Les03 Using Single Row Functions

# Purpose of chapter is to show how to further *Customize output*

**Objectives**

**After completing this lesson, you should be able to do the following:**

- Describe various types of functions that are available in SQL

- Use character, number, and date functions in **SELECT** Statements

- Describe the use of conversion functions

# Objectives

Functions 🡪 make the basic query block more powerful,

and

🡪 they are used to manipulate data values.

This is the first of two lessons that explore functions.

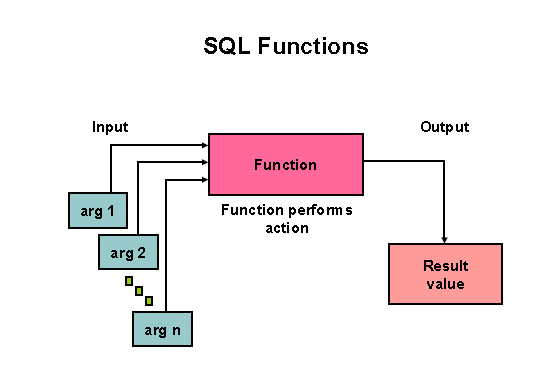
Focus is on

Single-row character, number, and date functions

Functions that convert data from one type to another

-- For example, conversion from character data to numeric data

SQL Functions 3-3



**SQL functions**

Functions are very powerful feature of SQL. They can be used to do the following:

Perform calculations on data

Modify individual data items

Manipulate output for groups of rows

Format dates and numbers for display

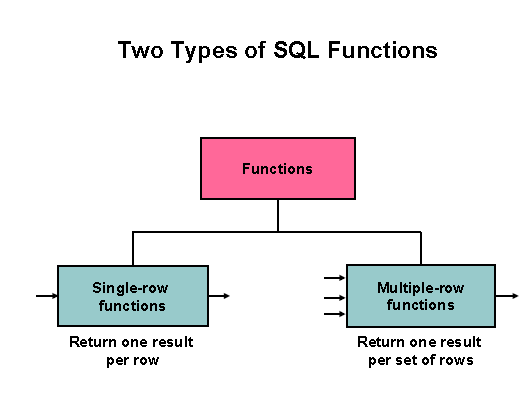
Convert column data types

SQL functions sometimes take arguments and always return a value

**Note:**

**Most of the functions that are described in this lesson are specific to a version of SQL**

**3-4**

****

**SQL functions**

**2 Types of Functions:**

**Single-Row functions**

**Multiple-row functions**

**Single-Row functions**

These functions operate on single rows only and return one result per row.

There are different types of Single-Row functions as follows:

Character

Number

Date

Conversion

General

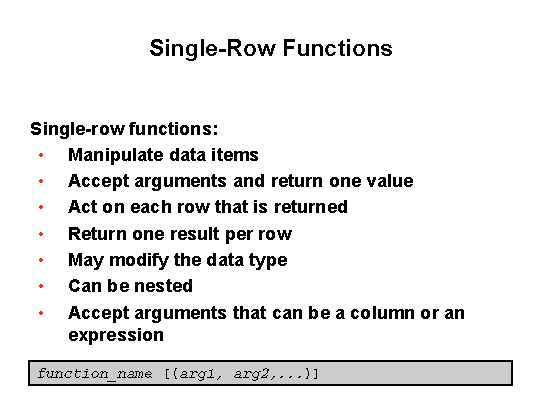
**Multiple-row functions**

Functions can manipulate groups of rows to give one result per group of rows.

These functions are also called group functions.

Note: we will only cover some of these on the course for all others refer to the oracle SQL reference guide.

**3-5**

****

**Single-Row functions**

These functions manipulate data items.

Be a set to one or more arguments and return a single value for each row that is retrieved by the query.

An argument can be one of the following:

User supplied constant

Variable value

Column name

Expression

The cheers of single row functions include:

Acts on each row that is returned by the query

Returns one result per row

May possibly return a different data type than the one that is referenced

The function expects one or more arguments

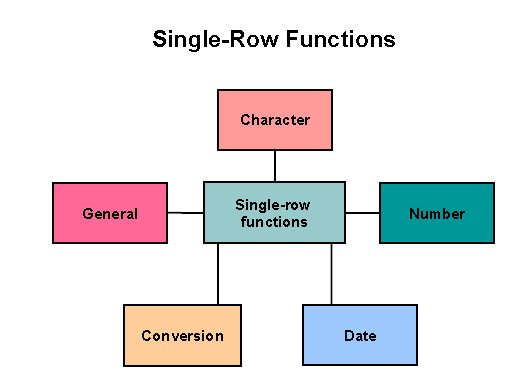
Can be used in THE Select

Where

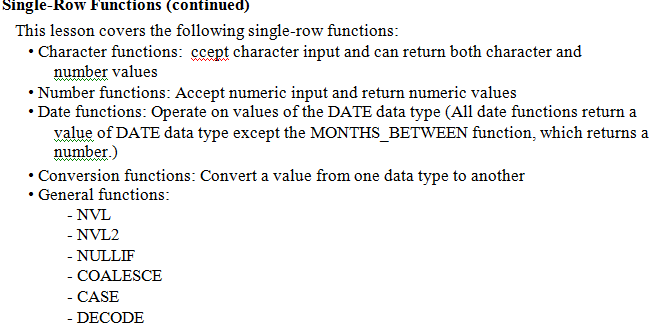
Order by

- can also be nested

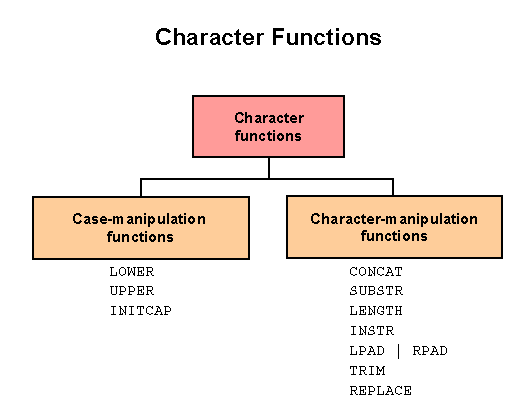
**3-6**

****

**Only the following are covered in this chapter**

****

**3-7**

****

**Function accepts character data 🡪 🡪 returns character and numeric data**

**2 groups 🡪 Case Manipulation**

**🡪 Character Manipulation**

EXAMPLES on next slides

**LOWER (**Column or Expression**)**

**UPPER**

**INITCAP –** changes string to Initial letter in each word is capitalized

**SUBSTR** – needs string or column and starting position and length

**CONCAT** – like || -- needs 2 arguments

**LENGTH** – returns number of characters in the expression

**SELECT LENGTH (CONCAT (first\_name, last\_name)) from employees**

**INSTR –** returns the numeric position of a named string

-- you can give it a starting position before counting

**LPAD** – pads the character value right justified

**RPAD** – pads the character value shown by the amount not filled by the filed

**select RPAD ( first\_name, 9 , '\*' ) from employees**

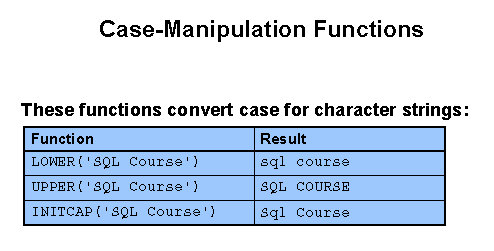
**TRIM**

**REPLACE**

**Examples on next set of slides**

|  |
| --- |
| Ellen\*\*\*\* |
| Curtis\*\*\* |
|  |

3-9

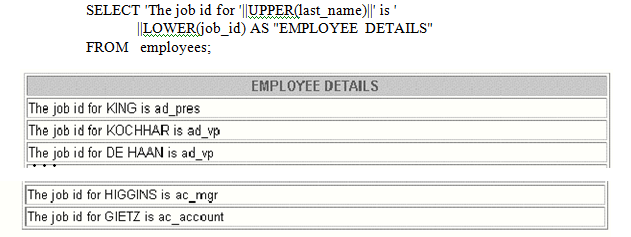


SELECT LOWER (first\_name)

FROM employees

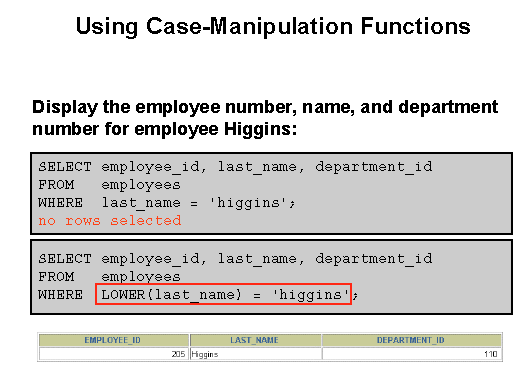


NOTE: The column headings are not business-like and need fixing



3-10

|  |
| --- |
| This would be a substitution variable to allow flexible inputs |



Example 1:

Because Higgins is all in lower case it does not find a match in the table

Example: 2

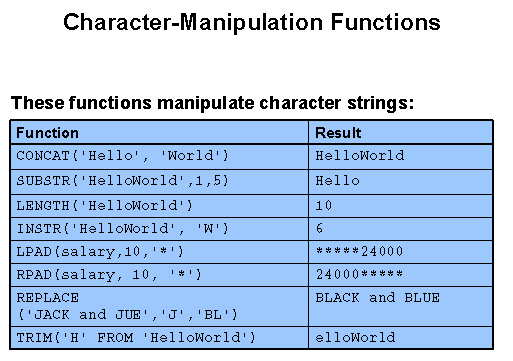
Convert the data stored in the database to LOWER case and match it to the input

This is done often on input screens

IMPROVEMENT:

Convert both the column and the input to the same case

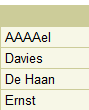
3-11



Demonstrate:

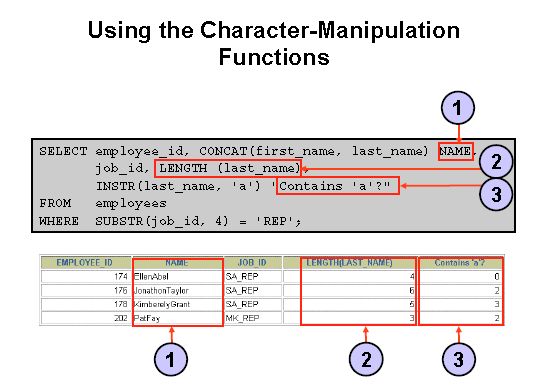
SELECT REPLACE (last\_name, 'Ab' , 'AAAA')

FROM employees



**Note: You can use functions such as UPPER and LOWER with ampersand substitution. For example, use UPPER ('&job\_title') so that the user does not have to enter the job title in a specific case.**

3-12



The slide example displays employee

1 first names and last names joined together,

2 the length of the employee last name, and

3 the numeric position of the letter a in the employee last name

For all employees

Who have the string REP contained in the job ID

Starting at the fourth position of the job ID.

**Exercise:**

Modify the SQL statement in the slide to display the data for those employees whose last names *end with the letter n.*

SELECT employee\_id,

CONCAT (first\_name, last\_name) NAME,

LENGTH (last\_name),

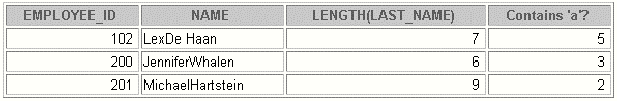
INSTR (last\_name, 'a') "Contains 'a'?"

FROM employees

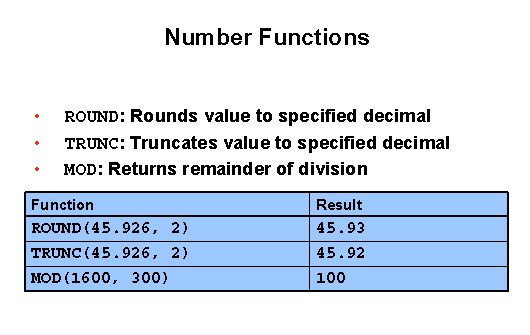
WHERE SUBSTR(last\_name, -1, 1) = 'n';

The -1 means start at 1 less than the end and process 1 value (which is now the end)

-- And is that value equal to n



3-13

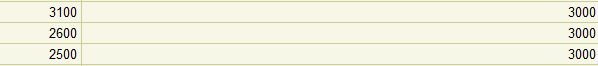


This is a straight forward example

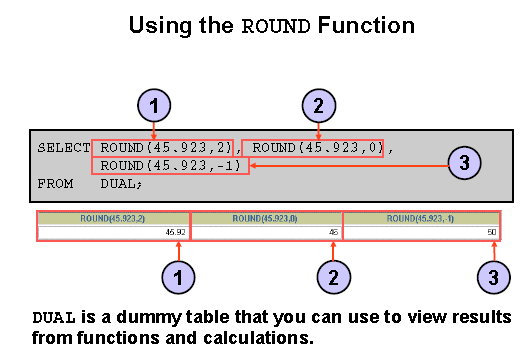
Try this

SELECT salary, round (salary, -3)

FROM employees



3-14



Again this is simple functions

**NOTE:**

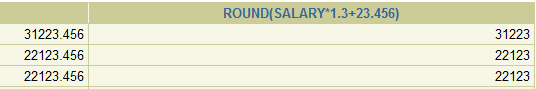
**DUAL used because SELECT and FROM are mandatory**

**… but the data doesn’t come from any columns or tables**

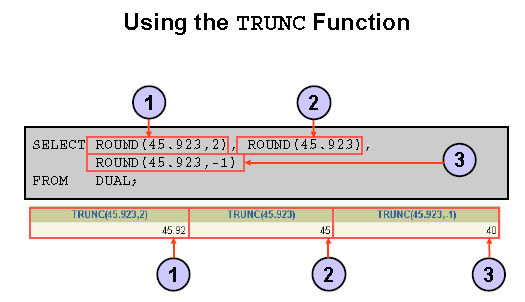
**If use 0 or no value it is rounded to zero decimal places**

SELECT salary \* 1.3 +23.456, round (salary \*1.3+23.456)

FROM employees

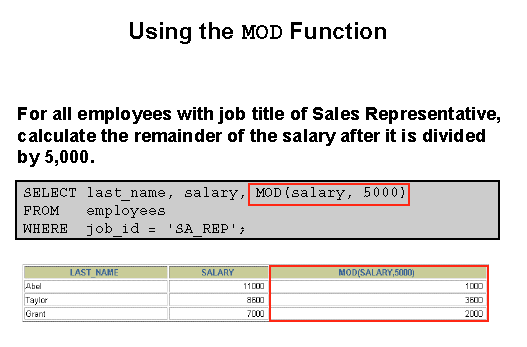


3-15



Works the same as ROUND

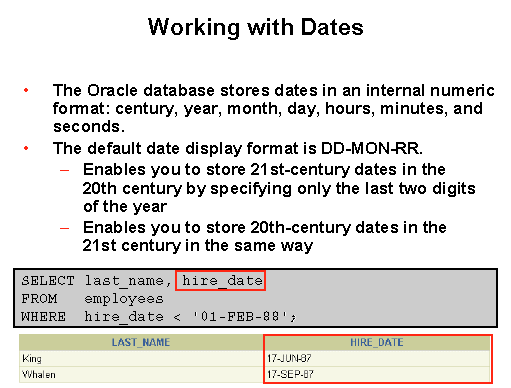
3-16



Gives the remainder …. Similar to C programming

Used often to determine if a value is **odd or even**

3-17 DATES



NOTE:

Default date display format. Company may choose different defaults for display.

Actual date stored differently.

June 17, 1987, 5:10:43 p.m

# RR – goes back to pre-2000 times to avoid a problem

The Oracle database stores dates in an internal numeric format, representing the century, year, month, day, hours, minutes, and seconds.

The default display and input format for any date is DD-MON-RR. Valid Oracle dates are between January 1, 4712 B.C., and December 31, 9999 A.D.

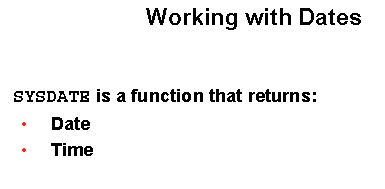
In the example in the slide, the HIRE\_DATE column output is displayed in the default format DD-MON-RR. However, dates are not stored in the database in this format. All the components of the date and time are stored. So, although a HIRE\_DATE such as 17-JUN-87 is displayed as day, month, and year, there is also time and century information associated with the date. The complete data might be June 17, 1987, 5:10:43 p.m.

CENTURY YEAR MONTH DAY HOUR MINUTE SECOND

19 87 06 17 17 10 43

Note: century or year stored as 4 digits even if displayed as 2

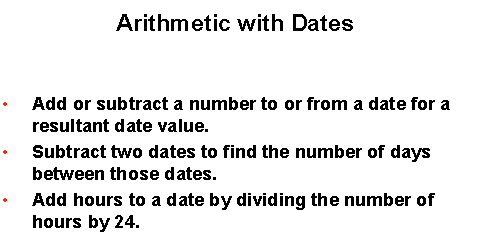
3-19



**SELECT SYSDATE**

**FROM DUAL**

**3-20**

****

Because the database stores dates as numbers, you can perform calculations using arithmetic operators such as addition and subtraction. You can add and subtract number constants as well as dates.

You can perform the following operations

Date + number

Date – number

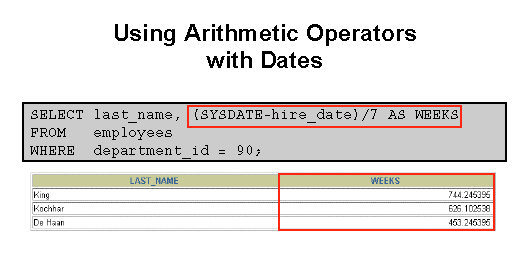
Date – Date

Date +number/24 Date -- Adds a number of hours to a date

MAJOR IMPORTANCE TO BUSINESS

BUSINESS RUNS ON DATES AND DOLLARS

**3-21**



**PROBLEM:**

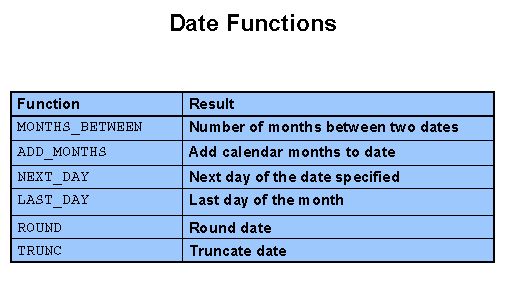
**Find how many weeks an employee has worked at the company**

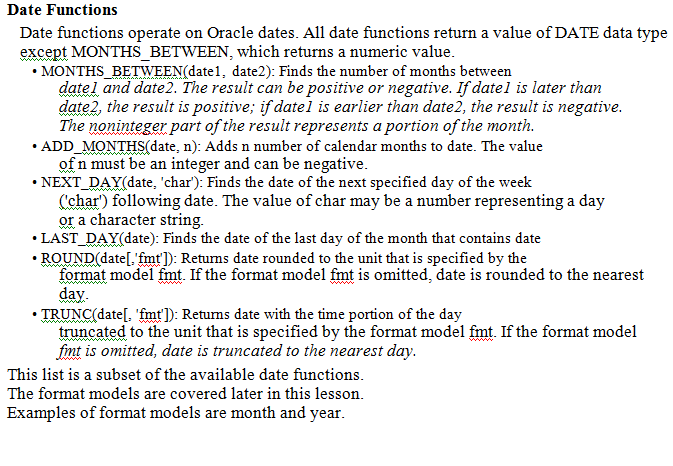
* **and only for department 90**

NOTE:

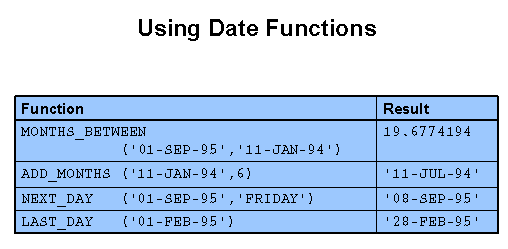
If you try this you get a different answer. SYSDATE is now and not when the slide was done

3-22



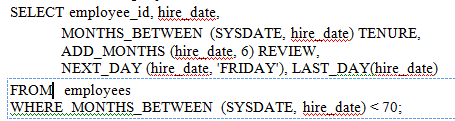


3-23

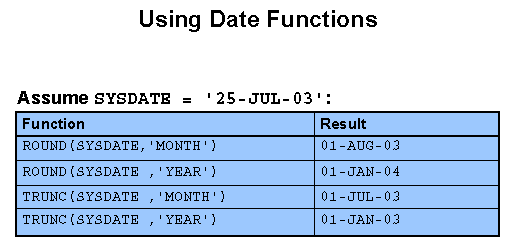


**PROBLEM: Try this**

Display the employee number, hire date, number of months employed, six-month review date, first Friday after hire date, and last day of the hire month for all employees who have been employed for fewer than 70 months.



3-24



The ROUND and TRUNC functions can be used for number and date values.

When used with dates, these functions round or truncate to the specified format model. Therefore, you can round dates to the nearest year or month.

**PROBLEM:**

Compare the hire dates for all employees who started in 1997. Display the employee number, hire date, and start month using the ROUND and TRUNC functions.

SELECT employee\_id,

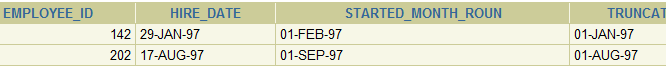
hire\_date,

ROUND(hire\_date, 'MONTH') as Started\_Month\_Rounded,

TRUNC(hire\_date, 'MONTH') as Truncated

FROM employees

WHERE hire\_date LIKE '%97';



Express in Jan 2015 has a different default date style

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **HIRE\_DATE** | | **STARTED\_MONTH\_ROUNDED** | **TRUNCATED** |
| 142 | 01/29/1997 | | 02/01/1997 | 01/01/1997 |
| 202 | 08/17/1997 | | 09/01/1997 | 08/01/1997 |
|  | |

EXERCISE for you to do at back of chapter

3-26

Conversion Functions

2 Types

- Implicit

- Explicit

3-27

IMPLICIT - what the Oracle software does itself.

EXPLICIT - what a specific conversion function does

See notes for IMPLICIT and EXPLICIT explanations

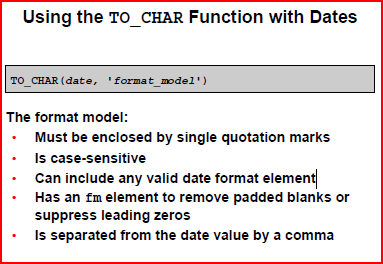
3-28

3-29

3-30

3-31

3-32



select last\_name, salary,

**TO\_CHAR (hire\_date, 'YYYY-Month-DD')**

from employees

where salary = '11000'

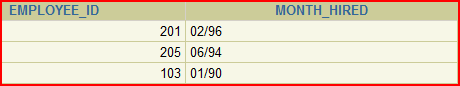


SELECT EMPLOYEE\_ID,

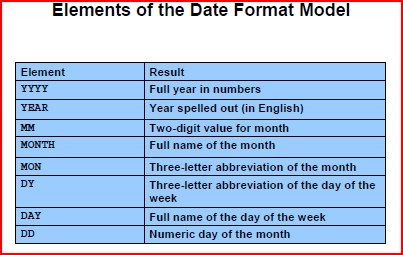
TO\_CHAR (HIRE\_DATE, 'MM/YY' ) Month\_Hired

FROM EMPLOYEES

WHERE LAST\_NAME like 'H%'

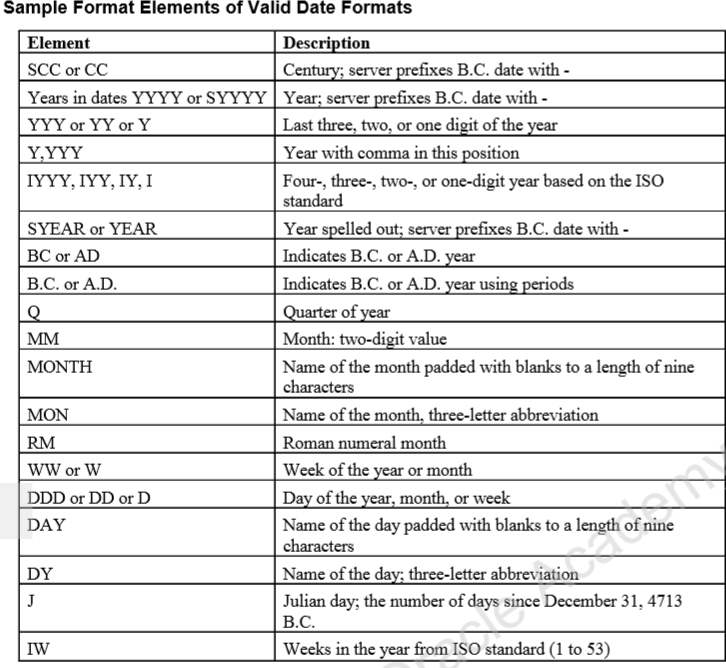


3-33 --4-12



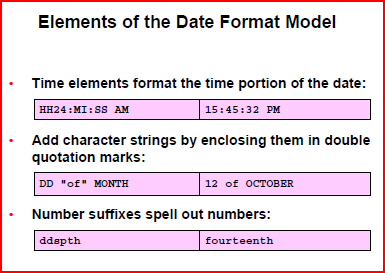
3-34 – 4-13

**MANY OTHERS**



Try out some of them to see what they do

3-35-4-14

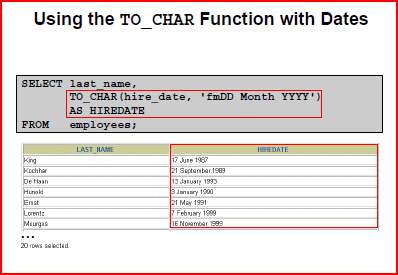


Again another set of formats

# REMEMBER:

# Business uses dates

3-37



# Using the TO\_CHAR function to add more control

SELECT last\_name,

**TO\_CHAR(hire\_date, 'fmDdspth "of" Month YYYY fmHH:MI')**

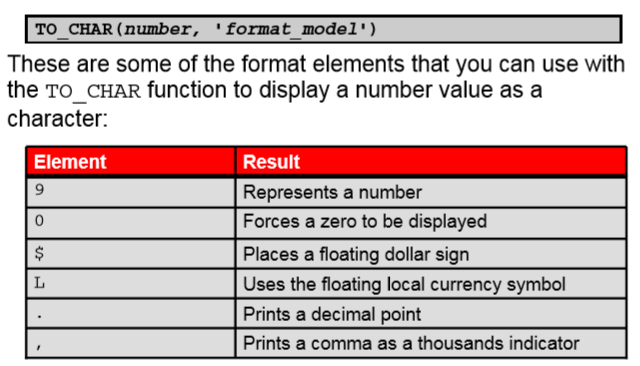
FROM employees

==> Try it with 24 hour format and see results

|  |  |
| --- | --- |
| **LAST\_NAME** | **TO\_CHAR(HIRE\_DATE,'FMDDSPTH"OF"MONTHYYYYFMHH:MI')** |
| King | Seventeenth of June 1987 12:00 |
| Kochhar | Twenty-First of September 1989 12:00 |
| De Haan | Thirteenth of January 1993 12:00 |
| Hunold | Third of January 1990 12:00 |
| Ernst | Twenty-First of May 1991 12:00 |
| Lorentz | Seventh of February 1999 12:00 |
| Mourgos | Sixteenth of November 1999 12:00 |
| Rajs | Seventeenth of October 1995 12:00 |
| Davies | Twenty-Ninth of January 1997 12:00 |
| Matos | Fifteenth of March 1998 12:00 |

Plus more rows

3-38 Using TO\_CHAR with number



SELECT last\_name,

TO\_CHAR(salary, '$99,999.00') as SALARY

FROM employees;

Problems of a floating dollar sign is that the field is left justified as a character field and numbers don't align well.

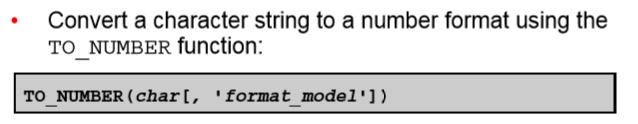
AGAIN SQL wasn't meant to be fancy

|  |  |
| --- | --- |
| **LAST\_NAME** | **SALARY** |
| King | $24,000.00 |
| Kochhar | $17,000.00 |
| De Haan | $17,000.00 |
| Hunold | $9,000.00 |
| Ernst | $6,000.00 |
| Lorentz | $4,200.00 |

Convert character string to NUMBER or DATE

03-39--4-20

General format of conver to a number

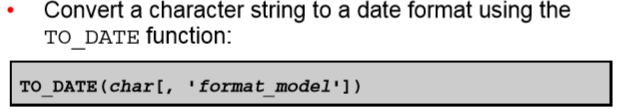


SELECT to\_number('1234')-2

from dual;

Convert a character to a date

4-20



Try this:

SELECT last\_name, to\_char(hire\_date, 'DD-Mon-YYYY')

from employees

where hire\_date < to\_date('01-Jan-90' , 'DD-Mon-YY');

NOTE the results

Wrong results because it assume with YY that it was 2090

Change it to RR

**TRY THIS**

Find employees hired on May 24, 1999

SELECT last\_name, hire\_date

from employees

where hire\_date = to\_date('May 24, 1999', 'fxMonth DD, YYYY');

NOTE: Do the above but add some spaces in the date

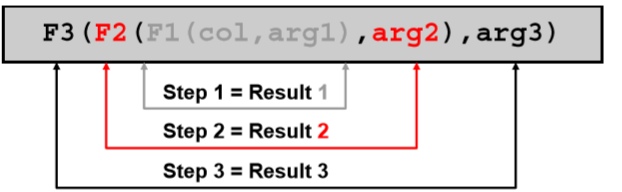
Add some spaces in the format and rerun

Nesting Functions

4-24

- Single row functions can be nested to any level

- Nested functions evaluate from the innermost or deepest level



Examples of Nesting Functions

4-25

**TRY THIS:**

Display the

- Last name of the employees in department 60

- And their new email name made up of first 4 characters of last name with \_US added all to appear in uppercase

- make the title of column 2 nicer

Example Higgins becomes HIGG\_US

SELECT last\_name,

UPPER (CONCAT(SUBSTR(LAST\_NAME, 1, 4) , '\_US')) as "Email"

FROM EMPLOYEES

WHERE DEPARTMENT\_ID = 60;

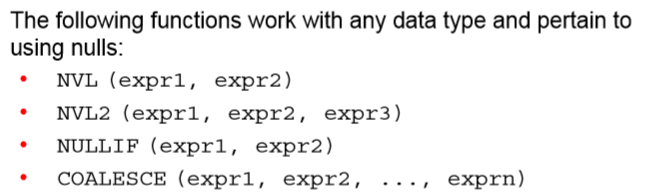
|  |  |
| --- | --- |
| **LAST\_NAME** | **Email** |
| Hunold | HUNO\_US |
| Ernst | ERNS\_US |
| Lorentz | LORE\_US |

General Functions

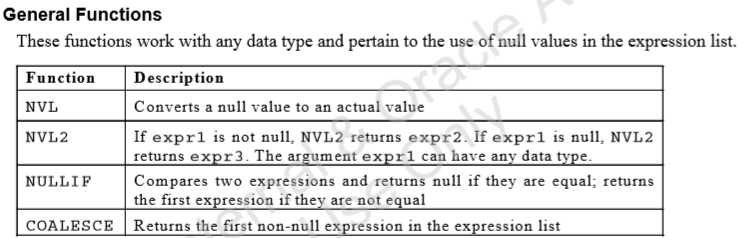
04-27

**Handling NULLS**

General Format



**The most used is NVL**



NULL Examples

PROBLEM 1:

List last name

Salary

And the result of multiplying salary times commission percent

SELECT last\_name, salary, salary\*commission\_pct

FROM employees;

The effect of a NULL value in a calculation is to result in a NULL displayed

Some of the output

|  |  |  |
| --- | --- | --- |
| Rajs | 3500 | - |
| Davies | 3100 | - |
| Matos | 2600 | - |
| Vargas | 2500 | - |
| Zlotkey | 10500 | 2100 |
| Abel | 11000 | 3300 |
| Taylor | 8600 | 1720 |
| Grant | 7000 | 1050 |
| Whalen | 4400 | - |
| Hartstein | 13000 | - |

Correction: (might be)

SELECT last\_name, salary, salary\* nvl(commission\_pct,0)

FROM employees;

|  |  |  |
| --- | --- | --- |
| Rajs | 3500 | 0 |
| Davies | 3100 | 0 |
| Matos | 2600 | 0 |
| Vargas | 2500 | 0 |
| Zlotkey | 10500 | 2100 |
| Abel | 11000 | 3300 |
| Taylor | 8600 | 1720 |
| Grant | 7000 | 1050 |
| Whalen | 4400 | 0 |
| Hartstein | 13000 | 0 |

**PROBLEM 2:**

Add up the totals – next chapter

NULL with date

4-28

NVL (hire\_date, '01-JAN-2015')

NULL with character

Suppose you are missing any value in a character field and you wanted to not leave it as NULL, but wanted it to appear as Unavailable.

NVL ( city , 'Unavailable' )

BAD EXAMPLE … but

SELECT last\_name, NVL(to\_char(commission\_pct), to\_char('???'))

FROM employees;

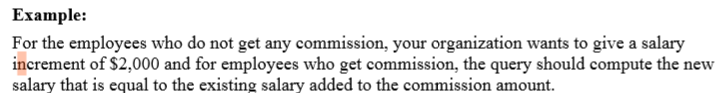
|  |  |
| --- | --- |
| Davies | ??? |
| Matos | ??? |
| Vargas | ??? |
| Zlotkey | .2 |
| Abel | .3 |
| Taylor | .2 |
| Grant | .15 |
| Whalen | ??? |
| Hartstein | ??? |

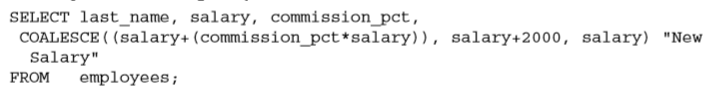
READ the book for the other NULLs

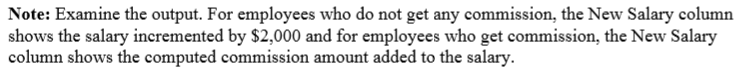
COALESCE

4-32 and 4-33

# Evaluates multiple expressions --- read the book







SELECT last\_name, salary, commission\_pct,

coalesce( (salary +(commission\_pct\*salary)),

salary + 2000,

salary) as "New Salary"

FROM employees;

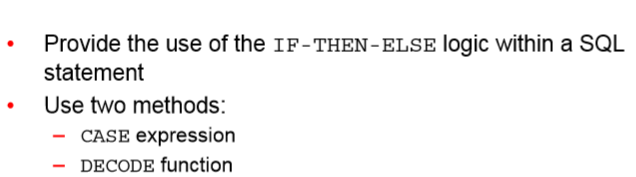
1st value evaluates as a NULL so filled with salary + 2000

1st value wasn't a null so the calculated expression appears of salary plus salary times commission

|  |  |  |  |
| --- | --- | --- | --- |
| Davies | 3100 | - | 5100 |
| Matos | 2600 | - | 4600 |
| Vargas | 2500 | - | 4500 |
| Zlotkey | 10500 | .2 | 12600 |
| Abel | 11000 | .3 | 14300 |
| Taylor | 8600 | .2 | 10320 |
| Grant | 7000 | .15 | 8050 |
| Whalen | 4400 | - | 6400 |
| Hartstein | 13000 | - | 15000 |

CONDITIONAL EXPRESSIONS

4-35

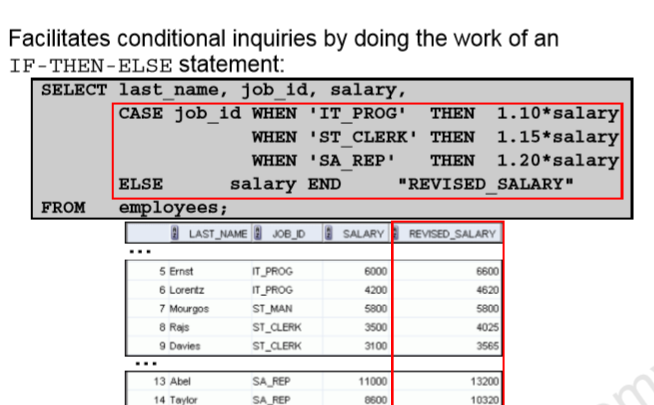


CASE applies to ANSI standard

DECODE is Oracle syntax (from an earlier period)

CASE

4-38



NOTE: -- ST\_MAN as a job\_id didn't fit any of the cases so the ELSE took effect and the new salary was just the same as the salary

DECODE

4-39

PLEASE READ