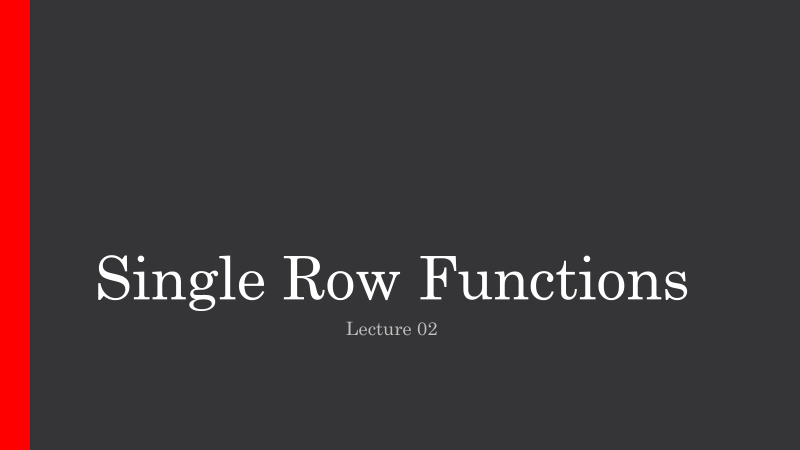
Les02 Using Single Row Functions



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

Companies need lots of different answers to use in decision making

**Objectives**

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

- Describe various types of functions that are available in SQL

- Use 1 character,

2 number, and

3 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.

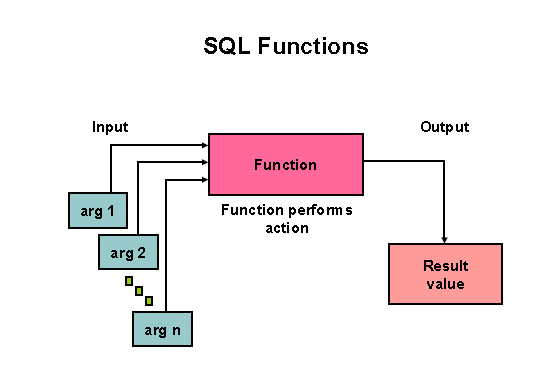
This is the first of two lessons that explore functions.

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**

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

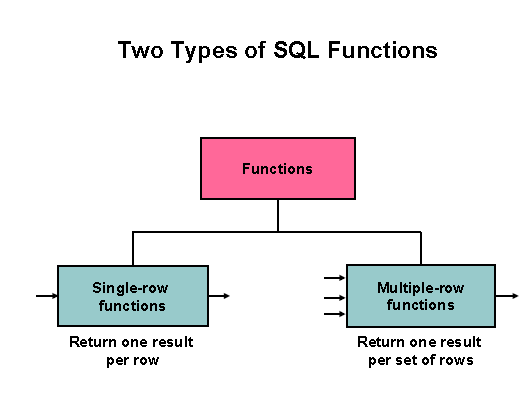
Format dates and numbers for display

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**

****

**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 for every row acted on.

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

Character

Number

Date

Conversion

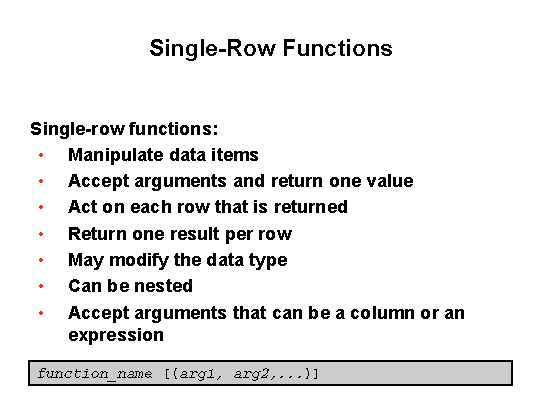
General

**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.

****



**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 actions 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

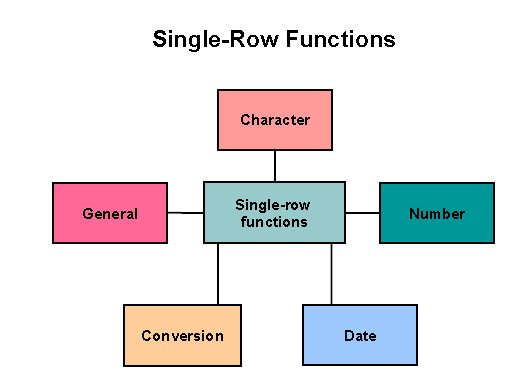
The function expects one or more arguments

Can be used in THE Select

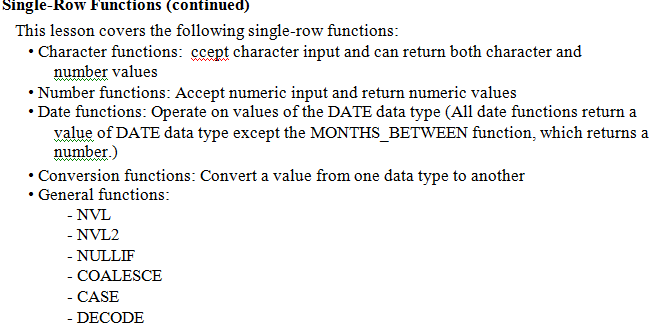
Where

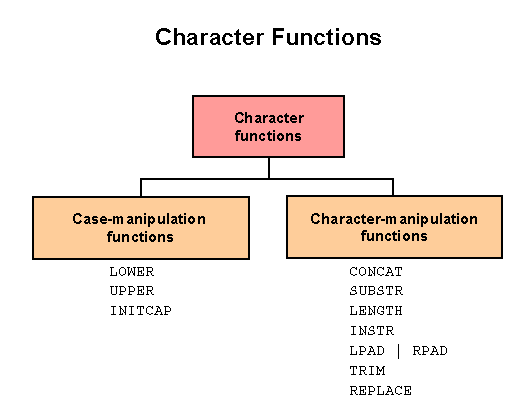
Order by

- can also be nested

****

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

****

****

**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

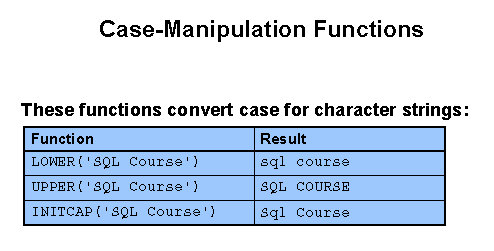
**select RPAD ( first\_name, 9 , '\*' ) from employees**

**TRIM**

**REPLACE**

**Examples on next set of slides**

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

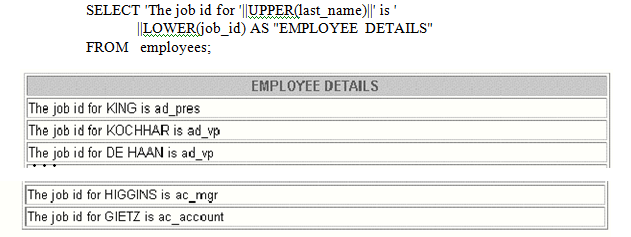


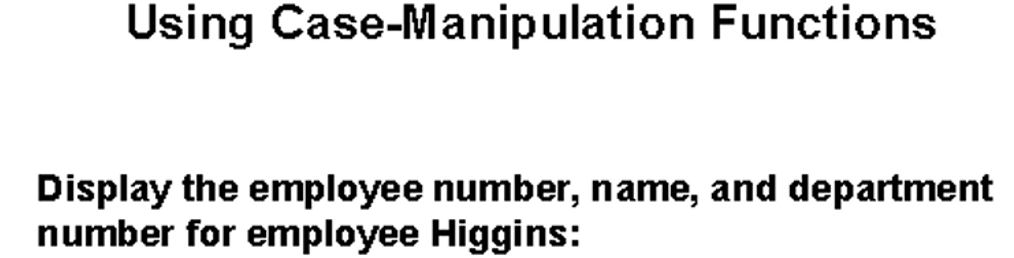
SELECT LOWER (first\_name)

FROM employee

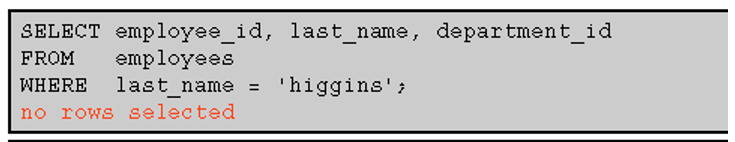


NOTE: The column headings are not business-like and need fixing





======================================================================



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

**Improve it**

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

select employee\_id, last\_name, department\_id

from employee

|  |
| --- |
| This would be a substitution variable to allow flexible inputs.  ‘&last\_name’ |

where lower(last\_name) = 'higgins'

**Make user enter the data**

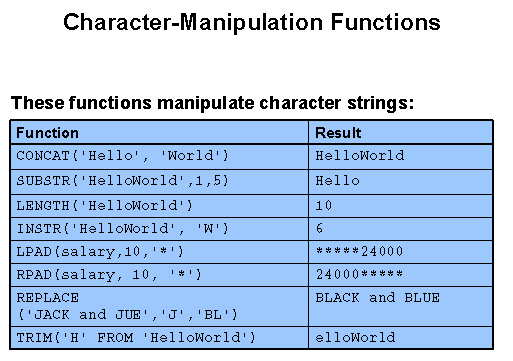
select employee\_id, last\_name, department\_id

from employees

where **lower**(last\_name) = **lower**(‘&Last\_Name’)

Case statement on BOTH sides covers all possibilites

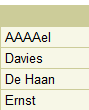




Demonstrate REPLACE:

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.**

Using Character Manipulation

**PROBLEM:**

Display the first name and last name joined. Call that column NAME

Display job\_id

Length of last\_name

What position in last name is the letter 'a'

But only show those whose where job\_id has REP starting in position 4



**SELECT**

**employee\_id,**

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

**job\_id,**

**LENGTH (last\_name),**

**INSTR (last\_name, 'a') "contains an 'a'"**

**FROM employee**

**WHERE SUBSTR (job\_id, 4) = 'REP';**

EMPLOYEE\_ID NAME JOB\_ID LENGTH(LAST\_NAME) contaisn an 'a'

----------- --------------------------------------------- ---------- ----------------- ---------------

174 EllenAbel SA\_REP 4 0

176 JonathonTaylor SA\_REP 6 2

178 KimberelyGrants SA\_REP 6 3

180 Spencede Man SA\_REP 6 5

202 PatFay MK\_REP 3 2

The above 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 string, employee last name

And for all employees

Who have the string REP contained in the job ID

Starting at the fourth position of the job ID.

**PROBLEM:**

Modify the previous SQL statement 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),

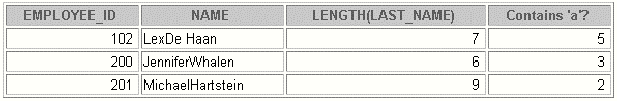
INSTR (last\_name, 'a') --"Contains 'a'?"  Gets position of letter a anywhere

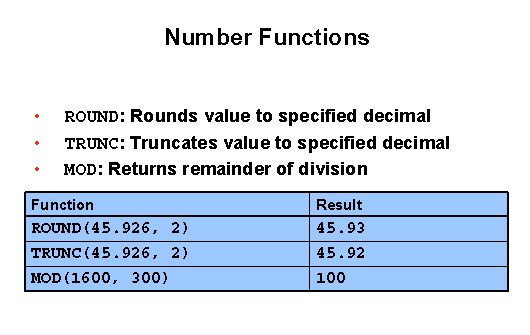
FROM employee

WHERE SUBSTR(last\_name, -1, 1) = 'n';  this gets those with last \_name ending in 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





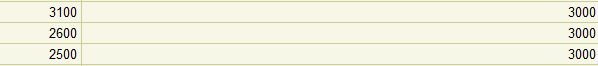
This is a straightforward example

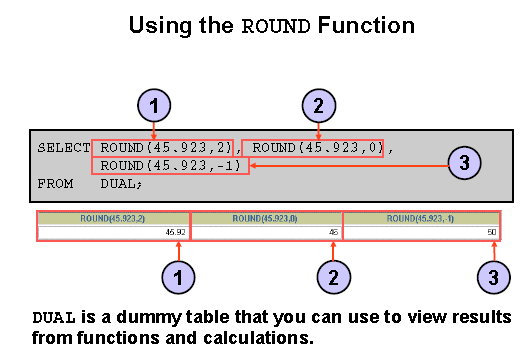
Try this

SELECT salary, -- the original salary in table

round (salary, -3) -- the same salary rounded

FROM employee;





Again, this is simple functions

**NOTE:**

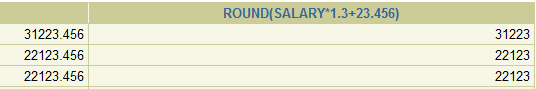
**DUAL used because SELECT and FROM are mandatory**

**… 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) -- rounding to whole dollars

FROM employees

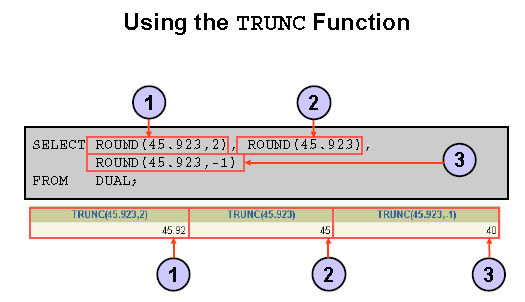


**13673.456 13673**

**14323.456 14323**

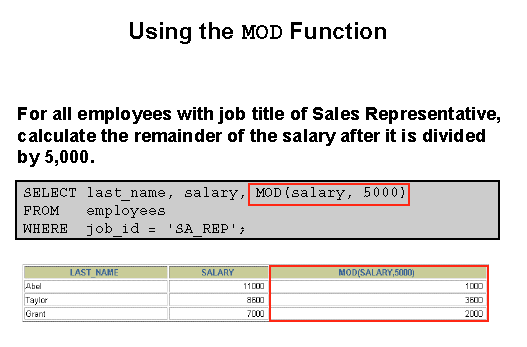
**11203.456 11203**

**9123.456 9123**



Replace in this slide ROUND with TRUNC!

What do you notice as compared to the previous slide?



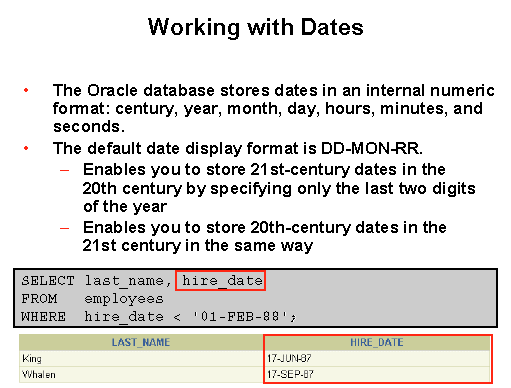
Gives the remainder .. AFTER the amount is subtracted as many times as possible

…. Like C programming

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



DATES



NOTE:

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

Actual date stored differently. Full date and time

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.

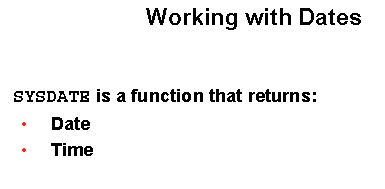
The default display and input format for any date is DD-MON-RR. Valid Oracle dates are between January 1, 4712 B.C., and December 31, 9999 A.D.

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

1987 06 17 17 10 43

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





**TEST IT**

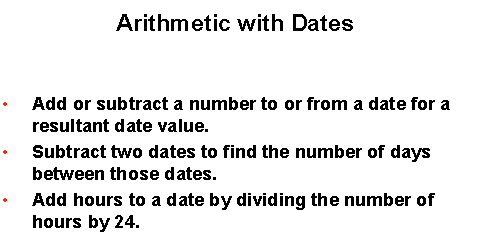
**SELECT SYSDATE**

**FROM DUAL**

**SYSDATE**

**---------**

**17-SEP-18**

****

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

On an invoice this would be billing date and due date of the sale

**If payment is due in 30 days**

**SELECT sysdate, sysdate + 30**

**FROM** **dual;**

**Using Arithmetic Operators with Dates**

**PROBLEM:**

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

* **and only for department 90**

**Answer looking for is:**

**LAST\_NAME Weeks Employed**

**------------------------- --------------**

**King 1526.509089**

**Kochhar 1408.366232**

**De Haan 1235.509089**

SELECT last\_name, (sysdate - hire\_date)/7 "Weeks Employed" -- returns days converted to weeks

FROM employee

WHERE department\_id = 90;

This answer is not very good …. Improve it

SELECT last\_name, trunc((sysdate - hire\_date)/7, 2) "Weeks Employed"

FROM employee

WHERE department\_id = 90;

LAST\_NAME Weeks Employed

Why does it end in .5 ?

------------------------- --------------

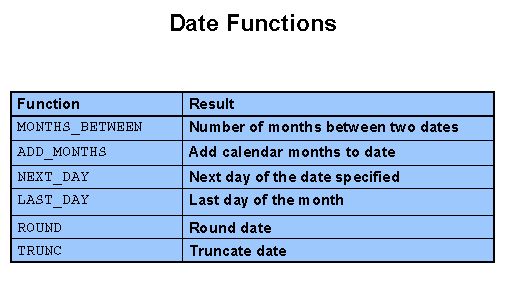
King 1526.5

Kochhar 1408.36

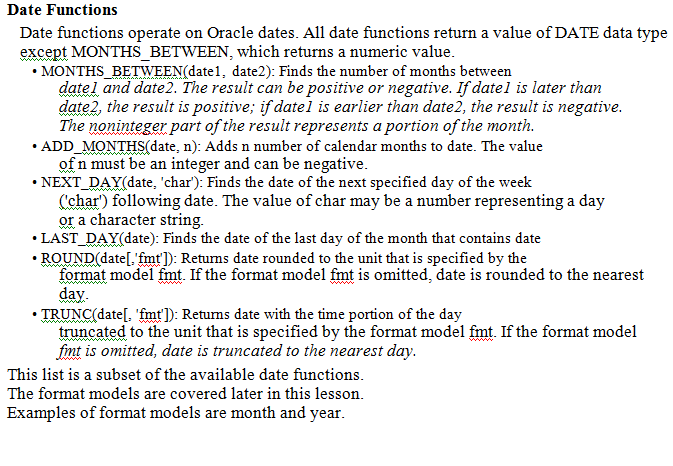
De Haan 1235.5

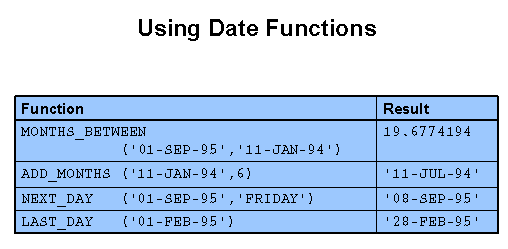
NOTE:

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



EX: next page





EXAMPLE:

SELECT NEXT\_DAY ('17-SEP-2018','TUESDAY') AS "Next Tuesday"

FROM dual;

**Next Tuesday**

**------------**

**18-SEP-18**

**PROBLEM: Try this**

Display the employee number, hire date,

- number of months employed,

- six-month from now is the employees review date,

- what is the first Friday after hire date, and

- last day of the hire month

for all employees who have been employed for fewer than 70 months.

ANS: Next page

SELECT employee\_id, hire\_date,

MONTHS\_BETWEEN (SYSDATE, hire\_date) "Seniority",

ADD\_MONTHS (hire\_date, 6) "Review Date",

NEXT\_DAY (hire\_date, 'Friday') “NDay”

LAST\_DAY (hire\_date) “LDay”

FROM employee

WHERE MONTHS\_BETWEEN (SYSDATE, hire\_date) > 70;

EMPLOYEE\_ID HIRE\_DATE Seniority Review Da NDay LDay

----------- --------- ---------- --------- --------- ---------

100 17-JUN-87 395.180352 17-DEC-87 19-JUN-87 30-JUN-87

101 21-SEP-89 368.05132 21-MAR-90 22-SEP-89 30-SEP-89

102 13-JAN-93 328.309384 13-JUL-93 15-JAN-93 31-JAN-93

103 03-JAN-90 364.631965 03-JUL-90 05-JAN-90 31-JAN-90

104 21-MAY-91 348.05132 21-NOV-91 24-MAY-91 31-MAY-91

107 07-FEB-99 255.502933 07-AUG-99 12-FEB-99 28-FEB-99

124 16-NOV-99 246.21261 16-MAY-00 19-NOV-99 30-NOV-99

141 17-OCT-95 295.180352 17-APR-96 20-OCT-95 31-OCT-95

142 29-JAN-97 279.793255 29-JUL-97 31-JAN-97 31-JAN-97

143 15-MAR-98 266.244868 15-SEP-98 20-MAR-98 31-MAR-98

144 09-JUL-98 262.438417 09-JAN-99 10-JUL-98 31-JUL-98

149 29-JAN-00 243.793255 29-JUL-00 04-FEB-00 31-JAN-00

174 11-MAY-96 288.3739 11-NOV-96 17-MAY-96 31-MAY-96

176 24-MAR-98 265.954546 24-SEP-98 27-MAR-98 31-MAR-98

178 24-MAY-99 251.954546 24-NOV-99 28-MAY-99 31-MAY-99

200 17-SEP-87 392.180352 17-MAR-88 18-SEP-87 30-SEP-87

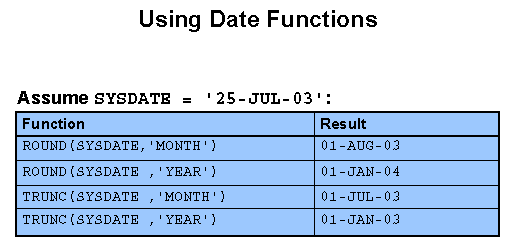
201 17-FEB-96 291.180352 17-AUG-96 23-FEB-96 29-FEB-96

202 17-AUG-97 273.180352 17-FEB-98 22-AUG-97 31-AUG-97

205 07-JUN-94 311.502933 07-DEC-94 10-JUN-94 30-JUN-94

206 07-JUN-94 311.502933 07-DEC-94 10-JUN-94 30-JUN-94

20 rows selected.



Using Date Functions

Try these to see result based on current sysdate

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.

ORIGINAL DATE is in SEPTEMBER 2018

|  |  |
| --- | --- |
| ROUND \_\_ MONTH  select round(sysdate, 'month')  from dual; | ROUND(SYSDATE,'MONTH')  ----------------------  01-OCT-18 |
| TRUNC \_\_\_ Month | TRUNC(SYSDATE,'MONTH')  ----------------------  01-SEP-18 |
|  |  |
| select round(sysdate, 'year')  from dual; | ROUND(SYSDATE,'YEAR')  ---------------------  01-JAN-19 |
|  | TRUNC(SYSDATE,'YEAR')  ---------------------  01-JAN-18 |

**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,

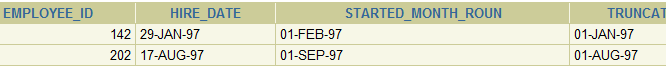
hire\_date,

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

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

FROM employee

WHERE hire\_date LIKE '%97';



OracleExpress in Jan 2015 has a different default date style, but result is the same data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |
|  | |

Conversion Functions

2 Types

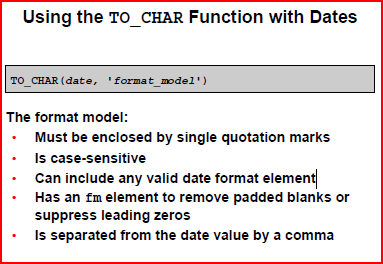
- Implicit

- Explicit

IMPLICIT - what the Oracle software does itself.

EXPLICIT - what a specific conversion function does

PURPOSE: To change the look of the date to meet requirements



select last\_name, salary,

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

from employee

where salary = 11000;



CHANGE REQUIREMENT

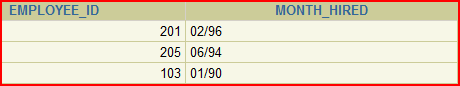
This example is changing it to MM/YY

SELECT EMPLOYEE\_ID,

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

FROM EMPLOYEE

WHERE LAST\_NAME like 'H%'



NOTE: you control the output format

**Then this**

SELECT EMPLOYEE\_ID,

TO\_CHAR (HIRE\_DATE, 'fmMM/DD/YY') “Hire\_Date”

FROM EMPLOYEE

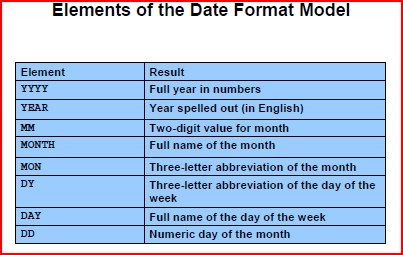
WHERE LAST\_NAME like 'H%'



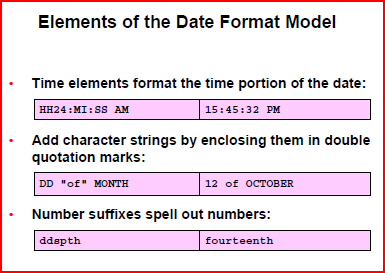
**Try fm with a lot of spaces**



**CAN YOU SEE THE DIFFERENCE**



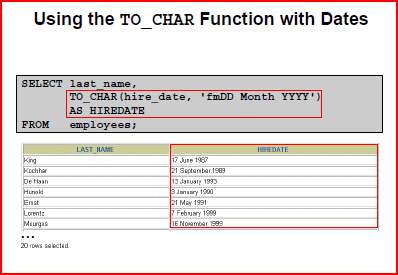




Again another set of formats

# REMEMBER:

# Business uses dates



Fm will get rid of leading zeros – see Lorentz

# Using the TO\_CHAR function to add more control

SELECT last\_name,

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

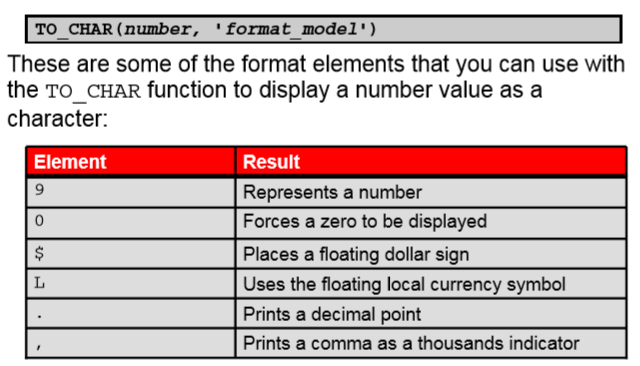
FROM employee

==> 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

Using TO\_CHAR with number



**SELECT last\_name,**

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

**FROM employee;**

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

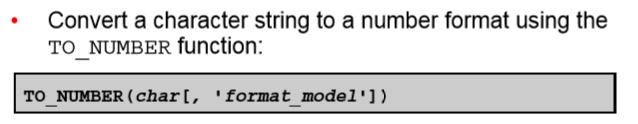
AGAIN SQL wasn't meant to be fancy.

BUT right justifies on other software

|  |  |
| --- | --- |
| **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

General format of conver to a number



SELECT to\_number('1234')-2 -- convert STRING of characters to a number less 2

from dual;

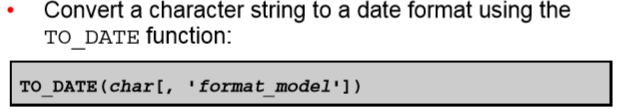
RESUKT:

TO\_NUMBER('1234')-2

-------------------

1232

Convert a character to a date



PROBLEM:

Find all persons hired BEFORE year 1990?

Try this:

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

from employee

where hire\_date < to\_date ('Januar 01, 90', 'Month dd, YY');

NOTE the results. Is it correct?

LAST\_NAME Hire Date

------------------------- --------------

King 17-Jun-1987

Kochhar 21-Sep-1989

De Haan 13-Jan-1993

Hunold 03-Jan-1990

Ernst 21-May-1991

Lorentz 07-Feb-1999

Mourgos 16-Nov-1999

Rajs 17-Oct-1995

… more rows here

Wrong results because it assumed with YY that it was 2090

Change it to RR model, and then it will understand that was 1990.

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

FROM employee

WHERE hire\_date < to\_date('January 01, 90', 'Month DD, RR');

LAST\_NAME Hire Date

------------------------- -----------

King 17-Jun-1987

Kochhar 21-Sep-1989

Whalen 17-Sep-1987

Same may be done when using 4 digits for a year in ANY format.

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

FROM employee

WHERE hire\_date < to\_date('January 01, 1990', 'Month dd, YYYY');

LAST\_NAME Hire Date

------------------------- -----------

King 17-Jun-1987

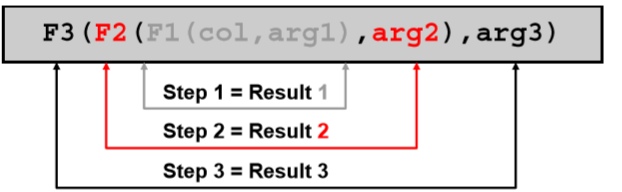
Kochhar 21-Sep-1989

Whalen 17-Sep-1987

Nesting Functions

- Single row functions can be nested to any level

- Nested functions evaluate from the innermost or deepest level



Examples of Nesting Functions

**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 much nicer looking

Example Higgins becomes HIGG\_US

SELECT last\_name,

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

FROM EMPLOYEE

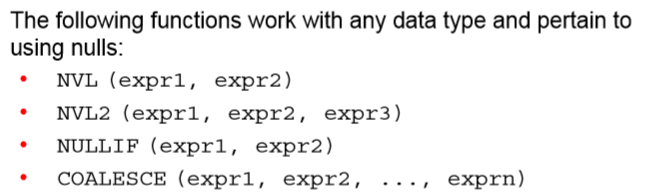
WHERE DEPARTMENT\_ID = 60;

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

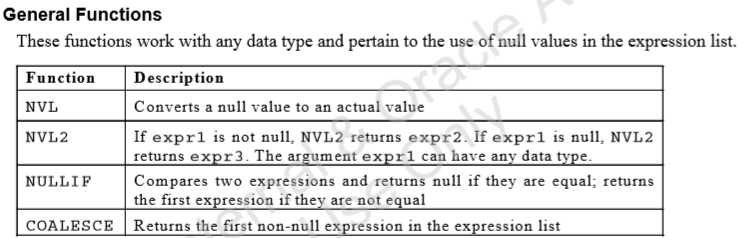
General Functions

**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 employee;

The effect of a NULL value in a calculation is to give a NULL result in display

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 employee;

|  |  |  |
| --- | --- | --- |
| 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

**NVL (hire\_date, '01-JAN-2015') if NULL then make it ...**

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 employee;

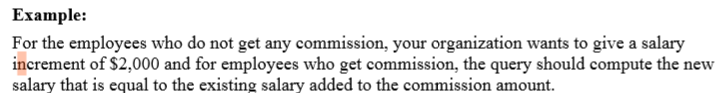
First needed to convert numeric field to a charater because want to diplay characters (the question mark)

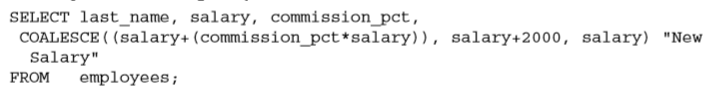
|  |  |
| --- | --- |
| Davies | ??? |
| Matos | ??? |
| Vargas | ??? |
| Zlotkey | .2 |
| Abel | .3 |
| Taylor | .2 |
| Grant | .15 |
| Whalen | ??? |
| Hartstein | ??? |

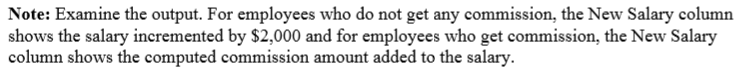
READ the PDF for the other NULLs

COALESCE

# Evaluates multiple expressions --- read the book







SELECT last\_name, salary, commission\_pct,

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

salary + 2000,

salary) as "New Salary"

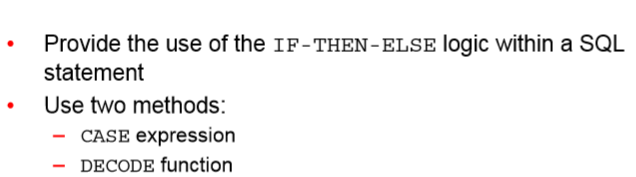
FROM employee;

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 |

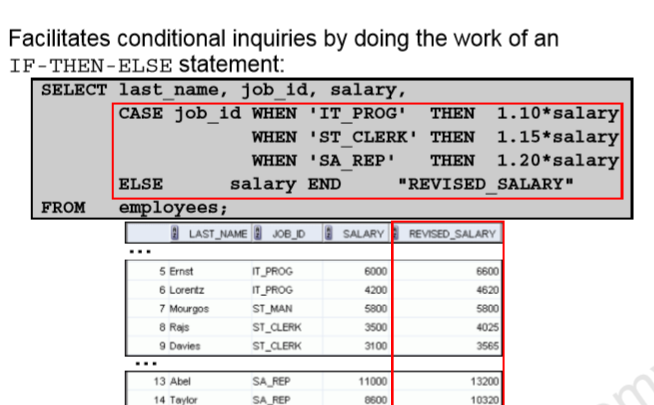
CONDITIONAL EXPRESSIONS



CASE applies to ANSI standard

DECODE is Oracle syntax (from an earlier period)

CASE



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

-- not using as I prefer case, but you could read it

PLEASE READ