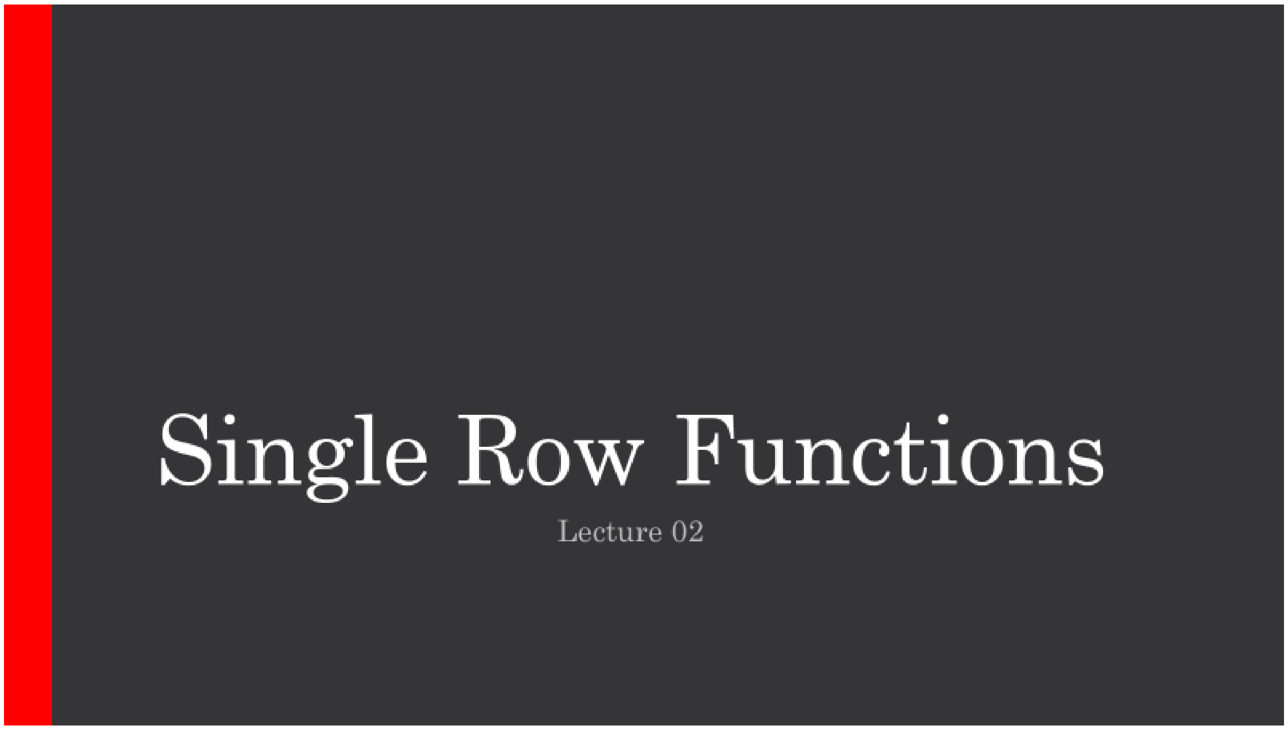
Les02 Using Single Row Functions



Single Row Functions

Lecture 02

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

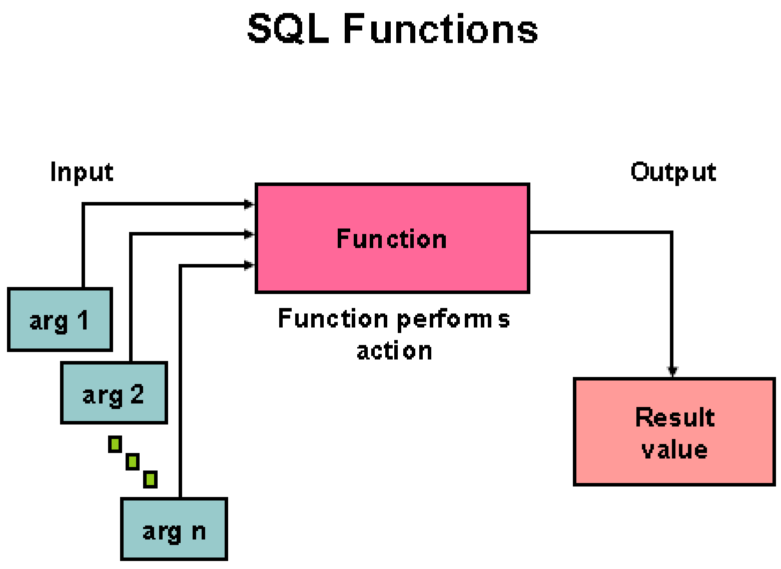
- Describe the use of conversion functions

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



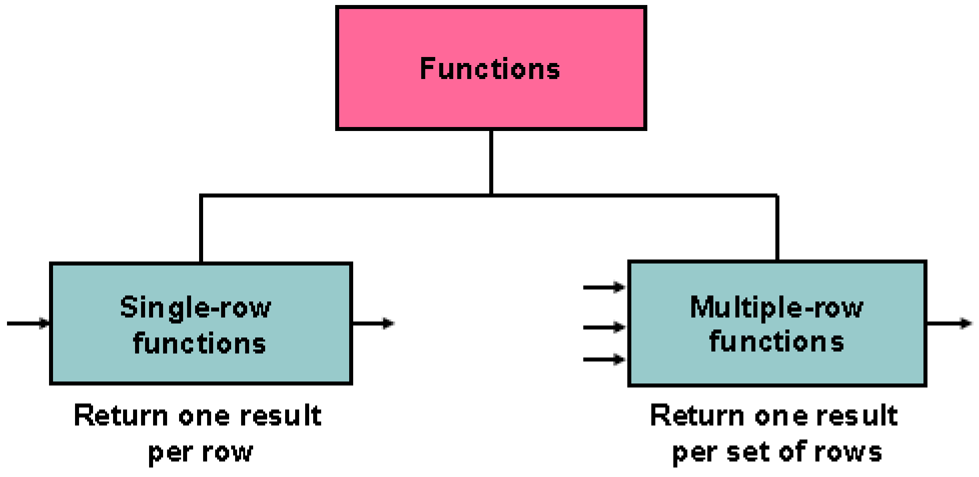
SQL functions
Functions are very powerful feature of SQL. They can be used to do the following:

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 <u>sometimes</u> take arguments and <u>always return a value</u>

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

Two Types of SQL Functions



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

Single-row functions:

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

```
function_name [(arg1, arg2,...)]
```

Single-Row functions

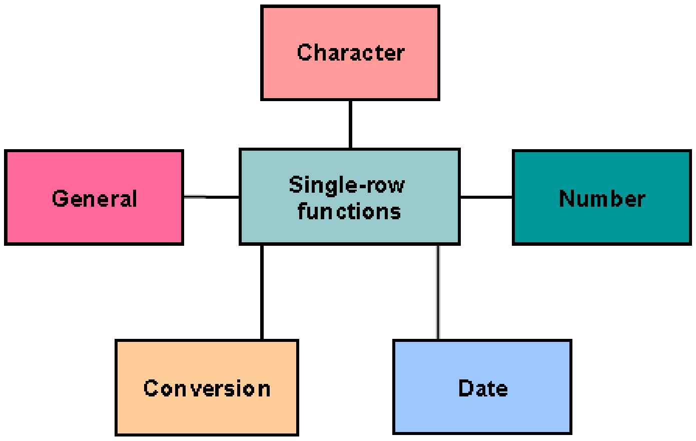
These functions manipulate data items.
Be a set to one or more arguments and return a single value for each row that is retrieved by the query.
An argument can be one of the following:

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

Single-Row Functions



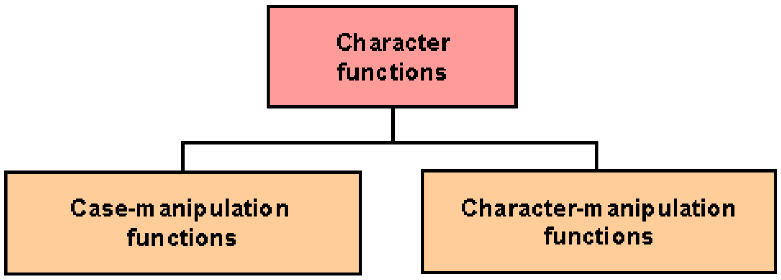
Only the following are covered in this chapter

Single-Row Functions (continued)

This lesson covers the following single-row functions:

- Character functions: ccept character input and can return both character and number values
- Number functions: Accept numeric input and return numeric values
- Date functions: Operate on values of the DATE data type (All date functions return a value of DATE data type except the MONTHS_BETWEEN function, which returns a number.)
- Conversion functions: Convert a value from one data type to another
- General functions:
  - NVL
  - NVL2
  - NULLIF
  - COALESCE
  - CASE
  - DECODE

Character Functions



Case-manipulation functions	Character-manipulation functions
LOWER	CONCAT
UPPER	SUBSTR
INITCAP	LENGTH
	INSTR
	LPAD | RPAD
	TRIM
	REPLACE

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

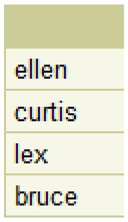
Ellen****
Curtis***

Case-Manipulation Functions

These functions convert case for character strings:

Function	Result
`LOWER('SQL Course')`	`sql course`
`UPPER('SQL Course')`	`SQL COURSE`
`INITCAP('SQL Course')`	`Sql Course`

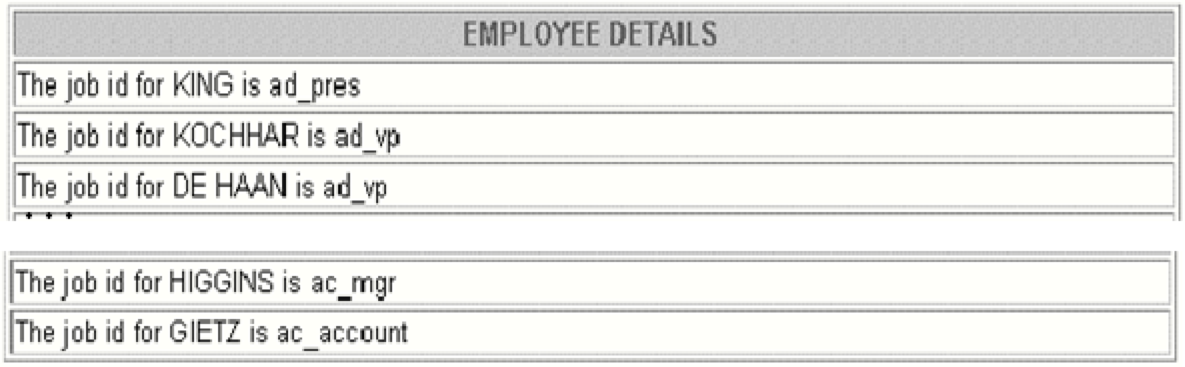
```
SELECT   LOWER (first_name)
FROM     employees
```



ellen
curtis
lex
bruce

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

```
SELECT 'The job id for '||UPPER(last_name)||' is '
          ||LOWER(job_id) AS "EMPLOYEE DETAILS"
FROM   employees;
```



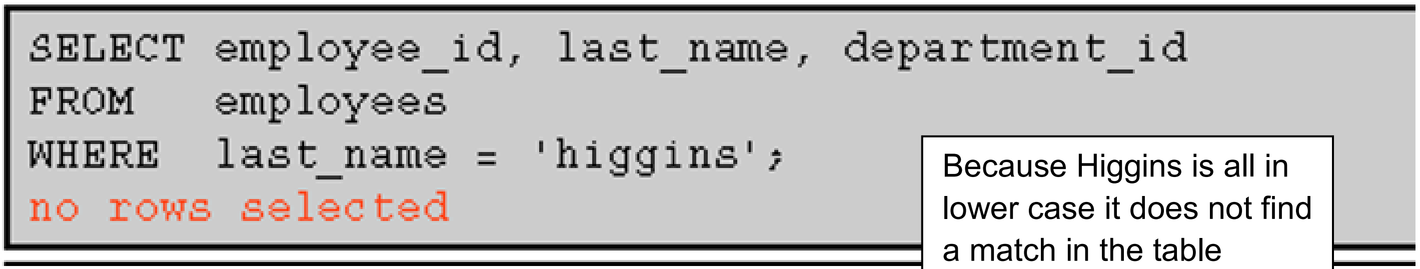
EMPLOYEE DETAILS
The job id for KING is ad_pres
The job id for KOCHHAR is ad_vp
The job id for DE HAAN is ad_vp
. . .
The job id for HIGGINS is ac_mgr
The job id for GIETZ is ac_account

3-10

Using Case-Manipulation Functions

Display the employee number, name, and department number for employee Higgins:

==



```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  last_name = 'higgins';
no rows selected
```

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

Improve it

select employee_id, last_name, department_id
from employees
where lower(last_name) = 'higgins'

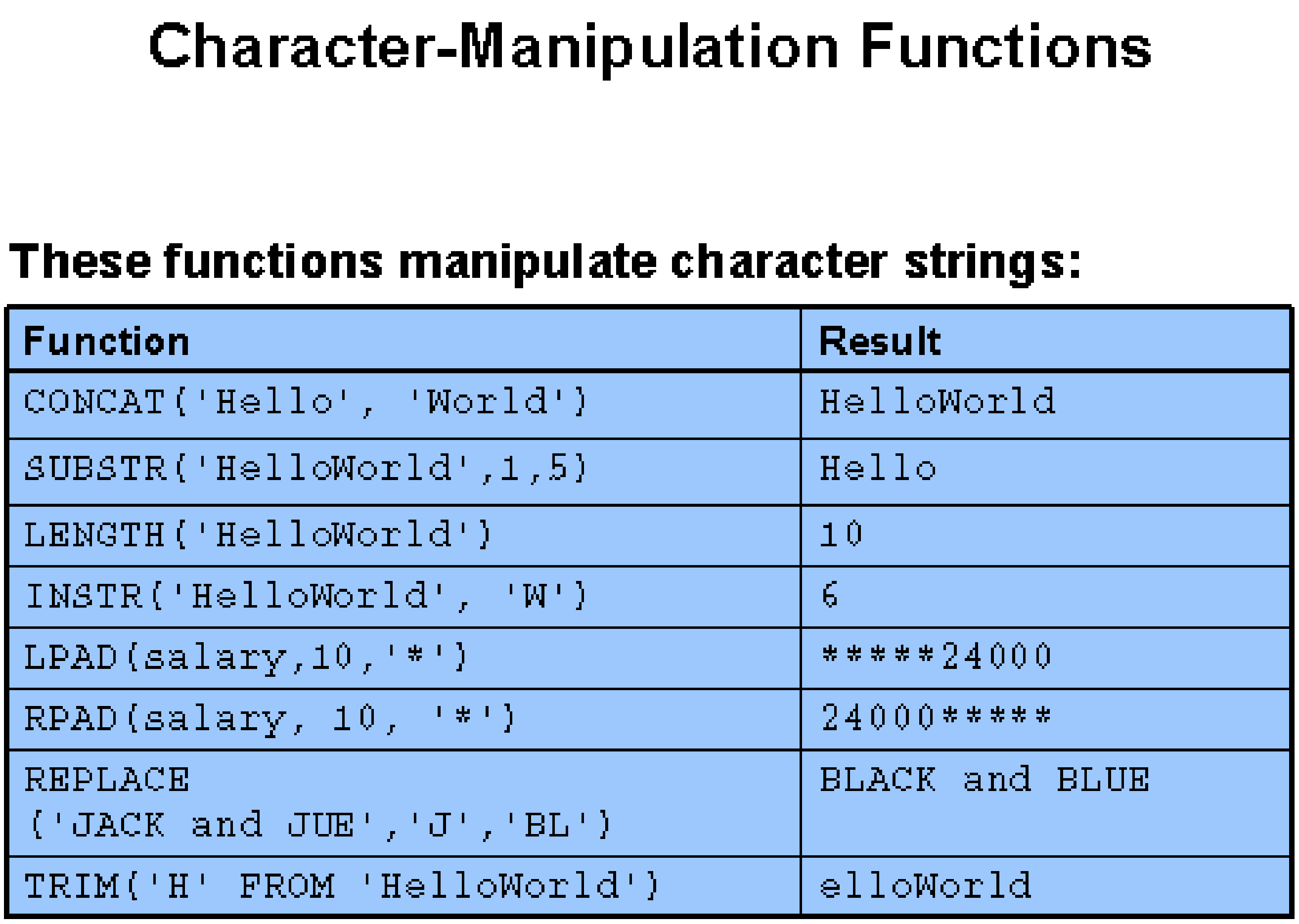
Convert the data stored in the database to LOWER case and match it to the input

Make user enter the data

select employee_id, last_name, department_id
from employees
where **lower**(last_name) = **lower**('&Last_Name')

This would be a substitution variable to allow flexible inputs.
'&last_name'

Case statement on BOTH sides covers all possibilites



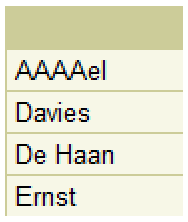
Character-Manipulation Functions

These functions manipulate character strings:

Function	Result
`CONCAT('Hello', 'World')`	`HelloWorld`
`SUBSTR('HelloWorld',1,5)`	`Hello`
`LENGTH('HelloWorld')`	`10`
`INSTR('HelloWorld', 'W')`	`6`
`LPAD(salary,10,'*')`	`*****24000`
`RPAD(salary, 10, '*')`	`24000*****`
`REPLACE ('JACK and JUE','J','BL')`	`BLACK and BLUE`
`TRIM('H' FROM 'HelloWorld')`	`elloWorld`

Demonstrate REPLACE:

```
SELECT   REPLACE (last_name, 'Ab', 'AAAA')
FROM   employees
```



AAAAel
Davies
De Haan
Ernst

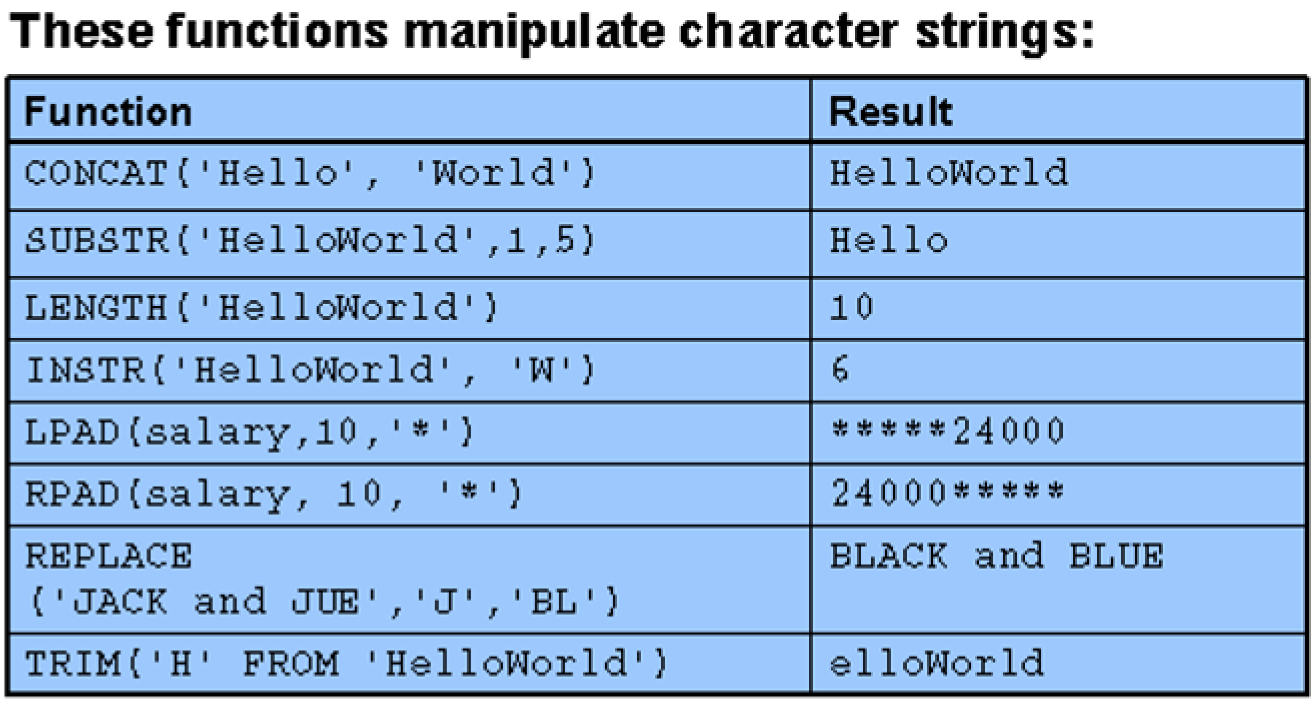
Note: You can use functions such as UPPER and LOWER with ampersand substitution. For example, use UPPER ('&job_title') so that the user does not have to enter the job title in a specific case.

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

==
Just a copy of a previous page to help you answer the above



These functions manipulate character strings:

Function	Result
`CONCAT('Hello', 'World')`	`HelloWorld`
`SUBSTR('HelloWorld',1,5)`	`Hello`
`LENGTH('HelloWorld')`	`10`
`INSTR('HelloWorld', 'W')`	`6`
`LPAD(salary,10,'*')`	`*****24000`
`RPAD(salary, 10, '*')`	`24000*****`
`REPLACE ('JACK and JUE','J','BL')`	`BLACK and BLUE`
`TRIM('H' FROM 'HelloWorld')`	`elloWorld`

```
SELECT
  employee_id,
  CONCAT (first_name, last_name) NAME,
  job_id,
  LENGTH (last_name),
  INSTR (last_name, 'a') "contains an 'a'"
FROM employees
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
          1 BjornFlertjan                                 SA_REP                     8               7
          3 GusGrovlin                                    SA_REP                     7               0
          4 BillSmertal                                   SA_REP                     7               6
          5 DaveMustaine                                  SA_REP                     8               5
          6 HenryHarvey                                   SA_REP                     6               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:

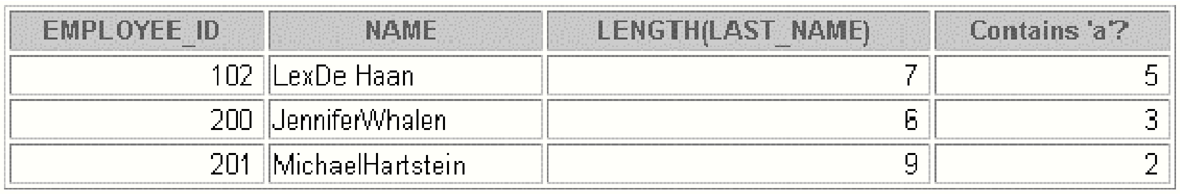
Modify the previous SQL statement to display the data for those employees whose last names *end with the letter a.*

```
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           employees
WHERE         SUBSTR(last_name, -1, 1) = 'a'; ← this gets those with last _name ending in a
```

```
EMPLOYEE_ID NAME                                          JOB_ID     LENGTH(LAST_NAME) contaisn an 'a'
----------- --------------------------------------------- ---------- ----------------- ---------------
         41 InigoMontoya                                  SA_REP                     7               7
```

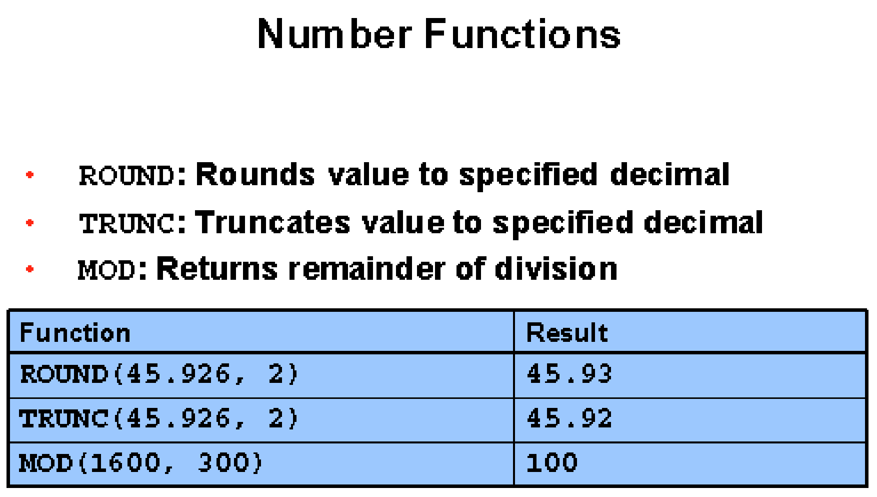
CHANGE SQL to search for last letter AN n

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



EMPLOYEE_ID	NAME	LENGTH(LAST_NAME)	Contains 'a'?
102	LexDe Haan	7	5
200	JenniferWhalen	6	3
201	MichaelHartstein	9	2

3-13



Number Functions

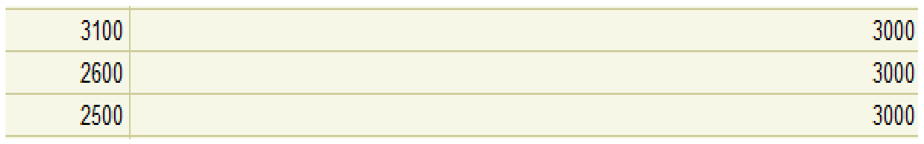
- `ROUND`: **Rounds value to specified decimal**
- `TRUNC`: **Truncates value to specified decimal**
- `MOD`: **Returns remainder of division**

Function	Result
`ROUND(45.926, 2)`	`45.93`
`TRUNC(45.926, 2)`	`45.92`
`MOD(1600, 300)`	`100`

This is a straightforward example

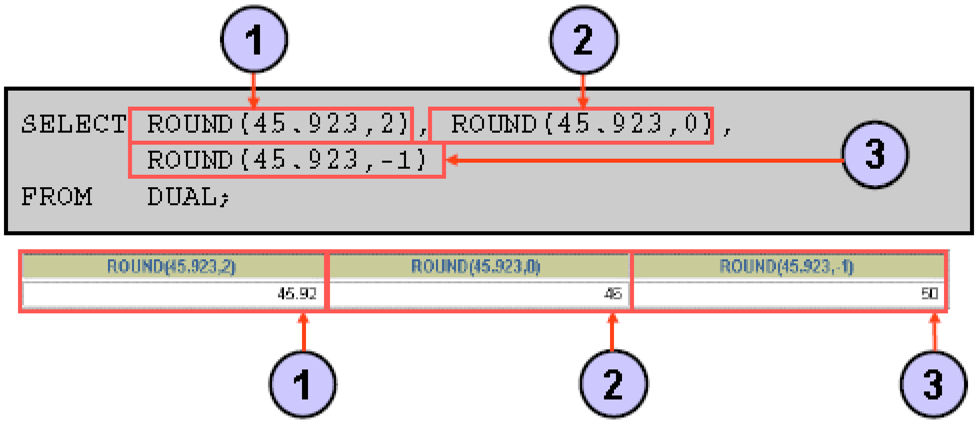
Try this

```
SELECT      salary,                 -- the original salary in table
            round (salary, -3)      -- the same salary rounded
FROM   employees
```



```
3100                                              3000
2600                                              3000
2500                                              3000
```

Using the ROUND Function



```
SELECT ROUND(45.923,2), ROUND(45.923,0),
       ROUND(45.923,-1)
FROM   DUAL;
```

ROUND(45.923,2)	ROUND(45.923,0)	ROUND(45.923,-1)
45.92	46	50

DUAL is a dummy table that you can use to view results from functions and calculations.

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

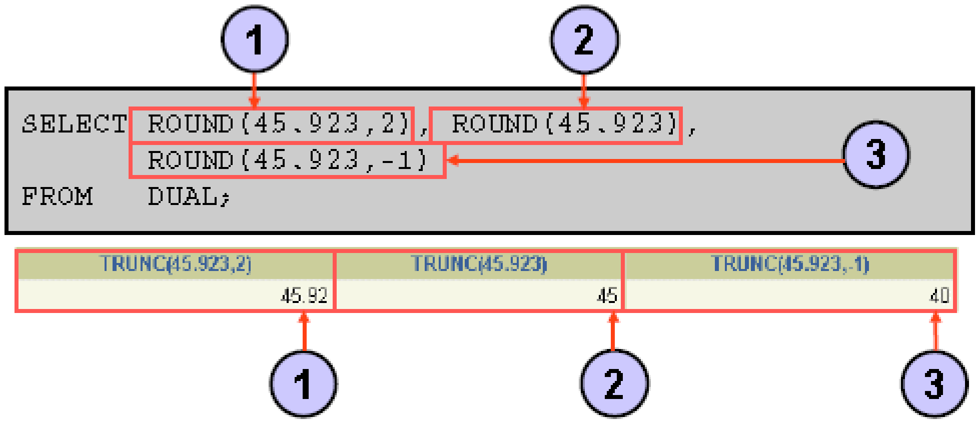
If use 0 or no value it is rounded to zero decimal places

SELECT   salary * 1.3 +23.456, round (salary *1.3+23.456) -- rounding to whole dollars
FROM   employees

	ROUND(SALARY*1.3+23.456)
31223.456	31223
22123.456	22123
22123.456	22123

```
13673.456                    13673
 14323.456                    14323
 11203.456                    11203
  9123.456                     9123
```

Using the TRUNC Function

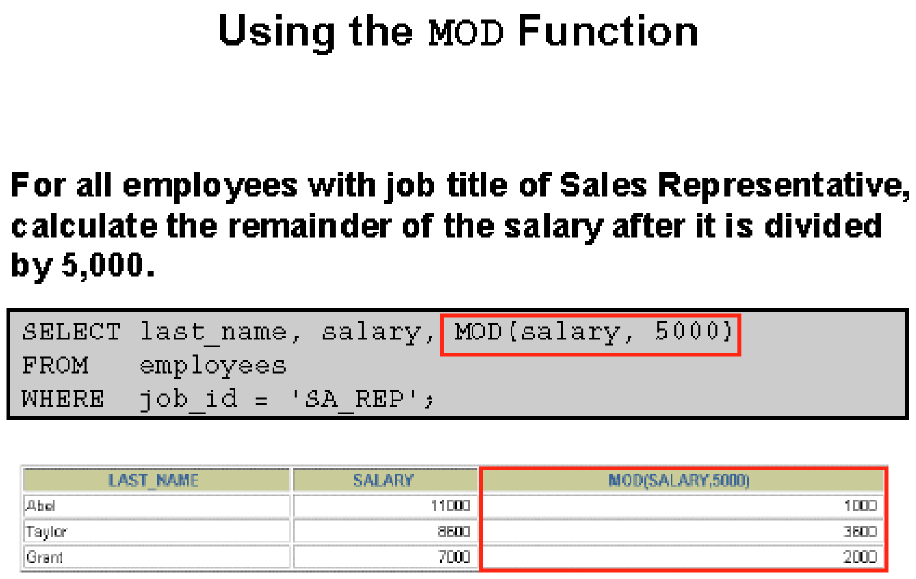


```
SELECT ROUND(45.923,2), ROUND(45.923),
       ROUND(45.923,-1)
FROM   DUAL;
```

TRUNC(45.923,2)	TRUNC(45.923)	TRUNC(45.923,-1)
45.92	45	40

Works the same as ROUND

3-16



Using the MOD Function

For all employees with job title of Sales Representative, calculate the remainder of the salary after it is divided by 5,000.

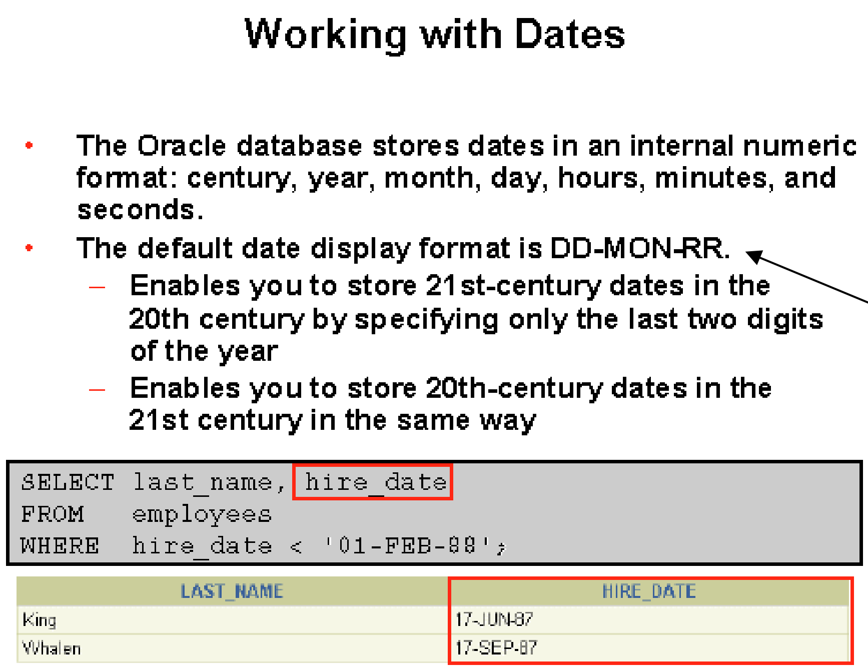
```
SELECT last_name, salary, MOD(salary, 5000)
FROM   employees
WHERE  job_id = 'SA_REP';
```

LAST_NAME	SALARY	MOD(SALARY,5000)
Abel	11000	1000
Taylor	8600	3600
Grant	7000	2000

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**

3-17 **DATES**



Working with Dates

- The Oracle database stores dates in an internal numeric format: century, year, month, day, hours, minutes, and seconds.
- The default date display format is DD-MON-RR.
  - Enables you to store 21st-century dates in the 20th century by specifying only the last two digits of the year
  - Enables you to store 20th-century dates in the 21st century in the same way

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date < '01-FEB-88';
```

LAST_NAME	HIRE_DATE
King	17-JUN-87
Whalen	17-SEP-87

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
19 87 06 17 17 10 43

Note: century or year stored as 4 digits even if displayed as 2
3-19

Working with Dates

SYSDATE is a function that returns:

- **Date**
- **Time**

TEST IT

```
SELECT   SYSDATE
FROM     DUAL
```

```
SYSDATE
---------
17-SEP-18
```

Arithmetic with Dates

- **Add or subtract a number to or from a date for a resultant date value.**
- **Subtract two dates to find the number of days between those dates.**
- **Add hours to a date by dividing the number of hours by 24.**

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

If payment is due in 30 days

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

```
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    employees
WHERE   department_id = 90;
```

This answer is not very good …. Improve it

```
SELECT  last_name, trunc((sysdate - hire_date)/7, 2) "Weeks Employed"
FROM    employees
WHERE   department_id = 90;
```

```
LAST_NAME                 Weeks Employed
------------------------- --------------
King                              1526.5
Kochhar                          1408.36
De Haan                           1235.5
```

Why does it end in .5 ?

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

Date Functions

Function	Result
MONTHS_BETWEEN	Number of months between two dates
ADD_MONTHS	Add calendar months to date
NEXT_DAY	Next day of the date specified
LAST_DAY	Last day of the month
ROUND	Round date
TRUNC	Truncate date

EX: next page

Date Functions

Date functions operate on Oracle dates. All date functions return a value of DATE data type except MONTHS_BETWEEN, which returns a numeric value.

- MONTHS_BETWEEN(date1, date2): Finds the number of months between *date1 and date2. The result can be positive or negative. If date1 is later than date2, the result is positive; if date1 is earlier than date2, the result is negative. The noninteger part of the result represents a portion of the month.*
- ADD_MONTHS(date, n): Adds n number of calendar months to date. The value of n must be an integer and can be negative.
- NEXT_DAY(date, 'char'): Finds the date of the next specified day of the week ('char') following date. The value of char may be a number representing a day or a character string.
- LAST_DAY(date): Finds the date of the last day of the month that contains date
- ROUND(date[,'fmt']): Returns date rounded to the unit that is specified by the format model fmt. If the format model fmt is omitted, date is rounded to the nearest day.
- TRUNC(date[, 'fmt']): Returns date with the time portion of the day truncated to the unit that is specified by the format model fmt. If the format model *fmt is omitted, date is truncated to the nearest day.*

This list is a subset of the available date functions.
The format models are covered later in this lesson.
Examples of format models are month and year.

Using Date Functions

Function	Result
MONTHS_BETWEEN ('01-SEP-95','11-JAN-94')	19.6774194
ADD_MONTHS ('11-JAN-94',6)	'11-JUL-94'
NEXT_DAY ('01-SEP-95','FRIDAY')	'08-SEP-95'
LAST_DAY ('01-FEB-95')	'28-FEB-95'

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.

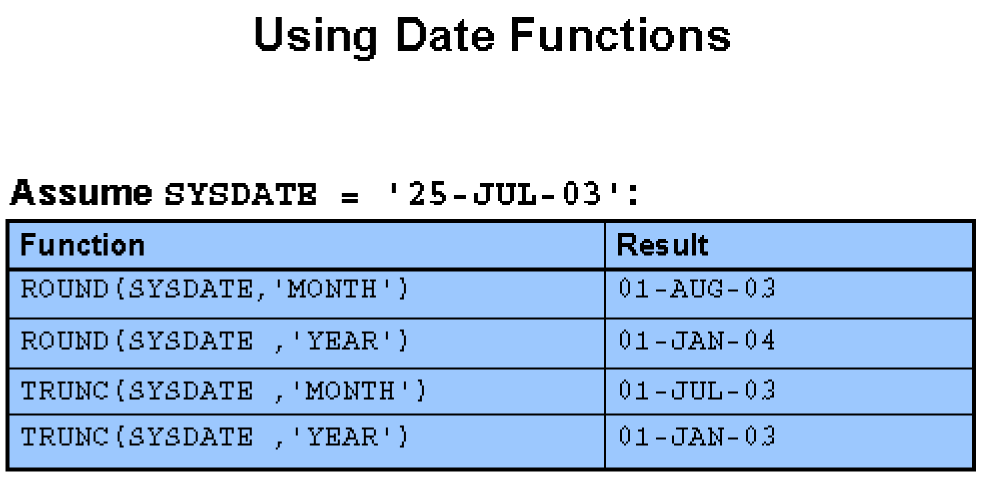
ANS: Next page

```
SELECT employee_id,
      hire_date,
      MONTHS_BETWEEN (SYSDATE, hire_date) "Seniority",
      ADD_MONTHS (hire_date, 6) "Review Date",
      NEXT_DAY (hire_date, 'Friday'),
      LAST_DAY (hire_date)
FROM employees
WHERE MONTHS_BETWEEN (SYSDATE, hire_date) > 70;
```

```
EMPLOYEE_ID HIRE_DATE  Seniority Review Date NEXT_DAY(HIRE_DATE,'FRIDAY') LAST_DAY(HIRE_DATE)
----------- --------- ---------- ----------- ---------------------------- -------------------
        100 17-JUN-87 359.0790647 17-DEC-87   19-JUN-87                    30-JUN-87
        101 21-SEP-89 331.9500325 21-MAR-90   22-SEP-89                    30-SEP-89
        102 13-JAN-93 292.208097 13-JUL-93   15-JAN-93                    31-JAN-93
        103 03-JAN-90 328.5306776 03-JUL-90   05-JAN-90                    31-JAN-90
        104 21-MAY-91 311.9500325 21-NOV-91   24-MAY-91                    31-MAY-91
        107 07-FEB-99 219.4016454 07-AUG-99   12-FEB-99                    28-FEB-99
        124 16-NOV-99 210.1113228 16-MAY-00   19-NOV-99                    30-NOV-99
        141 17-OCT-95 259.0790647 17-APR-96   20-OCT-95                    31-OCT-95
        142 29-JAN-97 243.691968 29-JUL-97   31-JAN-97                    31-JAN-97
        143 15-MAR-98 230.1435809 15-SEP-98   20-MAR-98                    31-MAR-98
        144 09-JUL-98 226.3371293 09-JAN-99   10-JUL-98                    31-JUL-98
        149 29-JAN-00 207.691968 29-JUL-00   04-FEB-00                    31-JAN-00
        174 11-MAY-96 252.2726131 11-NOV-96   17-MAY-96                    31-MAY-96
        176 24-MAR-98 229.8532583 24-SEP-98   27-MAR-98                    31-MAR-98
        178 24-MAY-99 215.8532583 24-NOV-99   28-MAY-99                    31-MAY-99
        200 17-SEP-87 356.0790647 17-MAR-88   18-SEP-87                    30-SEP-87
        201 17-FEB-96 255.0790647 17-AUG-96   23-FEB-96                    29-FEB-96
        202 17-AUG-97 237.0790647 17-FEB-98   22-AUG-97                    31-AUG-97
        205 07-JUN-94 275.4016454 07-DEC-94   10-JUN-94                    30-JUN-94
        206 07-JUN-94 275.4016454 07-DEC-94   10-JUN-94                    30-JUN-94
        207 01-JUL-00 202.5951938 01-JAN-01   07-JUL-00                    31-JUL-00

 21 rows selected --- currently 23 rows
```

3-24



Using Date Functions

Assume `SYSDATE = '25-JUL-03'`:

Function	Result
`ROUND(SYSDATE,'MONTH')`	`01-AUG-03`
`ROUND(SYSDATE ,'YEAR')`	`01-JAN-04`
`TRUNC(SYSDATE ,'MONTH')`	`01-JUL-03`
`TRUNC(SYSDATE ,'YEAR')`	`01-JAN-03`

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.

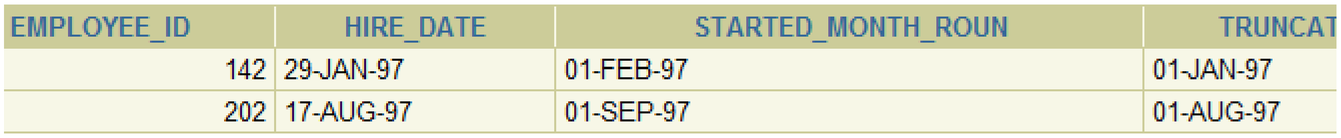
When used with dates, these functions round or truncate to the specified format model. Therefore, you can round dates to the nearest year or month.

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

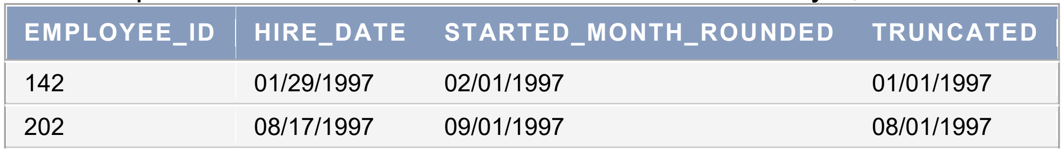
PROBLEM:
Compare the hire dates for all employees who started in 1997. Display the employee number, hire date, and start month using the ROUND and TRUNC functions.

```
SELECT      employee_id,
            hire_date,
            ROUND(hire_date, 'MONTH') as Started_Month_Rounded,
            TRUNC(hire_date, 'MONTH') as Truncated
FROM         employees
WHERE       hire_date LIKE '%97';
```



EMPLOYEE_ID	HIRE_DATE	STARTED_MONTH_ROUN	TRUNCAT
142	29-JAN-97	01-FEB-97	01-JAN-97
202	17-AUG-97	01-SEP-97	01-AUG-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

EXERCISE for you to do at back of Oracle book 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

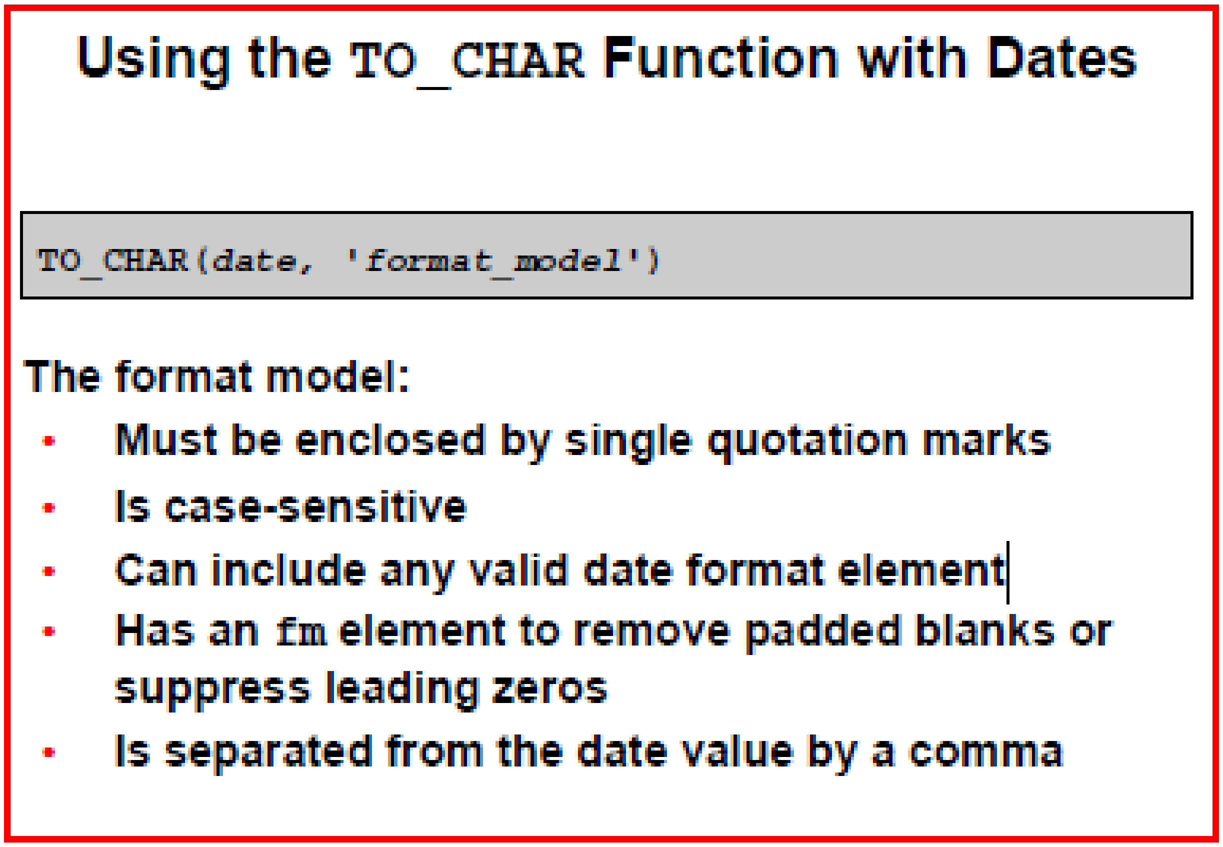
3-28

3-29

3-30

3-31

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



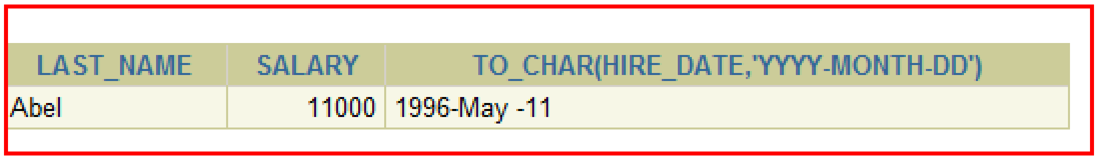
Using the TO_CHAR Function with Dates

```
TO_CHAR(date, 'format_model')
```

The format model:

- Must be enclosed by single quotation marks
- Is case-sensitive
- Can include any valid date format element
- Has an fm element to remove padded blanks or suppress leading zeros
- Is separated from the date value by a comma

```
select last_name, salary,
        TO_CHAR (hire_date, 'YYYY-Month-DD')
from employees
where salary = '11000'
```

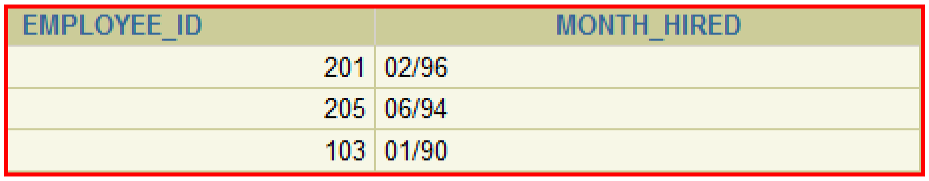


LAST_NAME	SALARY	TO_CHAR(HIRE_DATE,'YYYY-MONTH-DD')
Abel	11000	1996-May -11

CHANGE REQUIREMENT

This example is changing it to MM/YY

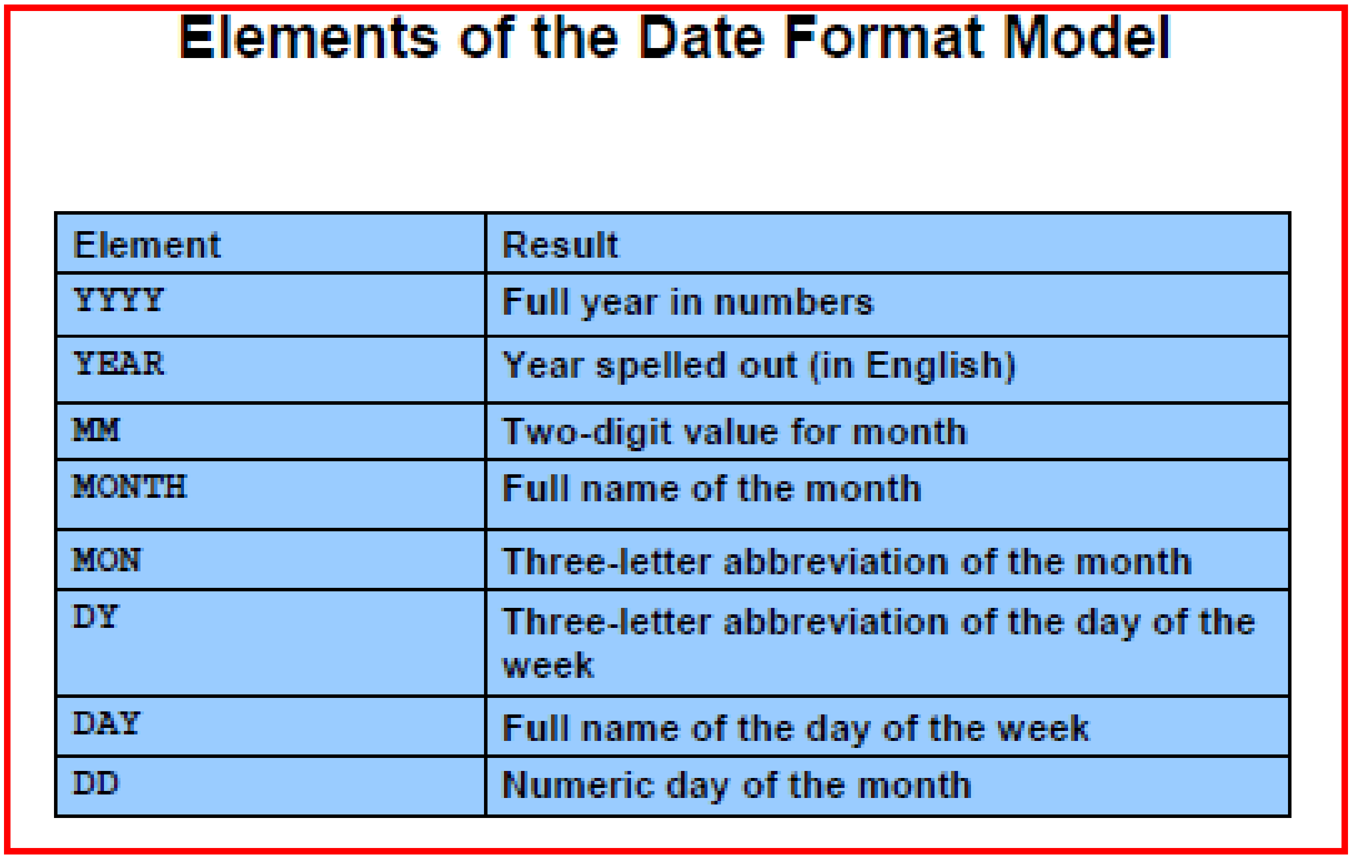
```
SELECT  EMPLOYEE_ID,
        TO_CHAR (HIRE_DATE, 'MM/YY') Month_Hired
FROM    EMPLOYEES
WHERE   LAST_NAME like 'H%'
```



EMPLOYEE_ID	MONTH_HIRED
201	02/96
205	06/94
103	01/90

NOTE: you control the output format

3-33 --4-12



Elements of the Date Format Model

Element	Result
YYYY	Full year in numbers
YEAR	Year spelled out (in English)
MM	Two-digit value for month
MONTH	Full name of the month
MON	Three-letter abbreviation of the month
DY	Three-letter abbreviation of the day of the week
DAY	Full name of the day of the week
DD	Numeric day of the month

Do This

```
SELECT      EMPLOYEE_ID,
            TO_CHAR (HIRE_DATE, 'MM/DD/YY')
FROM        EMPLOYEES
WHERE       LAST_NAME like 'H%'
```

Then this

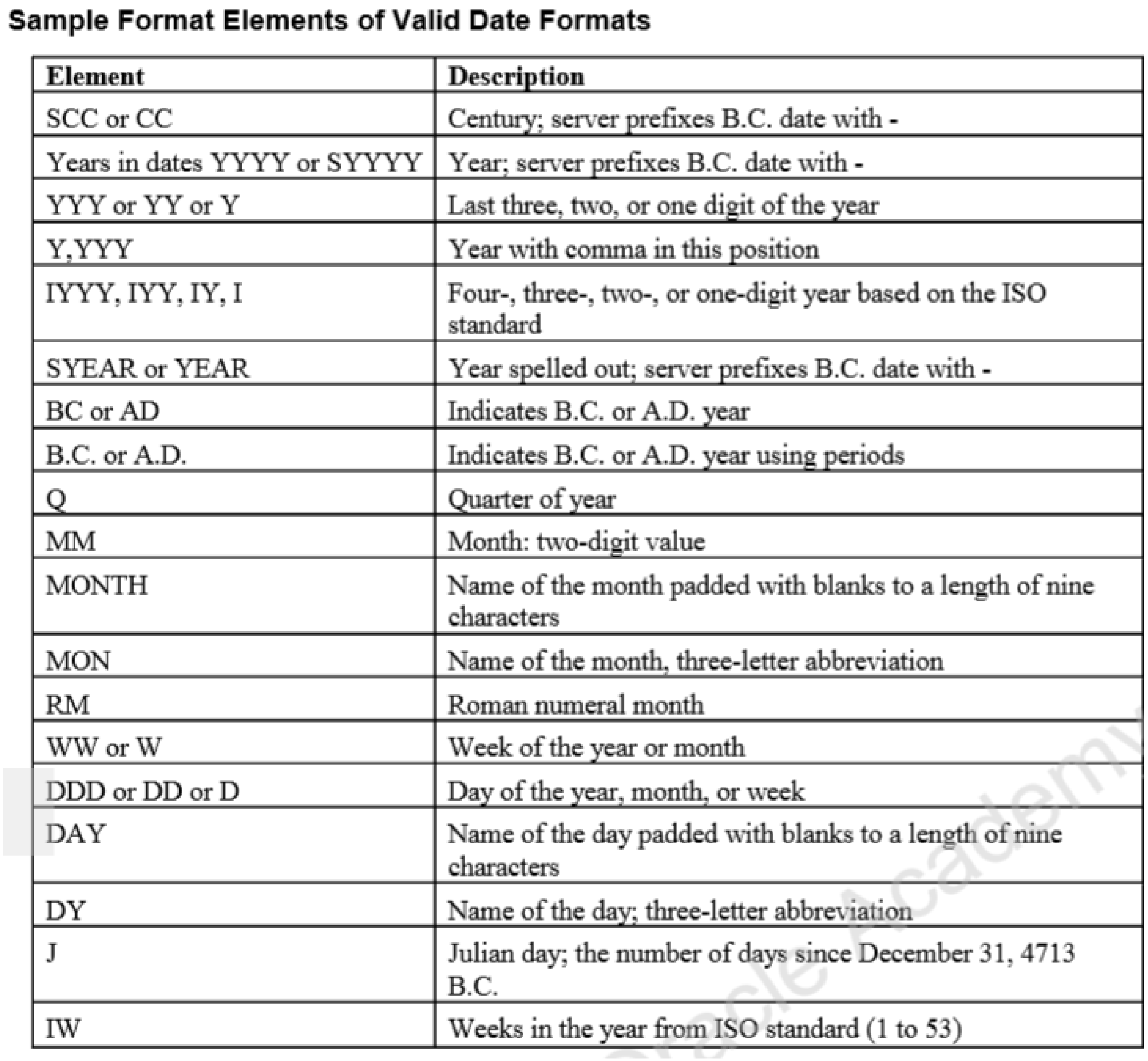
```
SELECT      EMPLOYEE_ID,
            TO_CHAR (HIRE_DATE, 'fmMM/DD/YY')HireDate
FROM        EMPLOYEES
WHERE       LAST_NAME like 'H%'
```

Try fm with a lot of spaces

CAN YOU SEE THE DIFFERENCE

3-34 – 4-13

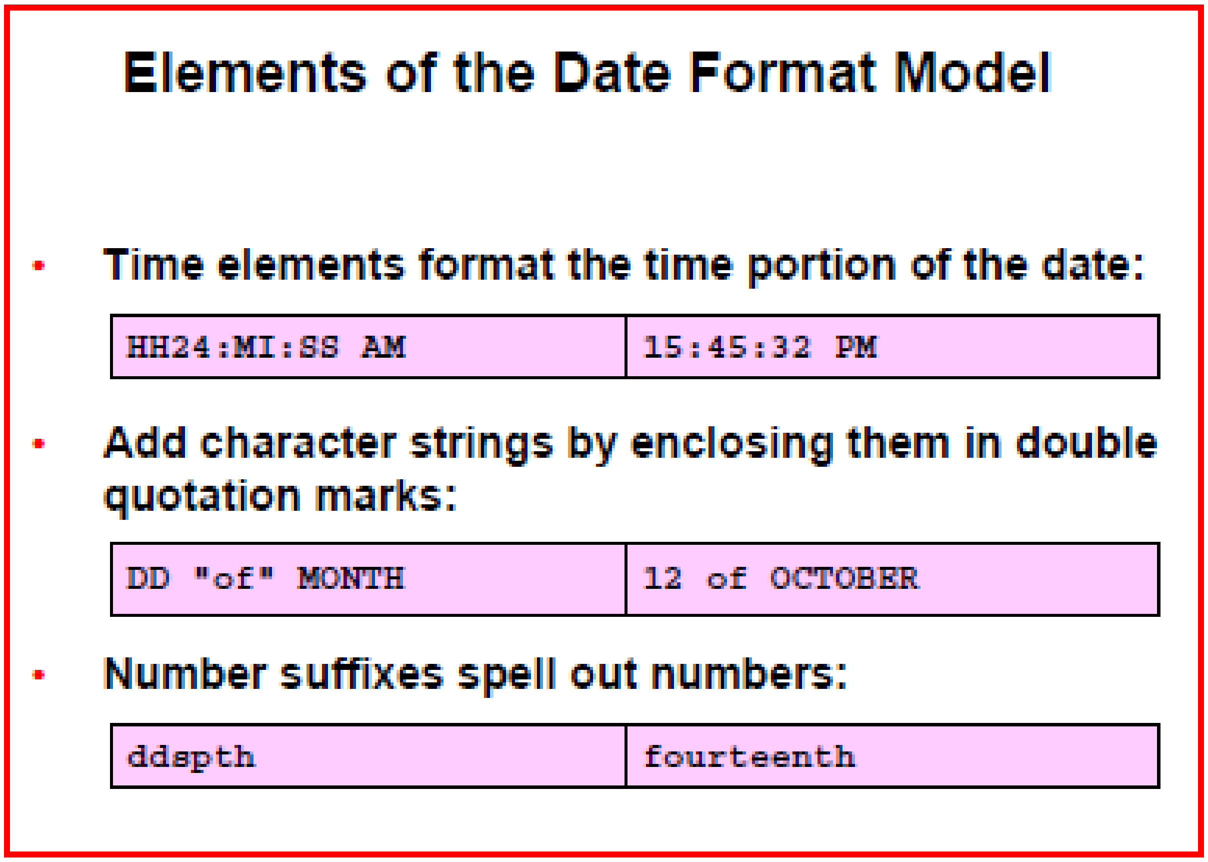
MANY OTHERS



Sample Format Elements of Valid Date Formats

Element	Description
SCC or CC	Century; server prefixes B.C. date with -
Years in dates YYYY or SYYYY	Year; server prefixes B.C. date with -
YYY or YY or Y	Last three, two, or one digit of the year
Y,YYY	Year with comma in this position
IYYY, IYY, IY, I	Four-, three-, two-, or one-digit year based on the ISO standard
SYEAR or YEAR	Year spelled out; server prefixes B.C. date with -
BC or AD	Indicates B.C. or A.D. year
B.C. or A.D.	Indicates B.C. or A.D. year using periods
Q	Quarter of year
MM	Month: two-digit value
MONTH	Name of the month padded with blanks to a length of nine characters
MON	Name of the month, three-letter abbreviation
RM	Roman numeral month
WW or W	Week of the year or month
DDD or DD or D	Day of the year, month, or week
DAY	Name of the day padded with blanks to a length of nine characters
DY	Name of the day; three-letter abbreviation
J	Julian day; the number of days since December 31, 4713 B.C.
IW	Weeks in the year from ISO standard (1 to 53)

Try out some of them to see what they do



Elements of the Date Format Model

- Time elements format the time portion of the date:

`HH24:MI:SS AM`	`15:45:32 PM`

- Add character strings by enclosing them in double quotation marks:

`DD "of" MONTH`	`12 of OCTOBER`

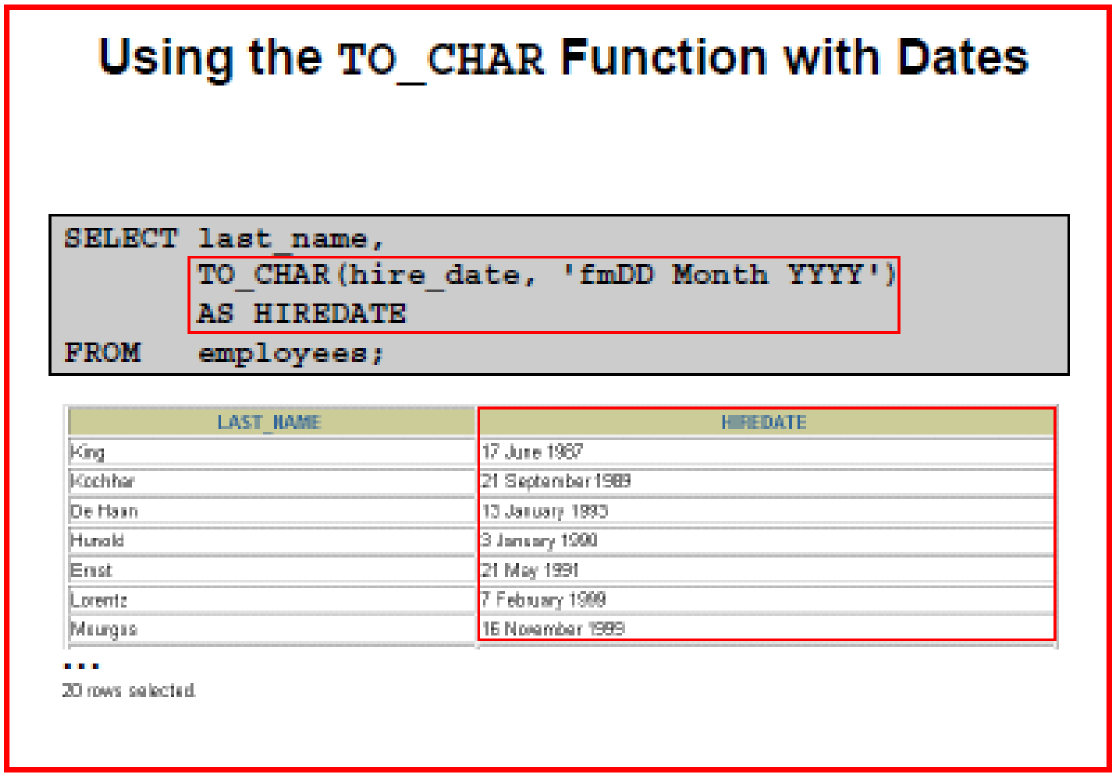
- Number suffixes spell out numbers:

`ddspth`	`fourteenth`

Again another set of formats

REMEMBER:

Business uses dates



Using the TO_CHAR Function with Dates

```
SELECT last_name,
       TO_CHAR(hire_date, 'fmDD Month YYYY')
       AS HIREDATE
FROM   employees;
```

LAST_NAME	HIREDATE
King	17 June 1987
Kochhar	21 September 1989
De Haan	13 January 1993
Hunold	3 January 1990
Ernst	21 May 1991
Lorentz	7 February 1999
Mourgos	16 November 1999

...

20 rows selected.

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      employees
```

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

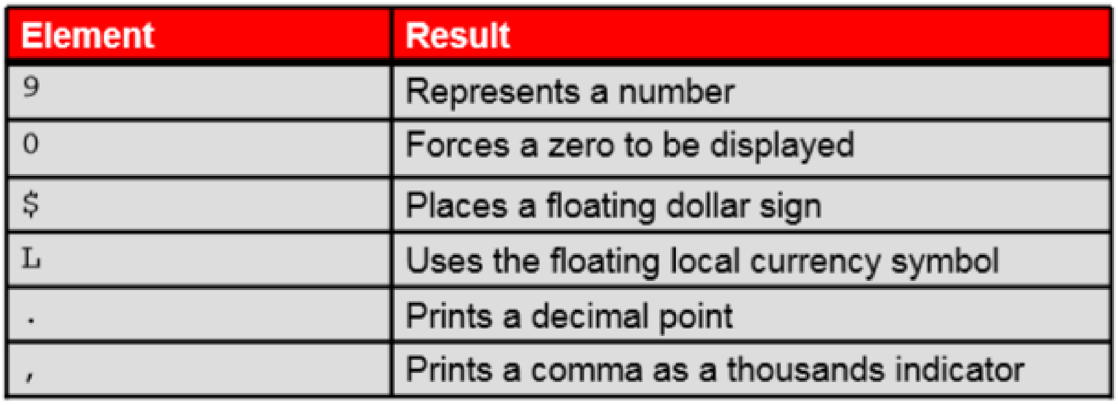
LAST_NAME	TO_CHAR(HIRE_DATE,'FMDDSPTH"OF"MONTHYYYYFMHH:MI')
King	Seventeenth of June 1987 12:00
Kochhar	Twenty-First of September 1989 12:00
De Haan	Thirteenth of January 1993 12:00
Hunold	Third of January 1990 12:00
Ernst	Twenty-First of May 1991 12:00
Lorentz	Seventh of February 1999 12:00
Mourgos	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

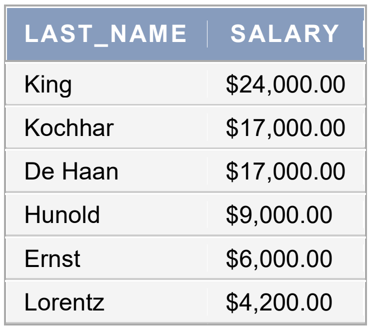
```
TO_CHAR(number, 'format_model')
```

These are some of the format elements that you can use with the `TO_CHAR` function to display a number value as a character:



Element	Result
9	Represents a number
0	Forces a zero to be displayed
$	Places a floating dollar sign
L	Uses the floating local currency symbol
.	Prints a decimal point
,	Prints a comma as a thousands indicator

```
SELECT    last_name,
          TO_CHAR(salary, '$99,999.00') as SALARY
FROM      employees;
```



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

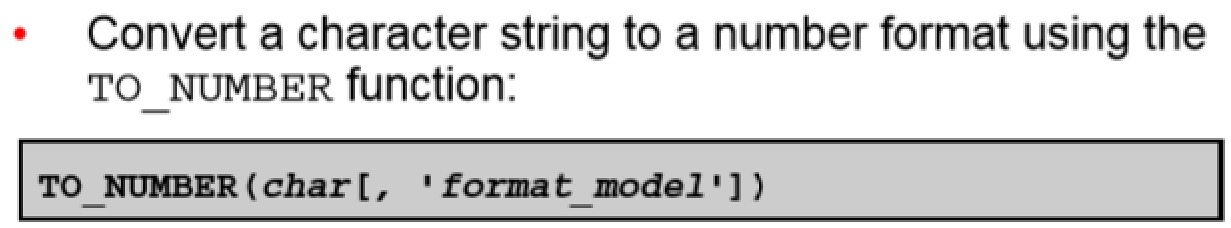
Problems of a floating dollar sign is that the field is left justified as a character field and numbers don't align well.

AGAIN SQL wasn't meant to be fancy.
BUT right justifies on other software

Convert character string to NUMBER or DATE

03-39--4-20

General format of conver to a number



- Convert a character string to a number format using the `TO_NUMBER` function:

```
TO_NUMBER(char[, 'format_model'])
```

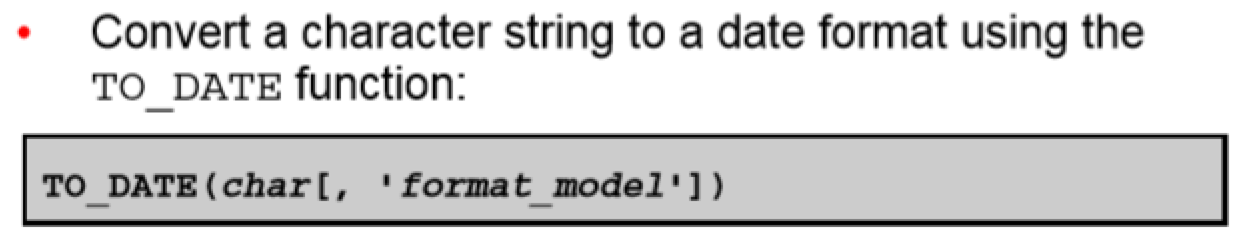
```
SELECT        to_number('1234')-2  -- convert STRING of characters to a number less 2
from dual;
```

RESUKT:

```
TO_NUMBER('1234')-2
-------------------
             1232
```

Convert a character to a date

4-20



- Convert a character string to a date format using the `TO_DATE` function:

```
TO_DATE(char[, 'format_model'])
```

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. Is it correct?

```
LAST_NAME                 TO_CHAR(HIRE_DATE,'DD-MON-YYYY')
------------------------- --------------------------------
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
```

Wrong results because it assumed with YY that it was 2090

Change it to RR

TRY THIS (fx means eXact formatting)

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');
```

It is selecting an employee with a specific hire date. The test for equal would not work unless the formats matched. Notice there are spaces between May and 24.

NOTE: 1 Repeat the code above, add some extra spaces in the date

2 Add some spaces in the format and rerun

Adding equal number of spaces in both

```
SELECT      last_name, hire_date
from        employees
where       hire_date = to_date('May    24, 1999', 'fxMonth    DD, YYYY');
```

RESULT: where are spaces. Using spaces to ensure matching types, but output said just to show name and hire date. What you see is hire date.

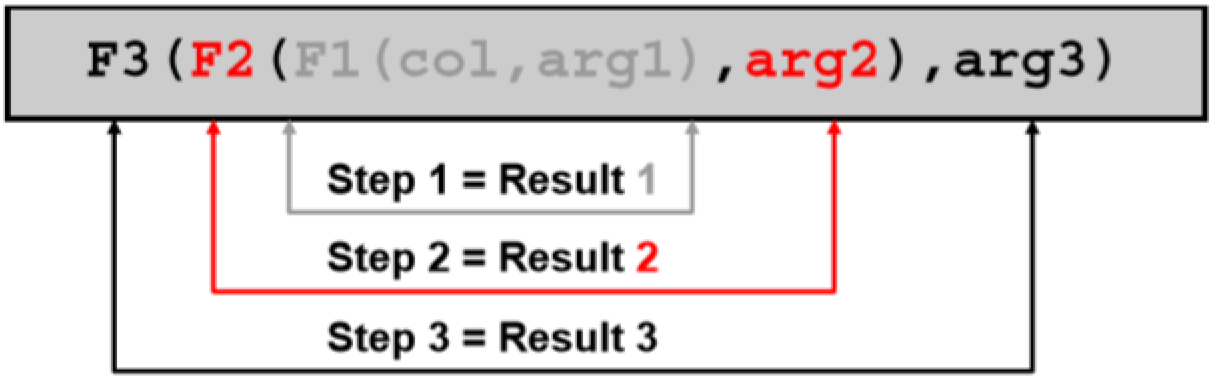
```
LAST_NAME                 HIRE_DATE
------------------------- ---------
Grant                     24-MAY-99
```

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

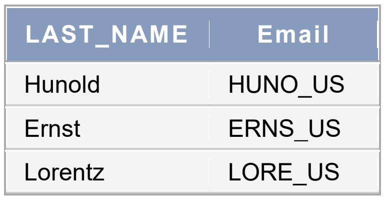
4-25

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 much nicer looking

Example Higgins becomes HIGG_US

```
SELECT      last_name,
            UPPER (CONCAT(SUBSTR(LAST_NAME, 1, 4) , '_US')) as "Email"
FROM        EMPLOYEES
WHERE        DEPARTMENT_ID = 60;
```



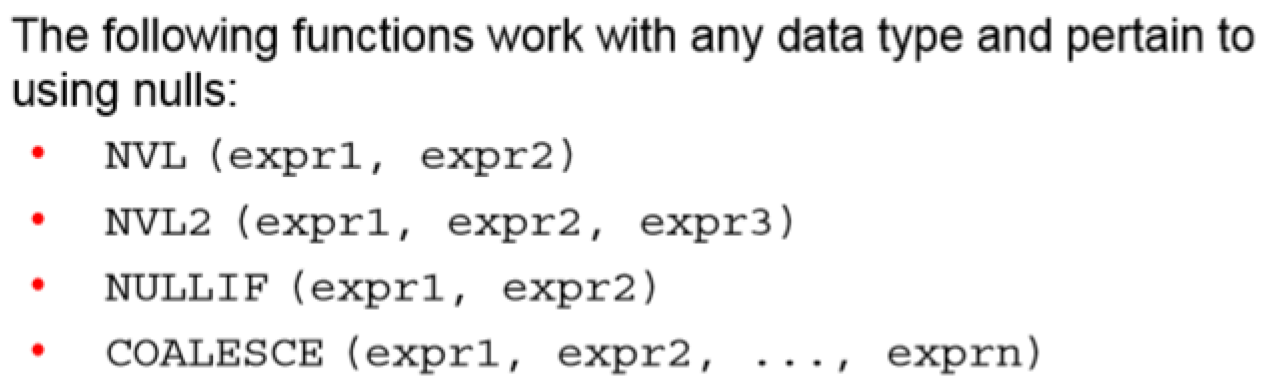
LAST_NAME	Email
Hunold	HUNO_US
Ernst	ERNS_US
Lorentz	LORE_US

General Functions

04-27

Handling NULLS

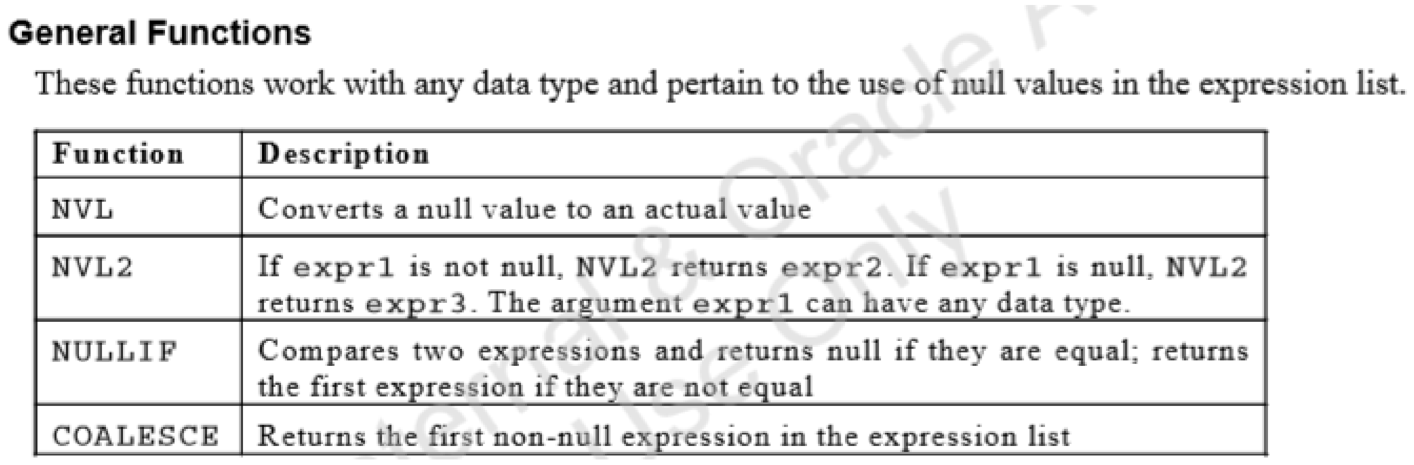
General Format



The following functions work with any data type and pertain to using nulls:

- `NVL (expr1, expr2)`
- `NVL2 (expr1, expr2, expr3)`
- `NULLIF (expr1, expr2)`
- `COALESCE (expr1, expr2, ..., exprn)`

The most used is NVL



General Functions

These functions work with any data type and pertain to the use of null values in the expression list.

Function	Description
NVL	Converts a null value to an actual value
NVL2	If `expr1` is not null, NVL2 returns `expr2`. If `expr1` is null, NVL2 returns `expr3`. The argument `expr1` can have any data type.
NULLIF	Compares two expressions and returns null if they are equal; returns the first expression if they are not equal
COALESCE	Returns the first non-null expression in the expression list

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 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        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')   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        employees;
```

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

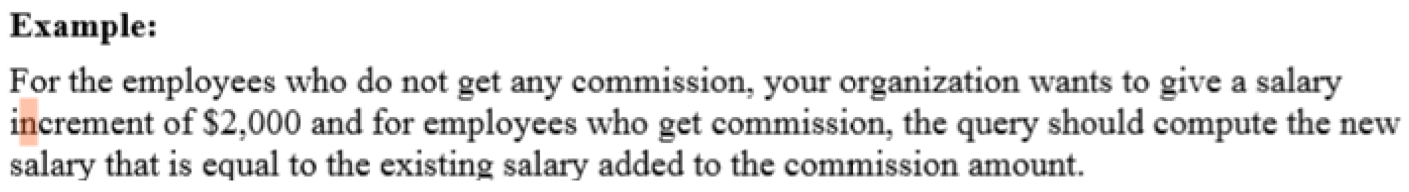
First needed to convert numeric field to a charater because want to diplay characters (the question mark)

READ the book for the other NULLs

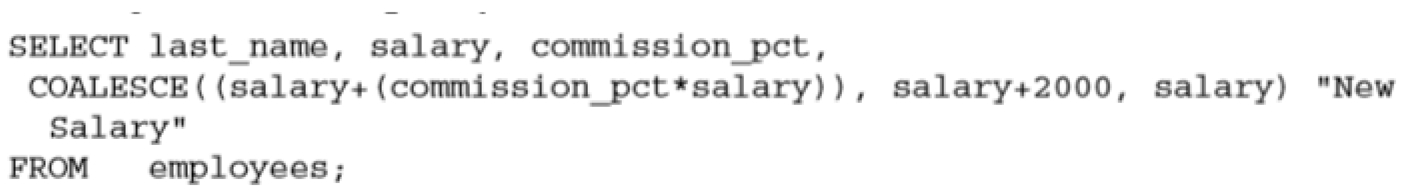
COALESCE

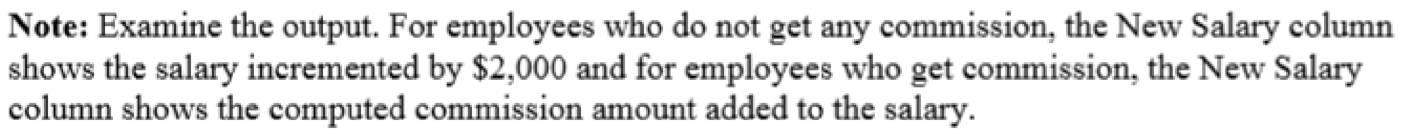
4-32 and 4-33

Evaluates multiple expressions --- read the book

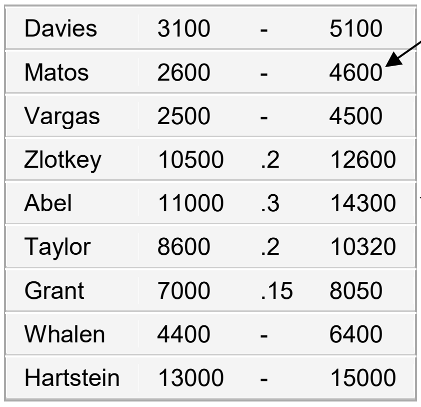

Example:

For the employees who do not get any commission, your organization wants to give a salary increment of $2,000 and for employees who get commission, the query should compute the new salary that is equal to the existing salary added to the commission amount.


```
SELECT last_name, salary, commission_pct,
 COALESCE((salary+(commission_pct*salary)), salary+2000, salary) "New
  Salary"
FROM   employees;
```


Note: Examine the output. For employees who do not get any commission, the New Salary column shows the salary incremented by $2,000 and for employees who get commission, the New Salary column shows the computed commission amount added to the salary.

```
SELECT     last_name, salary, commission_pct,
           coalesce( (salary +(commission_pct*salary)),
           salary + 2000,
           salary) as "New Salary"
FROM       employees;
```



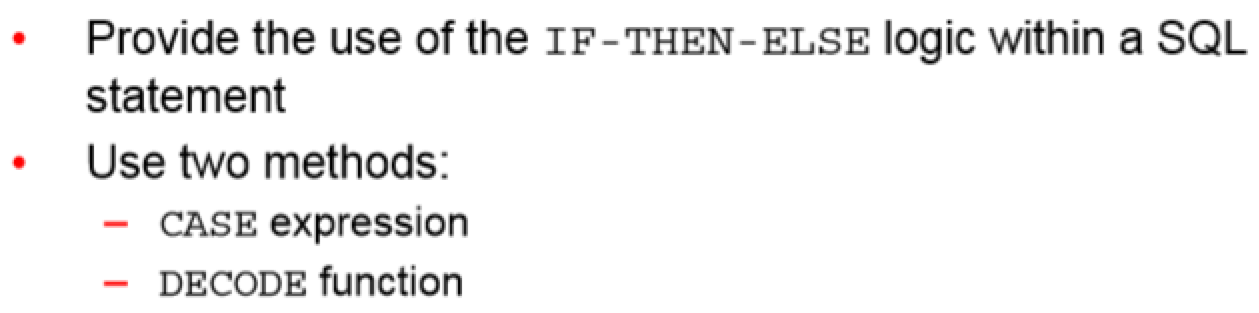
Davies	3100	-	5100
Matos	2600	-	4600
Vargas	2500	-	4500
Zlotkey	10500	.2	12600
Abel	11000	.3	14300
Taylor	8600	.2	10320
Grant	7000	.15	8050
Whalen	4400	-	6400
Hartstein	13000	-	15000

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

CONDITIONAL EXPRESSIONS

4-35

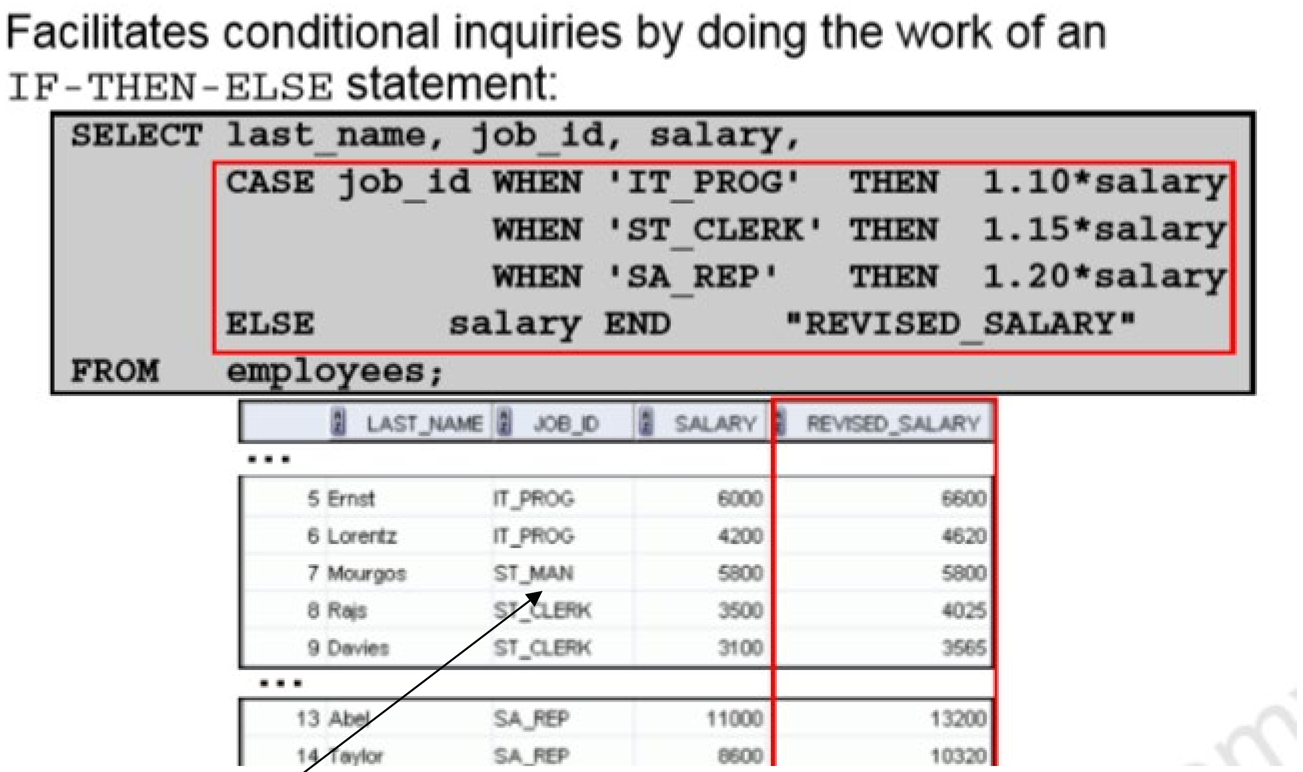


- Provide the use of the `IF-THEN-ELSE` logic within a SQL statement
- Use two methods:
  - `CASE` expression
  - `DECODE` function

CASE applies to ANSI standard
DECODE is Oracle syntax (from an earlier period)

CASE

4-38



Facilitates conditional inquiries by doing the work of an `IF-THEN-ELSE` statement:

```
SELECT last_name, job_id, salary,
       CASE job_id WHEN 'IT_PROG'  THEN  1.10*salary
                   WHEN 'ST_CLERK' THEN  1.15*salary
                   WHEN 'SA_REP'   THEN  1.20*salary
       ELSE      salary END     "REVISED_SALARY"
FROM   employees;
```

	LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...				
5	Ernst	IT_PROG	6000	6600
6	Lorentz	IT_PROG	4200	4620
7	Mourgos	ST_MAN	5800	5800
8	Rajs	ST_CLERK	3500	4025
9	Davies	ST_CLERK	3100	3565
...				
13	Abel	SA_REP	11000	13200
14	Taylor	SA_REP	8600	10320

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

4-39

PLEASE READ