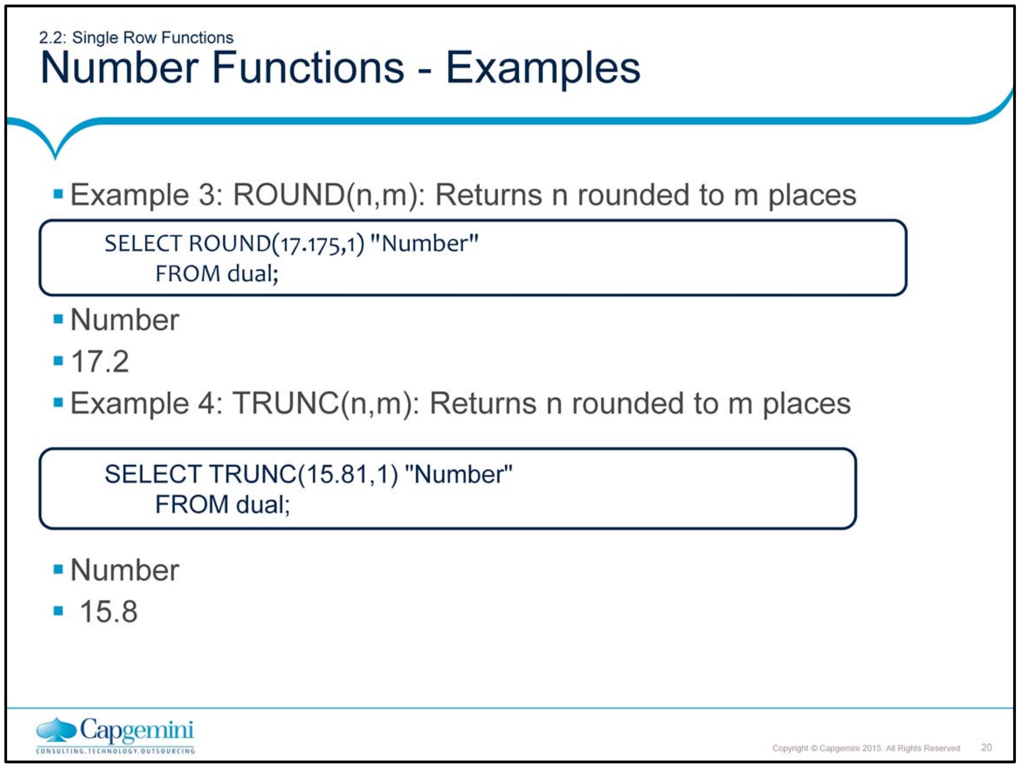
```
SELECT POWER(3,2) "Raised"
       FROM dual;
```

```
Raised
9
```

Examples of Number Functions:
DUAL table, which is shown in the slide, is a table owned by SYS.

SYS owns the “data dictionary”, and DUAL is part of the data dictionary.
DUAL is a small work-table, which consists of only one row and one column, and contains the value “x” in that column. Besides arithmetic calculations, it also supports “date retrieval” and it’s “formatting”.
Often a simple calculation needs to be done. A SELECT must have a table name in it’s FROM clause, else it fails.
To facilitate such calculations via a SELECT, the DUAL dummy table is provided.
The structure of the DUAL table can be viewed by using the SQL statement:

```
DESC DUAL;
```



2.2: Single Row Functions

Number Functions - Examples

- Example 3: ROUND(n,m): Returns n rounded to m places

```
SELECT ROUND(17.175,1) "Number"
      FROM dual;
```

- Number
- 17.2
- Example 4: TRUNC(n,m): Returns n rounded to m places

```
SELECT TRUNC(15.81,1) "Number"
      FROM dual;
```

- Number
- 15.8

Examples of Number (numeric) Functions (contd.):
Round(n,m):

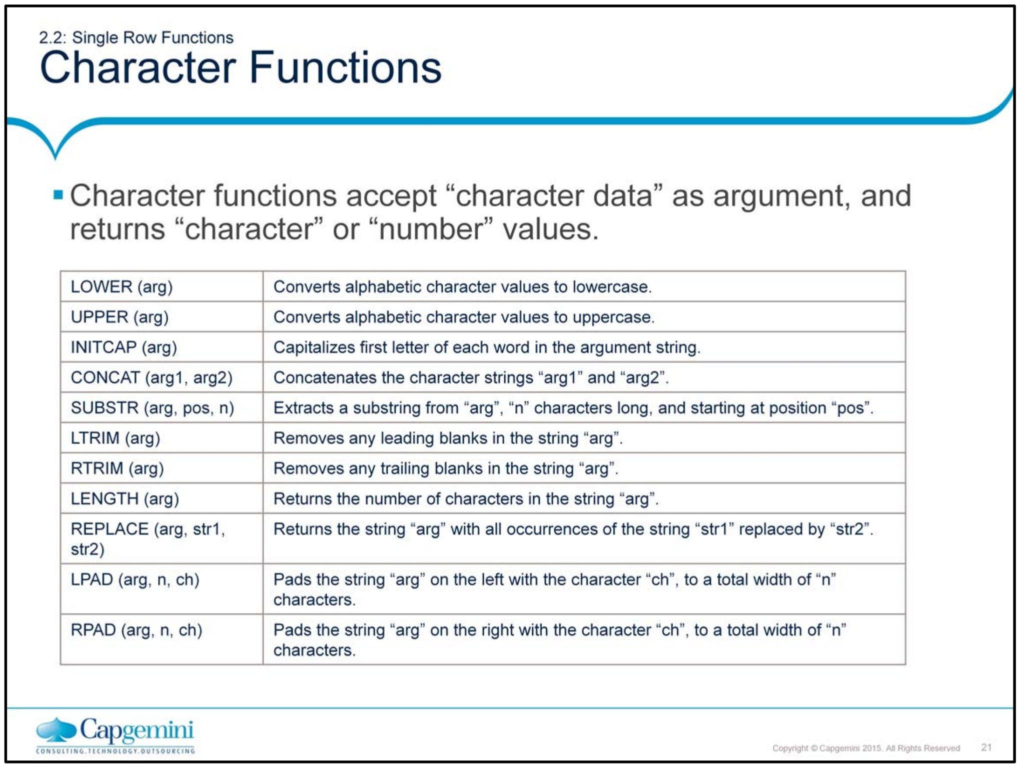
```
SELECT ROUND(17.175,-1) "Number"
      FROM dual;
```

O/P : 20

TRUNC(n,m):

```
SELECT TRUNC(15.81,-1) "Number“
      FROM dual;
```

O/P : 10



2.2: Single Row Functions

Character Functions

- Character functions accept “character data” as argument, and returns “character” or “number” values.

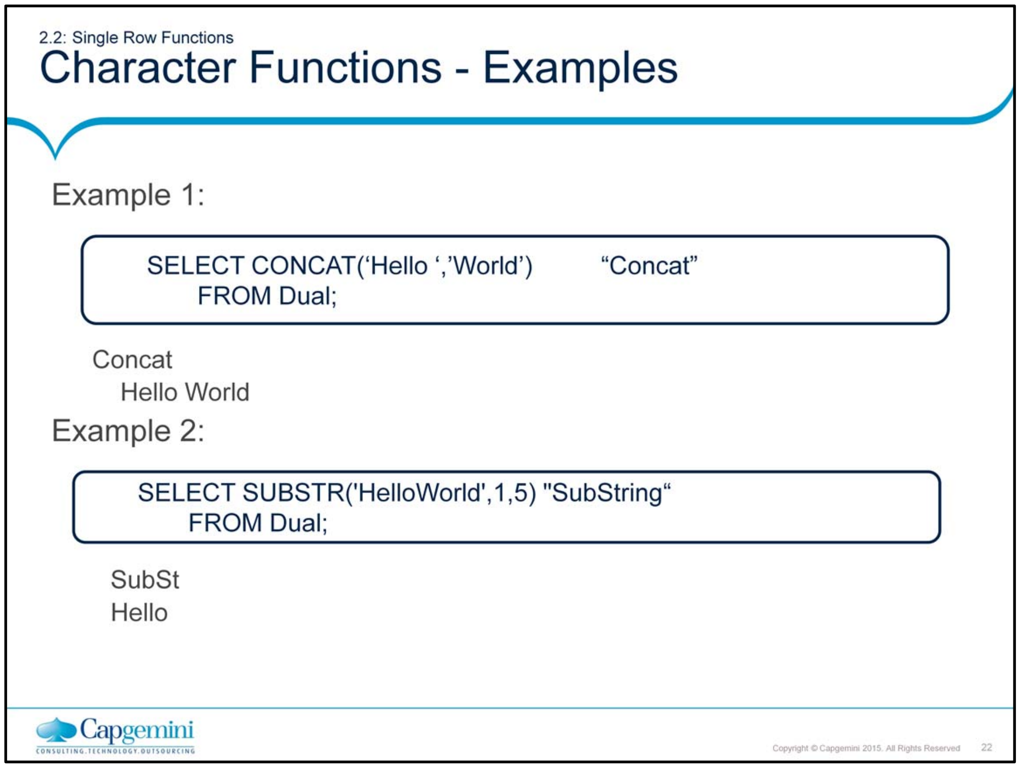
LOWER (arg)	Converts alphabetic character values to lowercase.
UPPER (arg)	Converts alphabetic character values to uppercase.
INITCAP (arg)	Capitalizes first letter of each word in the argument string.
CONCAT (arg1, arg2)	Concatenates the character strings “arg1” and “arg2”.
SUBSTR (arg, pos, n)	Extracts a substring from “arg”, “n” characters long, and starting at position “pos”.
LTRIM (arg)	Removes any leading blanks in the string “arg”.
RTRIM (arg)	Removes any trailing blanks in the string “arg”.
LENGTH (arg)	Returns the number of characters in the string “arg”.
REPLACE (arg, str1, str2)	Returns the string “arg” with all occurrences of the string “str1” replaced by “str2”.
LPAD (arg, n, ch)	Pads the string “arg” on the left with the character “ch”, to a total width of “n” characters.
RPAD (arg, n, ch)	Pads the string “arg” on the right with the character “ch”, to a total width of “n” characters.

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

Character Functions:
Example1:

```
SELECT Upper(‘Hello’), Lower(‘WORLD’)
FROM Dual;
```

Note: Functions can be nested to any depth. Evaluation starts with the “inner most functions”, and proceeds outwards.
For example: LENGTH(LTRIM(RTRIM(name)))



2.2: Single Row Functions

Character Functions - Examples

Example 1:

```
SELECT CONCAT('Hello ','World')        "Concat"
      FROM Dual;
```

```
Concat
  Hello World
```

Example 2:

```
SELECT SUBSTR('HelloWorld',1,5) "SubString"
      FROM Dual;
```

```
SubSt
Hello
```

Examples of Character Functions:
REPLACE function returns char with every occurrence of search_string replaced with replacement_string.

If replacement_string is omitted or NULL, then all occurrences of search_string are removed.
If search_string is NULL, then char is returned.

```
SELECT REPLACE('JACK and JUE','J','BL') "Changes"
          FROM DUAL;
```

```
Changes
BLACK and BLUE
```

CONCAT function returns "char1" concatenated with "char2". Both "char1" and "char2" can be any of the datatypes CHAR, VARCHAR2, NCHAR, NVARCHAR2, CLOB, or NCLOB. The string returned is in the same character set as char1. It's datatype depends on the datatypes of the arguments.

In concatenations of two different datatypes, the Oracle Database returns the datatype that results in a "lossless conversion". Therefore:

If one of the arguments is a LOB, then the returned value is a LOB.
If one of the arguments is a national datatype, then the returned value is a national datatype.

String Functions:

- The CONCAT function is equivalent to the “concatenation operator (||)”. The function is useful when there are spaces in the values to be concatenated. The “concatenation operator” does not permit spaces.

UPPER(string)

- This function converts all characters in string to uppercase.

```
SELECT UPPER(staff_name),staff_code
        FROM staff_master;
```

LOWER(string)

- This function converts all characters in string to lowercase.

```
SELECT LOWER(student_name)
        FROM student_master;
```

INITCAP(string):

- This function converts the first character of each word in string to uppercase and the rest of the characters to lowercase.

```
SELECT INITCAP(staff_name)
        FROM staff_master;
```

LPAD(string1,n,string2)

- This function adds string2 before string1 as many times as required to make the string1 length equal to “n” chars.
- To right align the names of staff_members:

```
SELECT LPad(staff_name,30)
        FROM staff_master;
```

LTRIM(string,CHAR set)

- This function removes chars from beginning of string as long as the character matches one of the chars in the CHAR set.

```
SELECT student_name,LTRIM(student_name,'MALICE') FROM
student_master;
```

RPAD(string1,n,string2)

- This function is similar to LPAD. It adds chars to the right end.

RTRIM(string,CHAR set)

- This function is similar to LTRIM. It removes chars from the right end.

SUBSTR(CHAR,m,n)

- This function returns the string from the mth character of the string to the nth character

<u>String Functions (contd.)</u>:

- **LENGTH function**
  It returns the length of char in characters
  It is of the form:
  Length (string)

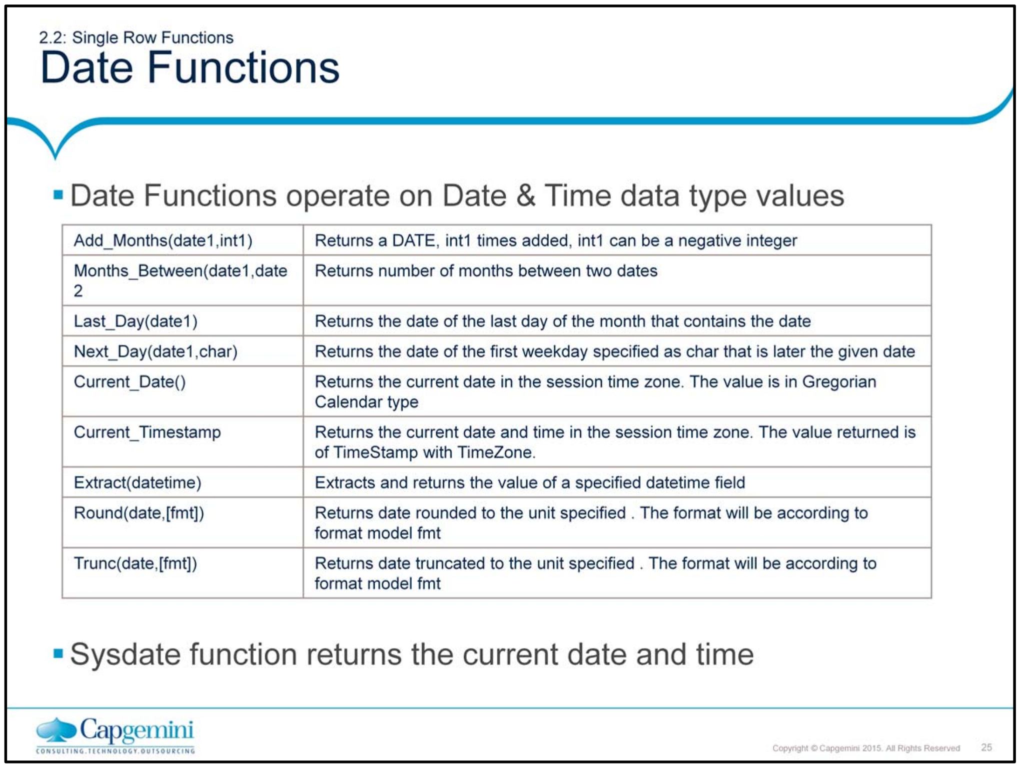
```
Select length ('candide') " length in characters"
       FROM dual;
```

INSTR(string, pattern[, start [,occurrence]])

- It returns the location of a character IN a STRING.

```
SELECT INSTR ('CORPORATEFLOOR','R',3,2) "Instring"
       FROM DUAL
```

```
Instring
6
```



2.2: Single Row Functions

Date Functions

- Date Functions operate on Date & Time data type values

Add_Months(date1,int1)	Returns a DATE, int1 times added, int1 can be a negative integer
Months_Between(date1,date2	Returns number of months between two dates
Last_Day(date1)	Returns the date of the last day of the month that contains the date
Next_Day(date1,char)	Returns the date of the first weekday specified as char that is later the given date
Current_Date()	Returns the current date in the session time zone. The value is in Gregorian Calendar type
Current_Timestamp	Returns the current date and time in the session time zone. The value returned is of TimeStamp with TimeZone.
Extract(datetime)	Extracts and returns the value of a specified datetime field
Round(date,[fmt])	Returns date rounded to the unit specified . The format will be according to format model fmt
Trunc(date,[fmt])	Returns date truncated to the unit specified . The format will be according to format model fmt

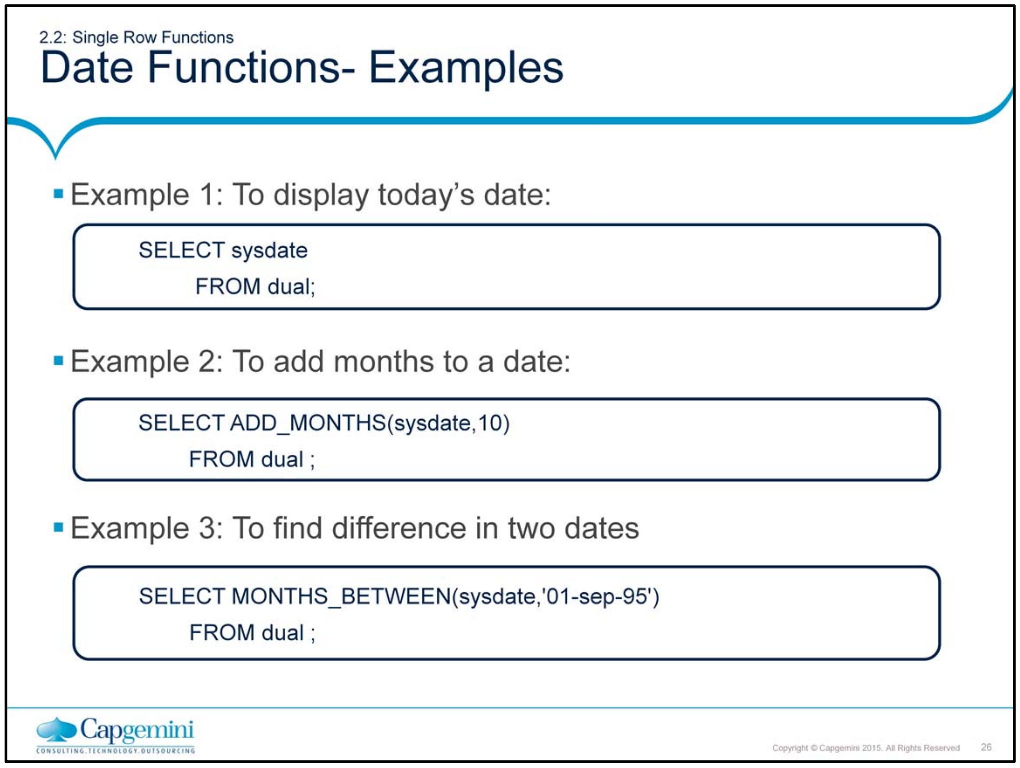
- Sysdate function returns the current date and time

25

DATE Functions
Datetime functions operate on date (DATE), timestamp, and interval values. Some of the datetime functions were designed for the Oracle DATE datatype (ADD_MONTHS, CURRENT_DATE, LAST_DAY, and NEXT_DAY).

    If you provide a timestamp value as their argument, Oracle Database internally converts the input type to a DATE value and returns a DATE value.
    The exceptions are:
        the MONTHS_BETWEEN function, which returns a number, and the ROUND and TRUNC functions, which do not accept timestamp or interval values at all.

The remaining datetime functions are designed to accept any of the three types of data (date, timestamp, and interval), and to return a value of one of these types.



2.2: Single Row Functions

Date Functions- Examples

- Example 1: To display today's date:

```
SELECT sysdate
        FROM dual;
```

- Example 2: To add months to a date:

```
SELECT ADD_MONTHS(sysdate,10)
       FROM dual ;
```

- Example 3: To find difference in two dates

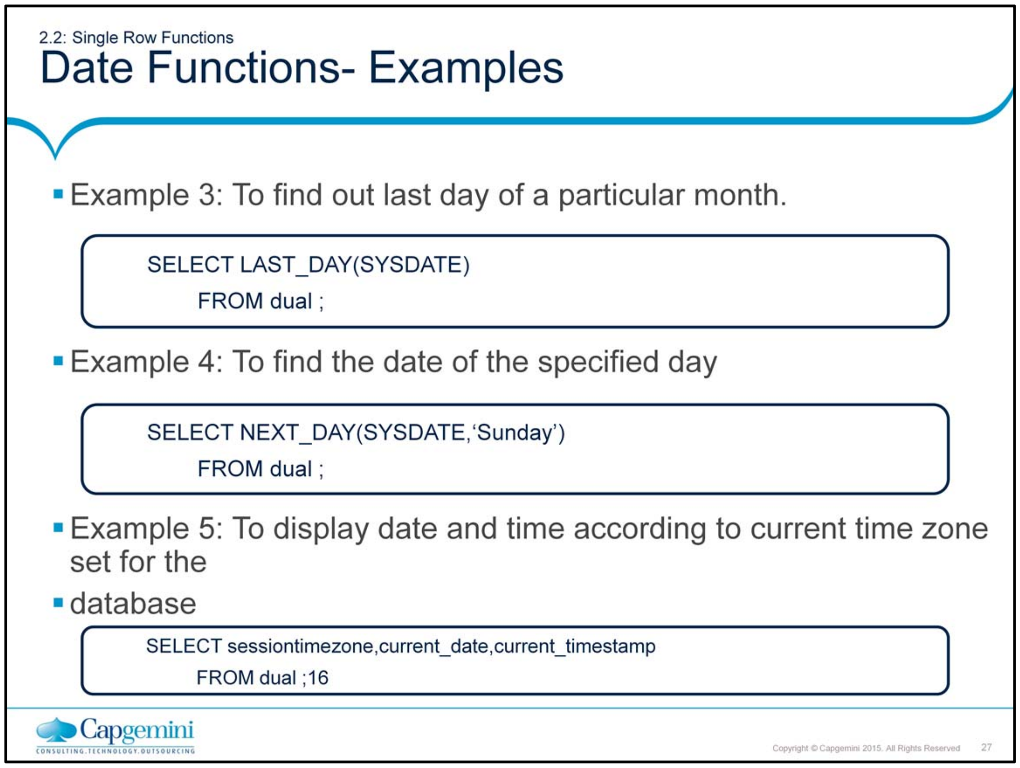
```
SELECT MONTHS_BETWEEN(sysdate,'01-sep-95')
       FROM dual ;
```

Date Functions:
ADD_MONTHS(DATE1,int1) returns the date as addition of "date" and "integer" months. The first argument can be a datetime value or any value that can be implicitly converted to DATE. The second argument can be an integer or any value that can be implicitly converted to an integer. The return type is always DATE.

```
SELECT book_code,ADD_MONTHS(book_issue_date,1)
FROM book_transaction;
```

MONTHS_BETWEEN (DATE1,DATE2)This function returns number of months between the two DATEs. The result is positive if date1 is later than date2 and result is negative if date2 is later than date2.

```
SELECT staff_code,
MONTHS_BETWEEN(TRUNC(sysdate),hiredate)
FROM staff_master;
```



2.2: Single Row Functions

Date Functions- Examples

- Example 3: To find out last day of a particular month.

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

- Example 4: To find the date of the specified day

```
SELECT NEXT_DAY(SYSDATE,'Sunday')
      FROM dual ;
```

- Example 5: To display date and time according to current time zone set for the
- database

```
SELECT sessiontimezone,current_date,current_timestamp
      FROM dual ;16
```

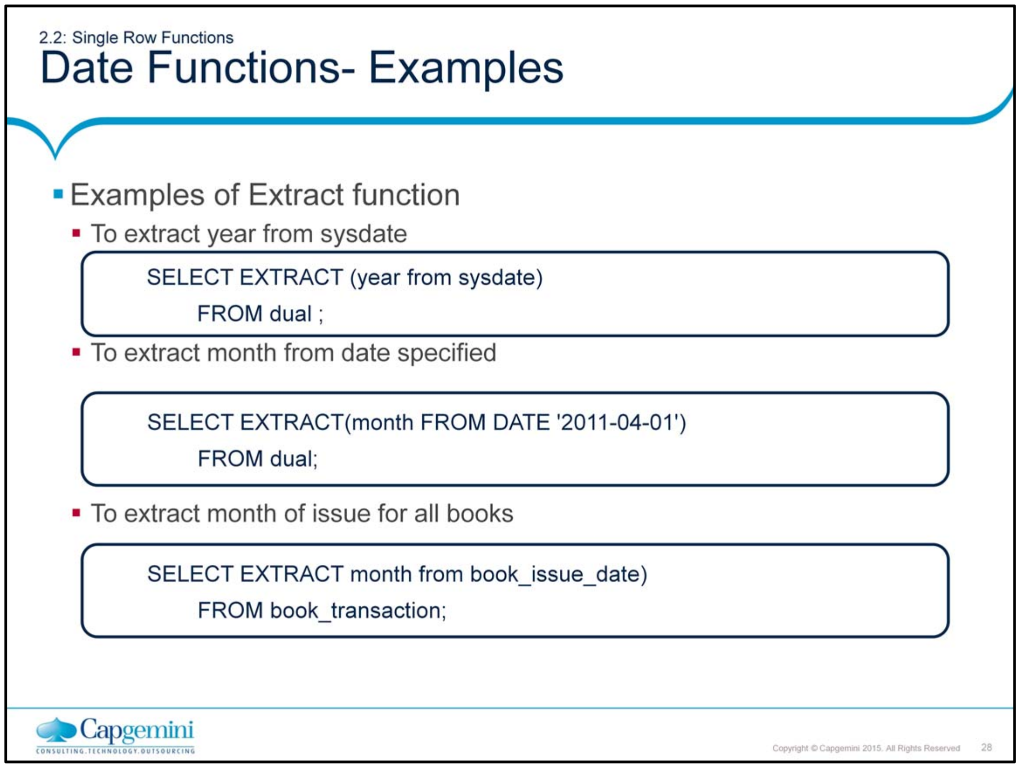
Date Functions (contd.):
LASTDAY(date1) returns the date of the last day of the month that contains the date. The return type is always DATE.
To display the date of the last day in the month the books were issued

```
SELECT book_issue_date, LAST_DAY(book_issue_date)
        FROM book_transaction;
```

NEXT_DAY(date1, char) Returns the date of the first weekday specified as char that is later the given date
To display the date on coming Friday in the week that books were issued

```
SELECT book_issue_date,
NEXT_DAY(book_issue_date,'Friday')
        FROM book_transaction
```

CURRENT_DATE & CURRENT_TIMESTAMP return current date and time respectively base on the timezone set for the database.
The query given on the slide above shows the date and time based on the offset time base on GMT which is according to the timezone set for the database



2.2: Single Row Functions

Date Functions- Examples

- Examples of Extract function
  - To extract year from sysdate

```
SELECT EXTRACT (year from sysdate)
       FROM dual ;
```

  - To extract month from date specified

```
SELECT EXTRACT(month FROM DATE '2011-04-01')
       FROM dual;
```

  - To extract month of issue for all books

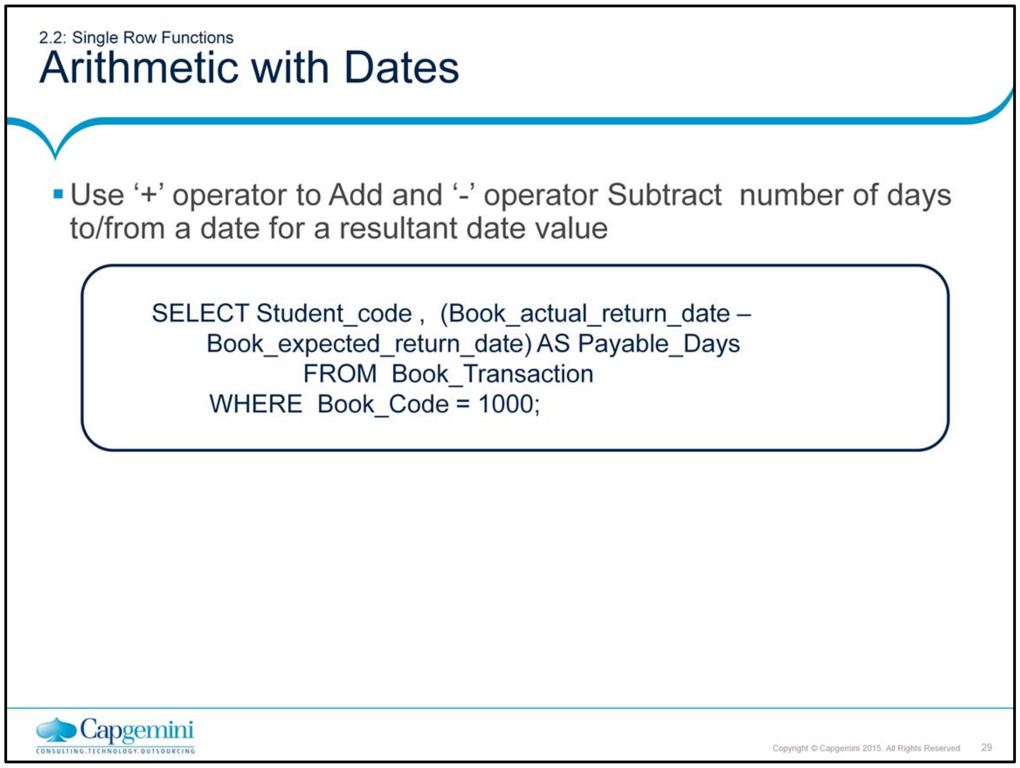
```
SELECT EXTRACT month from book_issue_date)
       FROM book_transaction;
```

Date Functions (contd.):
EXTRACT(datetime) extracts the value of a specified datetime field. This function is useful for manipulating datetime field values. For example you can extract only year, month or day from a given date value.
To display year of birth for all students

```
SELECT EXTRACT(year from student_dob)
        FROM student_master
```

TRUNC (DATE1)
This function truncates the time part from the DATE. This is required when we do DATE calculations.

```
SELECT staff_name, TRUNC(sysdate) - TRUNC(HIREDATE)
        FROM staff_master;
```



2.2: Single Row Functions

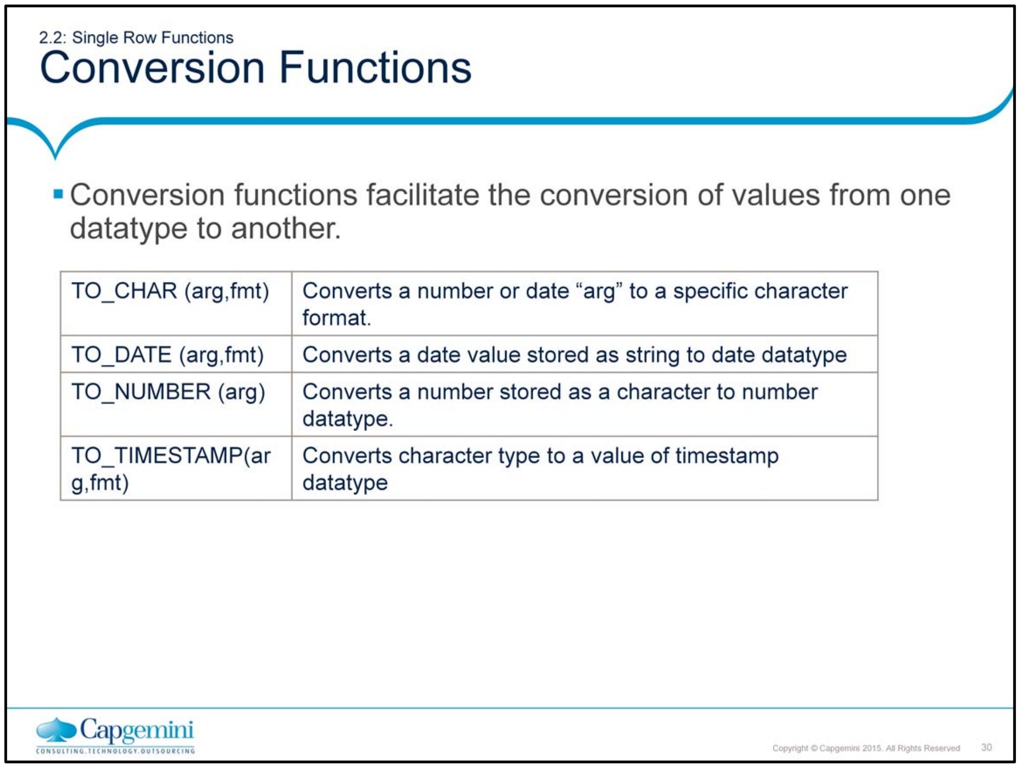
Arithmetic with Dates

- Use ‘+’ operator to Add and ‘-’ operator Subtract number of days to/from a date for a resultant date value

```
SELECT Student_code ,  (Book_actual_return_date –
      Book_expected_return_date) AS Payable_Days
                 FROM  Book_Transaction
      WHERE  Book_Code = 1000;
```

Given below are formats of Date Functions

Format	**Meaning**
YYYY	Four digit year
YY	Last two digits of the year
MM	Month (01-12 where JAN = 01 …)
MONTH	Name of the month stored as a length of nine characters.
MON	Name of the month in three letter format.
DD	Day of the month (01-31).
D	Day of the week.
DAY	Name of the day stored as a length of nine characters.
DY	Name of the day in three letter format.
FM	This prefix can be added to suppress blank padding.
HH	Hour of the day.
HH24	Hour in the 24 hour format.
MI	Minutes of the hour.
SS	Seconds of the minute.
TH	The suffix used with the day.



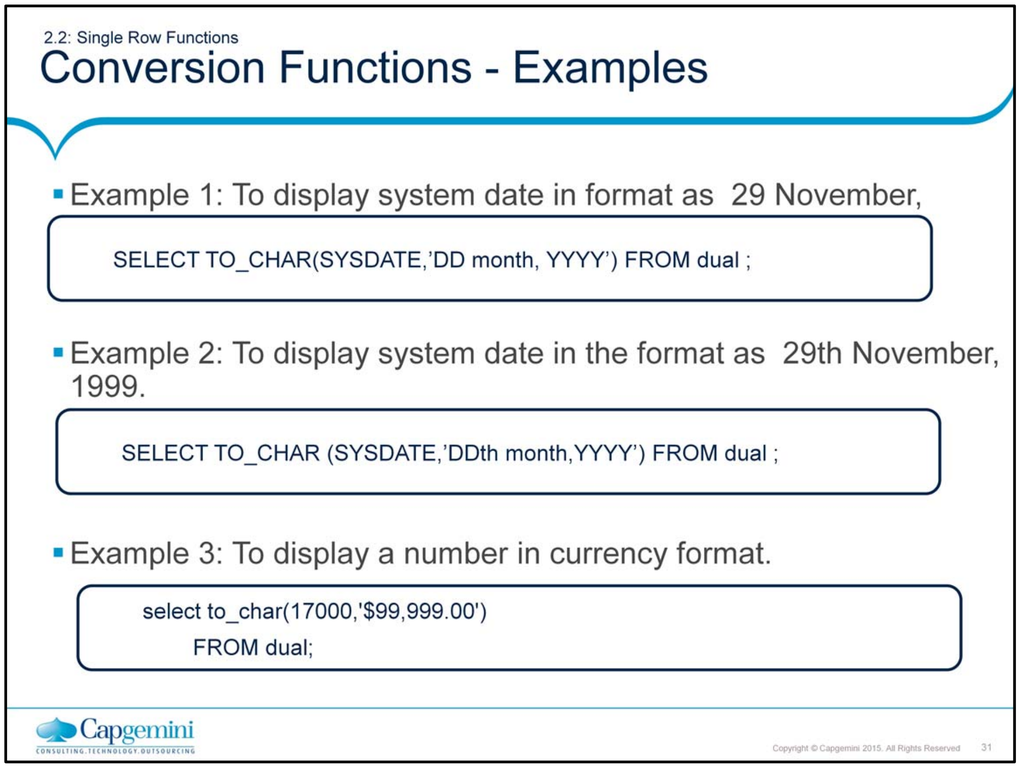
2.2: Single Row Functions

Conversion Functions

- Conversion functions facilitate the conversion of values from one datatype to another.

TO_CHAR (arg,fmt)	Converts a number or date “arg” to a specific character format.
TO_DATE (arg,fmt)	Converts a date value stored as string to date datatype
TO_NUMBER (arg)	Converts a number stored as a character to number datatype.
TO_TIMESTAMP(arg,fmt)	Converts character type to a value of timestamp datatype

Conversion Functions
Conversion functions convert values of one datatype to another. Usually the conversion function accepts two arguments wherein first is the input type and second is the output type.
Oracle takes care of implicit datatype conversion. Explicit datatype conversions are done using the conversion functions.
Although Oracle does provide implicit type conversion to ensure reliability of SQL statements you should use conversion functions



2.2: Single Row Functions

Conversion Functions - Examples

- Example 1: To display system date in format as 29 November,

```
SELECT TO_CHAR(SYSDATE,'DD month, YYYY') FROM dual ;
```

- Example 2: To display system date in the format as 29th November, 1999.

```
SELECT TO_CHAR (SYSDATE,'DDth month,YYYY') FROM dual ;
```

- Example 3: To display a number in currency format.

```
select to_char(17000,'$99,999.00')
    FROM dual;
```

TO_CHAR(DATE1,format)
This function converts the DATE given to the format specified.

```
SELECT TO_CHAR(TRUNC(sysdate), 'ddth fmMonth yy')
'DATE'
FROM dual;
```

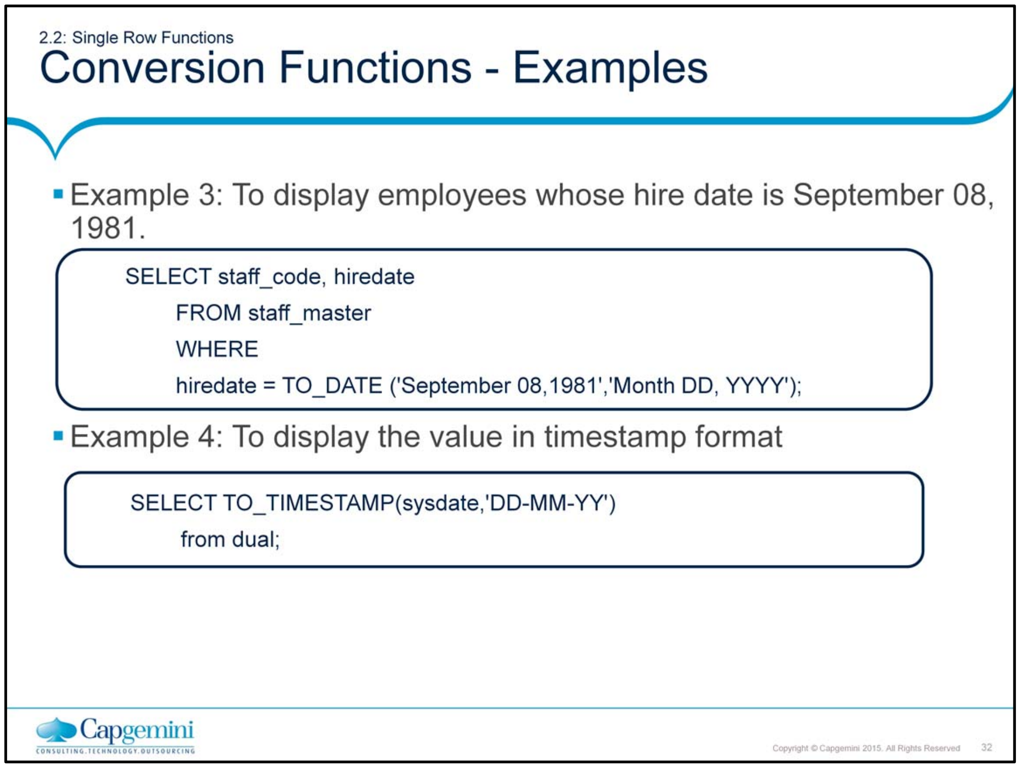
```
DATE
01st January 95
```

```
SELECT TO_CHAR(TRUNC(sysdate),'fmMonth') "DATE"
        FROM dual;
```

```
DATE
July
```

Example: To display the quarter which has the specified date.

```
SELECT to_char ( sysdate , 'Q')
        FROM  dual ;
```



2.2: Single Row Functions

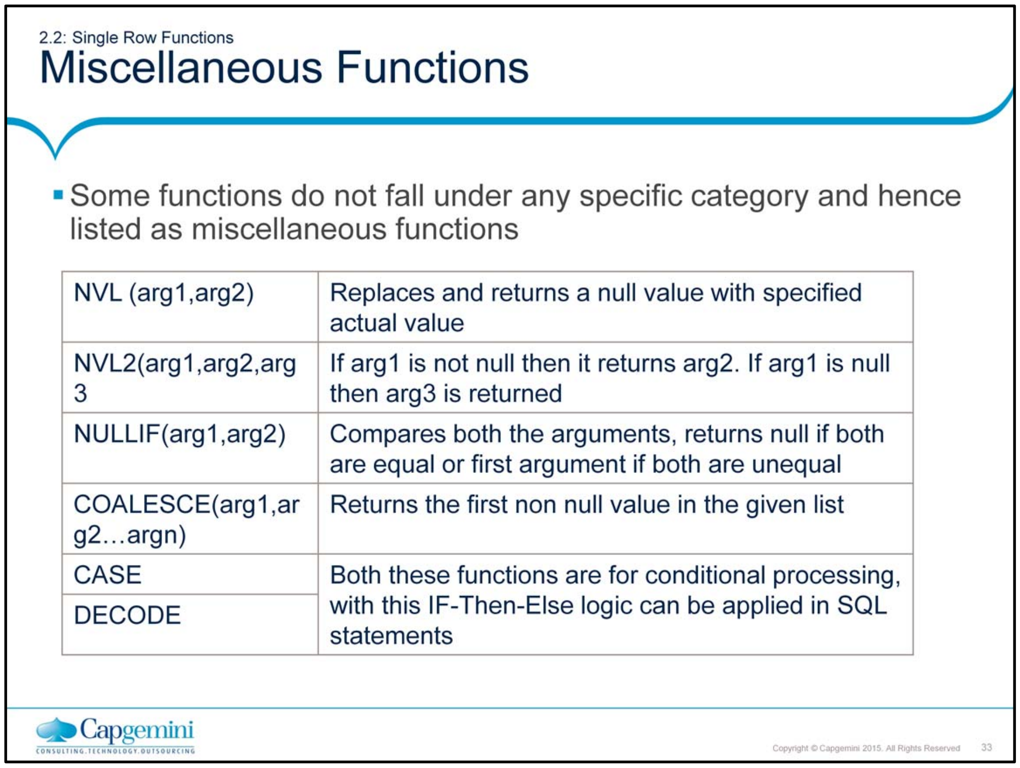
Conversion Functions - Examples

- Example 3: To display employees whose hire date is September 08, 1981.

```
SELECT staff_code, hiredate
    FROM staff_master
    WHERE
    hiredate = TO_DATE ('September 08,1981','Month DD, YYYY');
```

- Example 4: To display the value in timestamp format

```
SELECT TO_TIMESTAMP(sysdate,'DD-MM-YY')
    from dual;
```



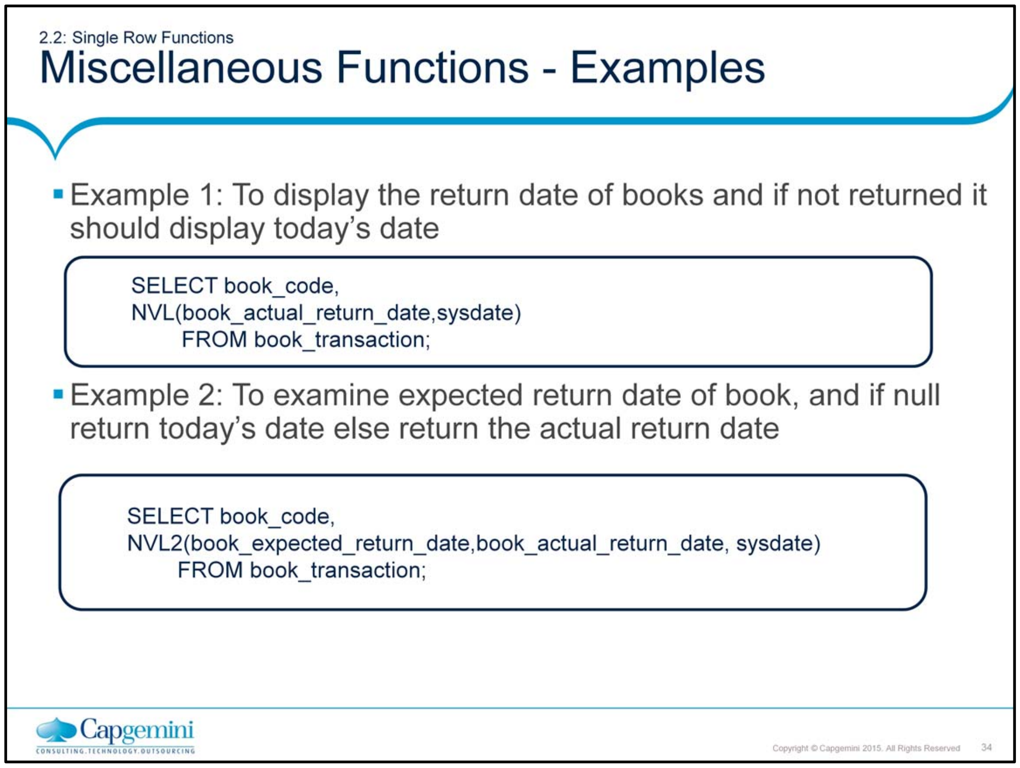
2.2: Single Row Functions

Miscellaneous Functions

- Some functions do not fall under any specific category and hence listed as miscellaneous functions

NVL (arg1,arg2)	Replaces and returns a null value with specified actual value
NVL2(arg1,arg2,arg3	If arg1 is not null then it returns arg2. If arg1 is null then arg3 is returned
NULLIF(arg1,arg2)	Compares both the arguments, returns null if both are equal or first argument if both are unequal
COALESCE(arg1,arg2…argn)	Returns the first non null value in the given list
CASE	Both these functions are for conditional processing, with this IF-Then-Else logic can be applied in SQL statements
DECODE	

Miscellaneous Functions
These functions are sometimes also called as General Functions. These functions work with any datatype values



2.2: Single Row Functions

Miscellaneous Functions - Examples

- Example 1: To display the return date of books and if not returned it should display today's date

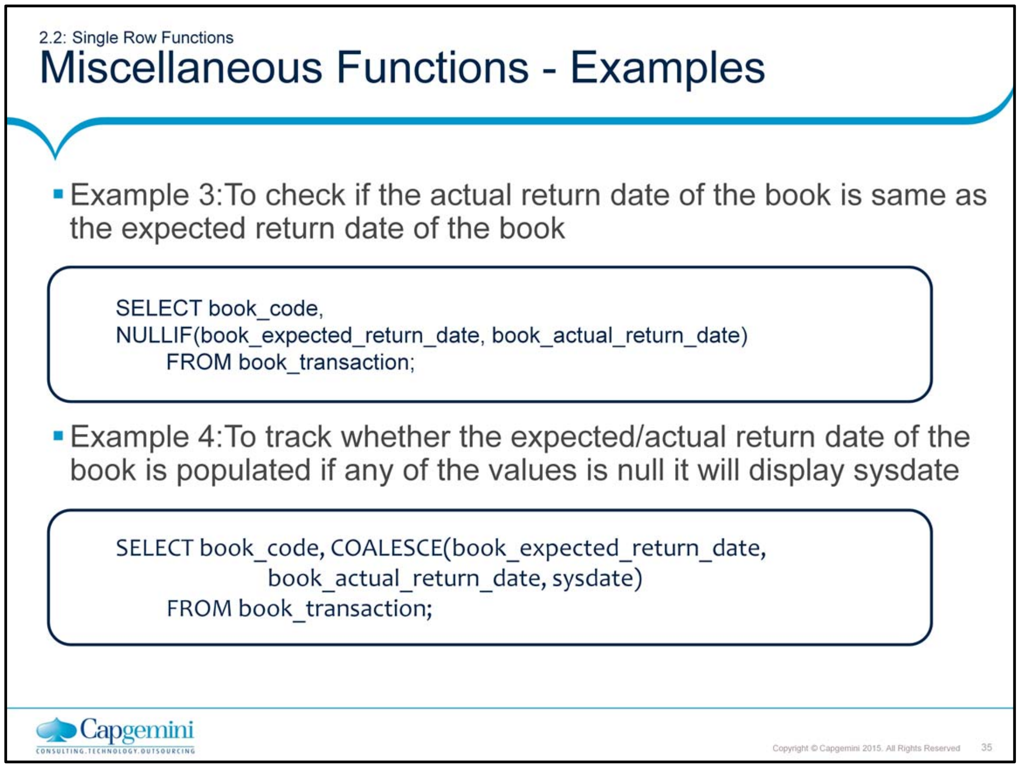
```
SELECT book_code,
NVL(book_actual_return_date,sysdate)
       FROM book_transaction;
```

- Example 2: To examine expected return date of book, and if null return today's date else return the actual return date

```
SELECT book_code,
NVL2(book_expected_return_date,book_actual_return_date, sysdate)
       FROM book_transaction;
```

Miscellaneous Single-row Functions:
NVL ()
Many times there are records holding NULL values in a table. When an output of such a table is displayed, it is difficult to understand the reason of NULL or BLANK values shown in the output.
The only way to overcome this problem is to replace NULL values with some other meaningful value while computing the records. This can be done using the NVL function.
NVL2()
The NVL2 function can be thought of as an extension to NVL function. But this function examines the first value. If the value is not null then returns the second value and if null then returns the third value.

Examples for both functions are shown on the slide



2.2: Single Row Functions

Miscellaneous Functions - Examples

- Example 3:To check if the actual return date of the book is same as the expected return date of the book

```
SELECT book_code,
NULLIF(book_expected_return_date, book_actual_return_date)
        FROM book_transaction;
```

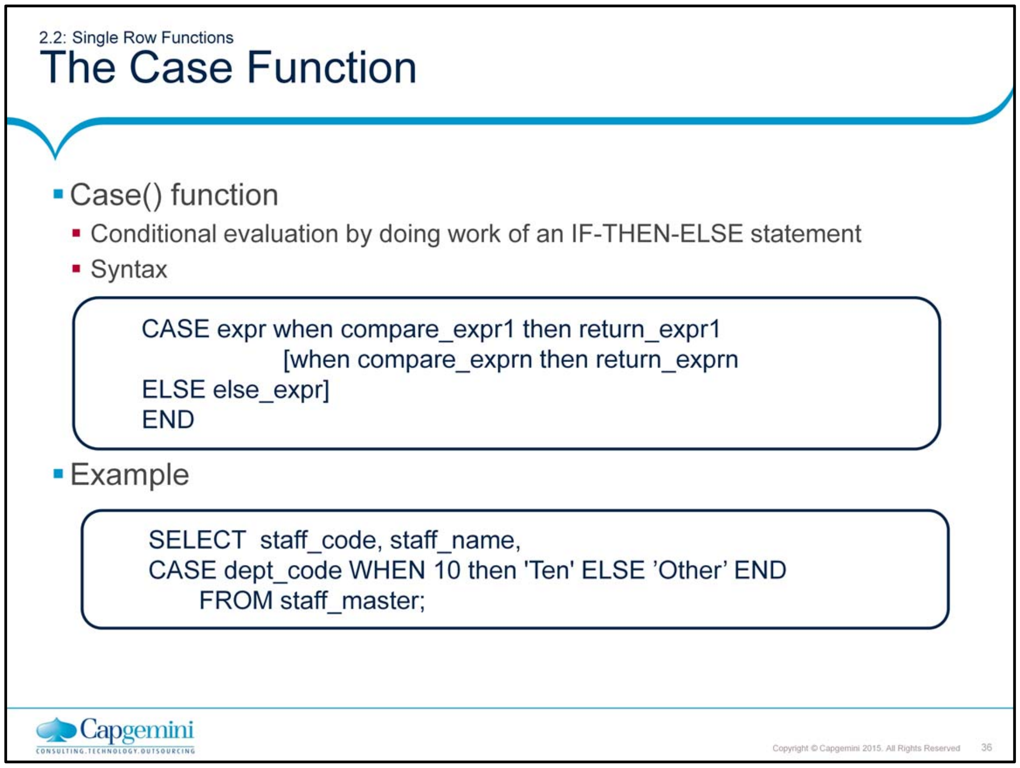
- Example 4:To track whether the expected/actual return date of the book is populated if any of the values is null it will display sysdate

```
SELECT book_code, COALESCE(book_expected_return_date,
                  book_actual_return_date, sysdate)
        FROM book_transaction;
```

NullIF ()
This function compares the arguments provided. If they are equal, the function returns null and if not then it returns the first argument.
You cannot specify the literal NULL as the first argument.

COALESE()
The Coalesce function returns the first non null value is the given argument list. The benefit of using this function versus NVL() us that it can take multiple alternate values.

Examples for both the functions are shown on the slide



2.2: Single Row Functions

The Case Function

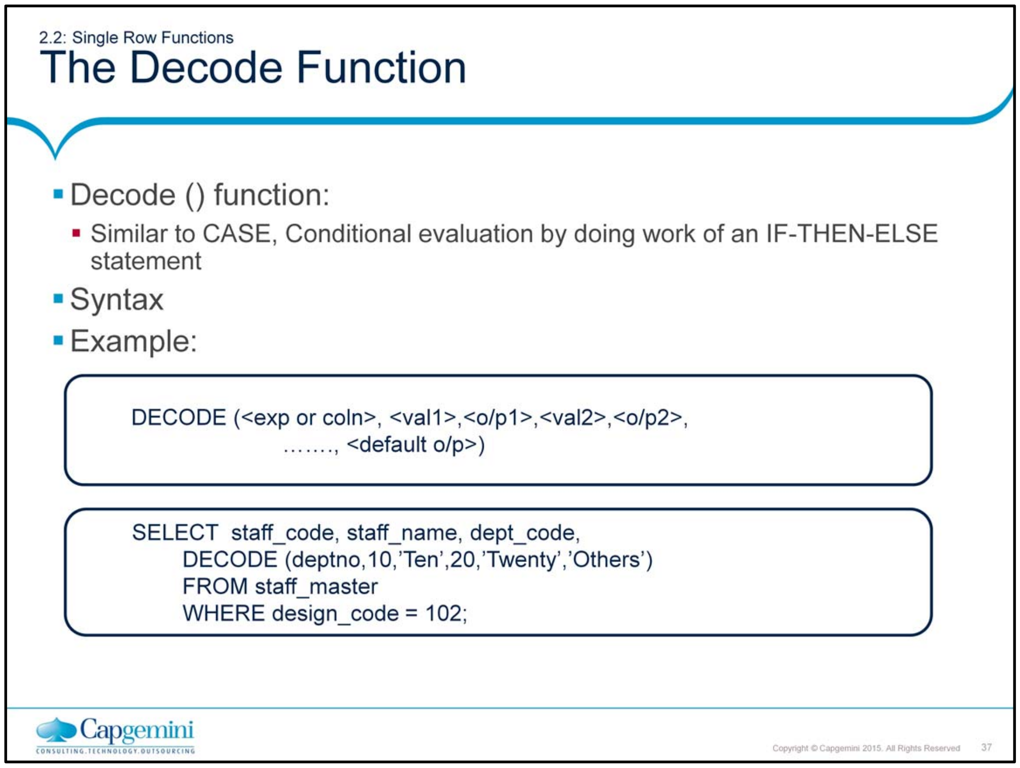
- Case() function
  - Conditional evaluation by doing work of an IF-THEN-ELSE statement
  - Syntax

```
CASE expr when compare_expr1 then return_expr1
              [when compare_exprn then return_exprn
ELSE else_expr]
END
```

- Example

```
SELECT  staff_code, staff_name,
CASE dept_code WHEN 10 then 'Ten' ELSE ’Other’ END
      FROM staff_master;
```

CASE()
In a simple case expression, Oracle searches for the first WHEN…THEN pair for which expr is equal to comparison expr and returns return_expr.
The expressions used should be of same datatype.
CASE is capable of more logical comparisons like <> etc..
Also CASE can work with predicates and subqueries



2.2: Single Row Functions

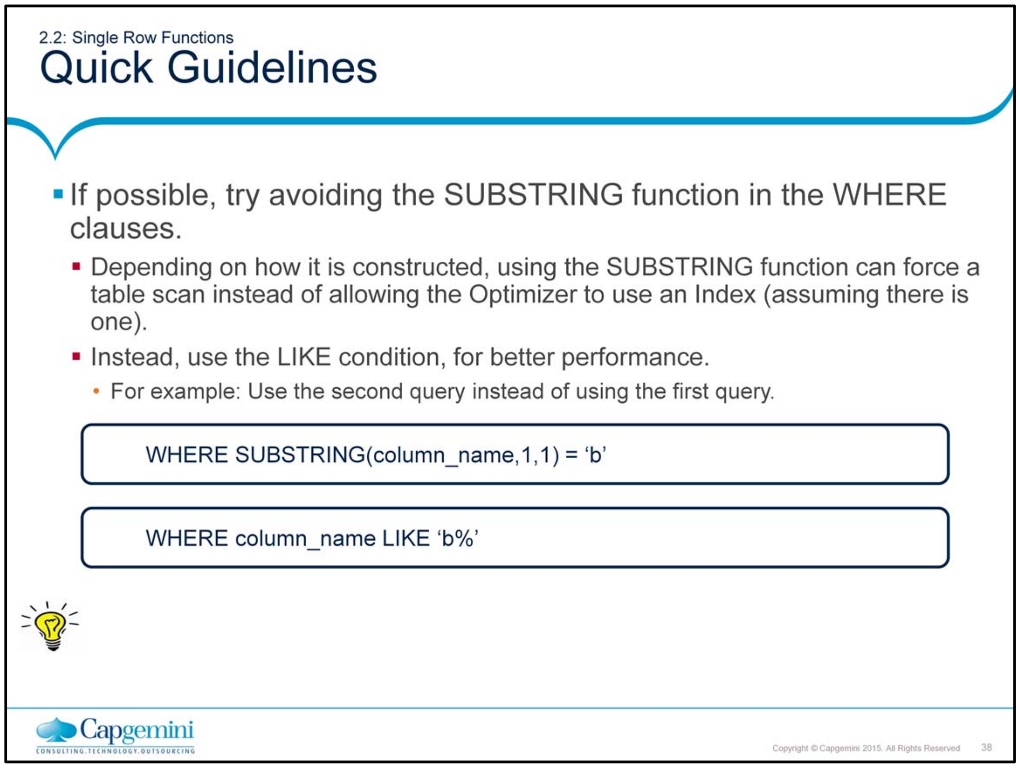
The Decode Function

- Decode () function:
  - Similar to CASE, Conditional evaluation by doing work of an IF-THEN-ELSE statement
- Syntax
- Example:

```
DECODE (<exp or coln>, <val1>,<o/p1>,<val2>,<o/p2>,
                ……., <default o/p>)
```

```
SELECT  staff_code, staff_name, dept_code,
     DECODE (deptno,10,'Ten',20,'Twenty','Others')
     FROM staff_master
     WHERE design_code = 102;
```

DECODE () function
This function decodes the expression after comparing it to each search value. If the expression is the same as search, result is returned. If the default value is omitted, a null value is returned where a search value does not match any of the result values.
As compared to CASE, DECODE can do an equality check only. The expressions in DECODE can work only on scalar values.
DECODE can work as a function inside SQL only but CASE can be more efficient substitute in PL/SQL blocks.



2.2: Single Row Functions

Quick Guidelines

- If possible, try avoiding the SUBSTRING function in the WHERE clauses.
  - Depending on how it is constructed, using the SUBSTRING function can force a table scan instead of allowing the Optimizer to use an Index (assuming there is one).
  - Instead, use the LIKE condition, for better performance.
    - For example: Use the second query instead of using the first query.

```
WHERE SUBSTRING(column_name,1,1) = ‘b’
```

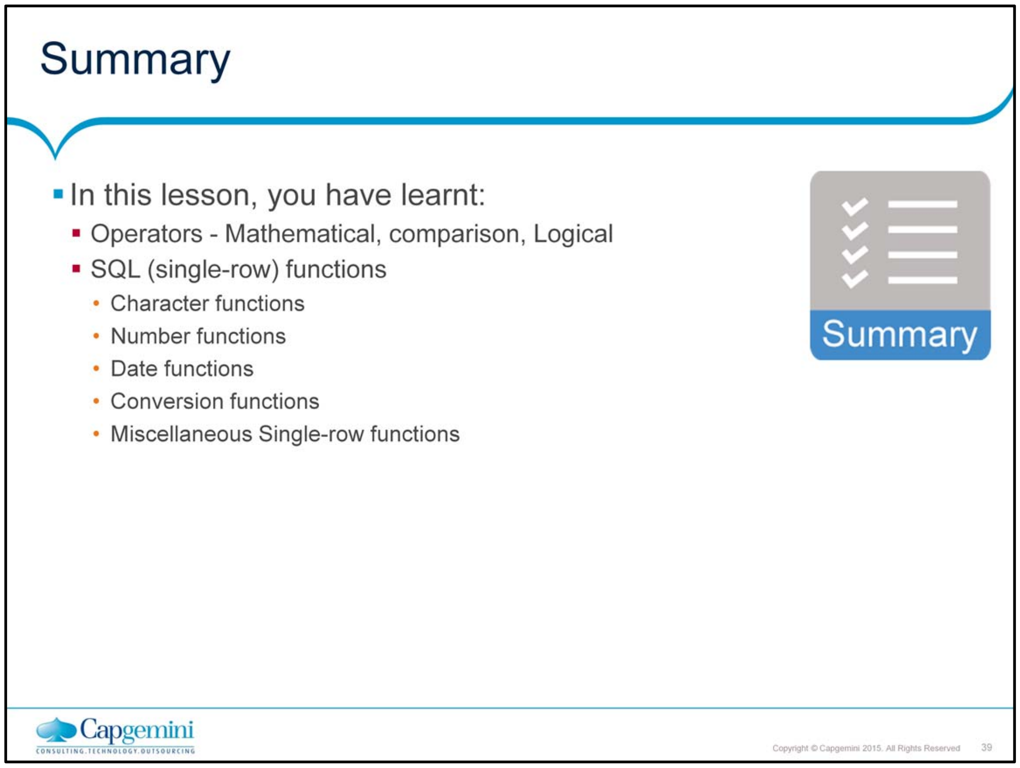
```
WHERE column_name LIKE ‘b%’
```

38

Tips and Tricks:
If possible, try avoiding the SUBSTRING function in your WHERE clauses. Depending on how it is constructed, using the SUBSTRING function can force a table scan instead of allowing the Optimizer to use an Index (assuming there is one).

If the substring you are searching for does not include the first character of the column you are searching for, then a table scan is performed.

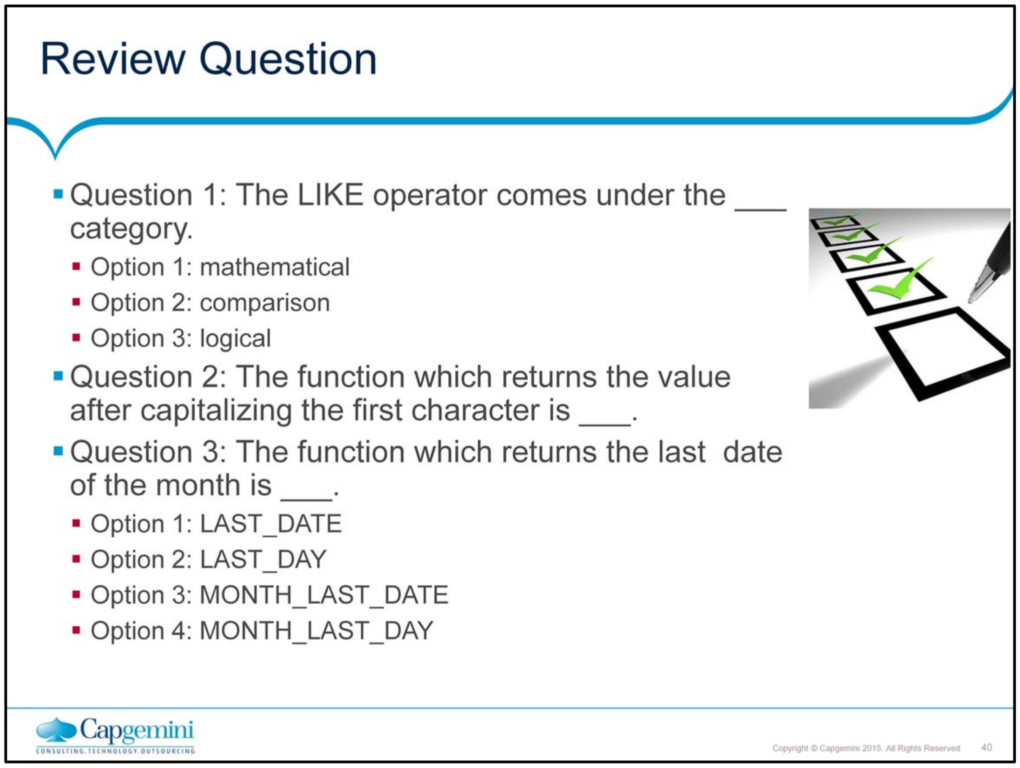
In case of conversion functions, If value to be converted is not in right format, then Oracle will throw an error



Summary

- In this lesson, you have learnt:
  - Operators - Mathematical, comparison, Logical
  - SQL (single-row) functions
    - Character functions
    - Number functions
    - Date functions
    - Conversion functions
    - Miscellaneous Single-row functions

Add the notes here.



Review Question

- Question 1: The LIKE operator comes under the ___ category.
  - Option 1: mathematical
  - Option 2: comparison
  - Option 3: logical
- Question 2: The function which returns the value after capitalizing the first character is ___.
- Question 3: The function which returns the last date of the month is ___.
  - Option 1: LAST_DATE
  - Option 2: LAST_DAY
  - Option 3: MONTH_LAST_DATE
  - Option 4: MONTH_LAST_DAY

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

40

Add the notes here.