ORACLE

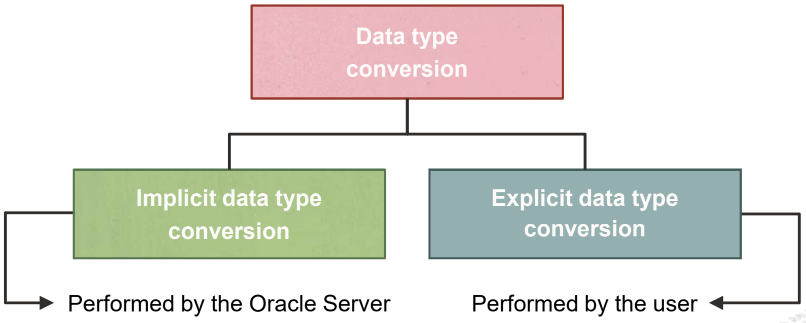
5

Using Conversion Functions and Conditional Expressions

Lesson Agenda

- Implicit and explicit data type conversion
- TO_CHAR, TO_DATE, TO_NUMBER functions in Oracle
- Using the CAST() function in MySQL
- General functions
- Conditional expressions
- JSON functions

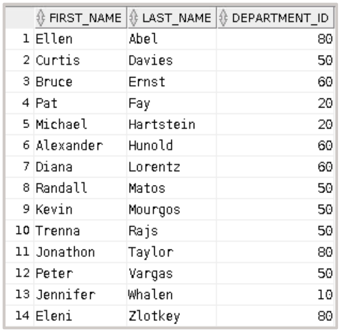
Conversion Functions


Data type conversion
Implicit data type conversion
Explicit data type conversion
Performed by the Oracle Server
Performed by the user

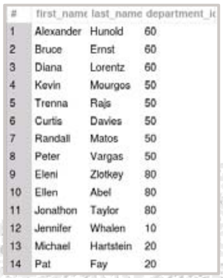
Implicit Data Type Conversion of Strings to Numbers

In expressions, the database can automatically convert strings to numbers. In this example, two strings are concatenated and then implicitly converted to a number for comparison with the numeric department ID in the WHERE clause.

```
SELECT first_name, last_name, department_id
FROM employees WHERE department_id < CONCAT('9', '0');
```



	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
1	Ellen	Abel	80
2	Curtis	Davies	50
3	Bruce	Ernst	60
4	Pat	Fay	20
5	Michael	Hartstein	20
6	Alexander	Hunold	60
7	Diana	Lorentz	60
8	Randall	Matos	50
9	Kevin	Mourgos	50
10	Trenna	Rajs	50
11	Jonathon	Taylor	80
12	Peter	Vargas	50
13	Jennifer	Whalen	10
14	Eleni	Zlotkey	80

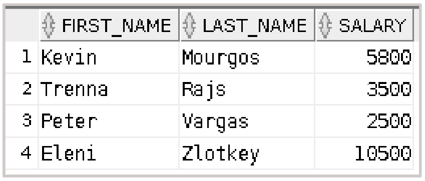


#	first_nam	last_name	department_i
1	Alexander	Hunold	60
2	Bruce	Ernst	60
3	Diana	Lorentz	60
4	Kevin	Mourgos	50
5	Trenna	Rajs	50
6	Curtis	Davies	50
7	Randall	Matos	50
8	Peter	Vargas	50
9	Eleni	Zlotkey	80
10	Ellen	Abel	80
11	Jonathon	Taylor	80
12	Jennifer	Whalen	10
13	Michael	Hartstein	20
14	Pat	Fay	20

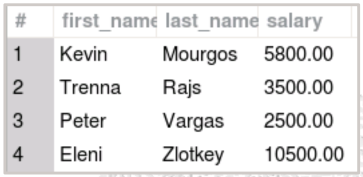
Implicit Data Type Conversion of Numbers to Strings

In expressions, the database can automatically convert numbers to strings. In this example, the salary column is converted to a string to determine if it contains a character.

```
SELECT first_name, last_name, salary
FROM employees
WHERE INSTR(salary, '5') > 0;
```



	FIRST_NAME	LAST_NAME	SALARY
1	Kevin	Mourgos	5800
2	Trenna	Rajs	3500
3	Peter	Vargas	2500
4	Eleni	Zlotkey	10500



#	first_name	last_name	salary
1	Kevin	Mourgos	5800.00
2	Trenna	Rajs	3500.00
3	Peter	Vargas	2500.00
4	Eleni	Zlotkey	10500.00

Lesson Agenda

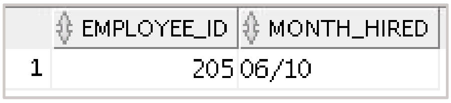
- Implicit and explicit data type conversion
- TO_CHAR, TO_DATE, TO_NUMBER functions in Oracle
- Using the CAST( )function in MySQL
- General functions
- Conditional expressions
- JSON functions

Using the TO_CHAR Function with Dates

Example:

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

```
SELECT employee_id, TO_CHAR(hire_date, 'MM/YY') Month_Hired
FROM   employees
WHERE  last_name = 'Higgins';
```



```
    EMPLOYEE_ID  MONTH_HIRED
1           205  06/10
```

Elements of the Date Format Model

Element	Result
YYYY	Full year in numbers
YEAR	Year spelled out (in English)
MM	Two-digit value for the 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

Elements of the Date Format Model

Time elements help you format the time portion of the date:

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

Add character strings by enclosing them within double quotation marks:

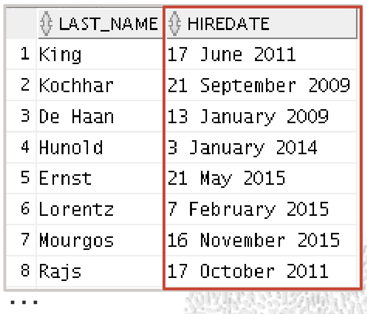
`DD "of" MONTH`	`12 of OCTOBER`

Number suffixes help in spelling out numbers:

`ddspth`	`fourteenth`

Using the TO_CHAR Function with Dates

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



	LAST_NAME	HIREDATE
1	King	17 June 2011
2	Kochhar	21 September 2009
3	De Haan	13 January 2009
4	Hunold	3 January 2014
5	Ernst	21 May 2015
6	Lorentz	7 February 2015
7	Mourgos	16 November 2015
8	Rajs	17 October 2011

...

Using the TO_CHAR Function with Numbers

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

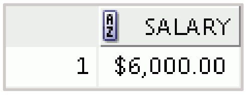
```
TO_CHAR(number[, 'format_model'])
```

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

Using the TO_CHAR Function with Numbers

Let us look at an example:

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



```
     SALARY
  1  $6,000.00
```

Using the TO_NUMBER and TO_DATE Functions

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

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

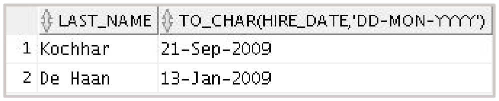
- Convert a character string to a date format using the TO_DATE function:

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

Using TO_CHAR and TO_DATE Functions with the RR Date Format

To find employees hired before 2010, use the RR date format, which produces the correct result if the command is run now or before the year 2049:

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
FROM   employees
WHERE hire_date < TO_DATE('01 Jan, 10','DD Mon,RR');
```



	LAST_NAME	TO_CHAR(HIRE_DATE,'DD-MON-YYYY')
1	Kochhar	21-Sep-2009
2	De Haan	13-Jan-2009

Lesson Agenda

- Implicit and explicit data type conversion
- TO_CHAR, TO_DATE, TO_NUMBER functions in Oracle
- Using the CAST( ) function
- General functions
- Conditional expressions
- JSON functions

Using the CAST( ) function in Oracle

CAST lets you convert one data type to another.

```
CAST(input_value as destination_type)
```

Examples:

1

```
SELECT first_name, last_name, department_id
FROM employees
WHERE department_id < CAST(CONCAT('9', '0') AS
DECIMAL(2,0));
```

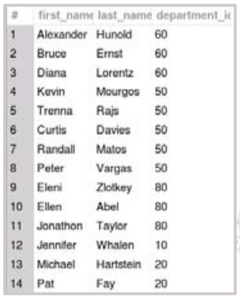
2

```
SELECT first_name, last_name, salary
FROM employees
WHERE INSTR(CAST(salary AS VARCHAR2(30)), '5')
> 0;
```

Explicit Data Type Conversion of Strings to Numbers in MySQL

You can use the `CAST()` function to explicitly convert strings to numbers. In this example, two strings are concatenated and then explicitly converted to a `DECIMAL` numeric data type to compare to the numeric department ID.

```
SELECT first_name, last_name, department_id FROM employees
WHERE department_id < CAST(CONCAT('9', '0') AS DECIMAL(2,0));
```

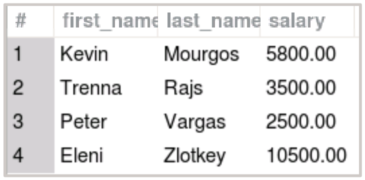


#	first_name	last_name	department_id
1	Alexander	Hunold	60
2	Bruce	Ernst	60
3	Diana	Lorentz	60
4	Kevin	Mourgos	50
5	Trenna	Rajs	50
6	Curtis	Davies	50
7	Randall	Matos	50
8	Peter	Vargas	50
9	Eleni	Zlotkey	80
10	Ellen	Abel	80
11	Jonathon	Taylor	80
12	Jennifer	Whalen	10
13	Michael	Hartstein	20
14	Pat	Fay	20

Explicit Data Type Conversion of Numbers to Strings in MySQL

You can use the `CAST()` function to explicitly convert strings to numbers. In this example, the salary column is explicitly converted to a string to determine if it contains a character.

```
SELECT first_name, last_name, salary
FROM employees
WHERE INSTR(CAST(salary AS CHAR), '5');
```



#	first_name	last_name	salary
1	Kevin	Mourgos	5800.00
2	Trenna	Rajs	3500.00
3	Peter	Vargas	2500.00
4	Eleni	Zlotkey	10500.00

Lesson Agenda

- Implicit and explicit data type conversion
- TO_CHAR, TO_DATE, TO_NUMBER functions in Oracle
- Using the CAST( ) function in MySQL
- General functions
- Conditional expressions
- JSON functions

General Functions

The following functions pertain to using nulls and can be used with any data type:

```
NVL (expr1, expr2)
NVL2 (expr1, expr2, expr3)
NULLIF (expr1, expr2)
COALESCE (expr1, expr2,
              ..., exprn)
IFNULL (expr1, expr2)
In MySQL
```

NVL Function (Oracle) and IFNULL() Function (MySQL)

Converts a null value to an actual value:

- Data types that can be used are date, character, and number.
- Data types for both expressions must match.
- Examples:

Oracle	MySQL
NVL(commission_pct,0)	IFNULL(commission_pct,0)
NVL(hire_date,'01-JAN-97')	IFNULL(hire_date, '1997-01-01')
NVL(job_id,'No Job Yet')	IFNULL(job_id,'No Job Yet')

```
NVL (expr1, expr2)
```

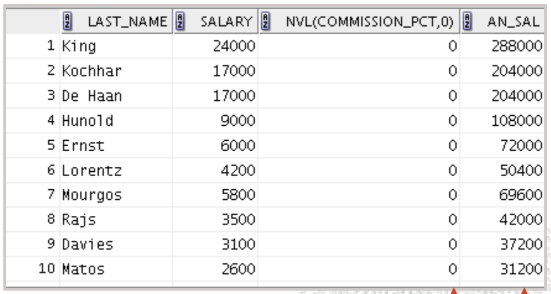
```
IFNULL(expr1, expr2)
```

Using the NVL Function in Oracle

```
SELECT last_name, salary, NVL(commission_pct, 0),
   (salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL
FROM employees;
```

1

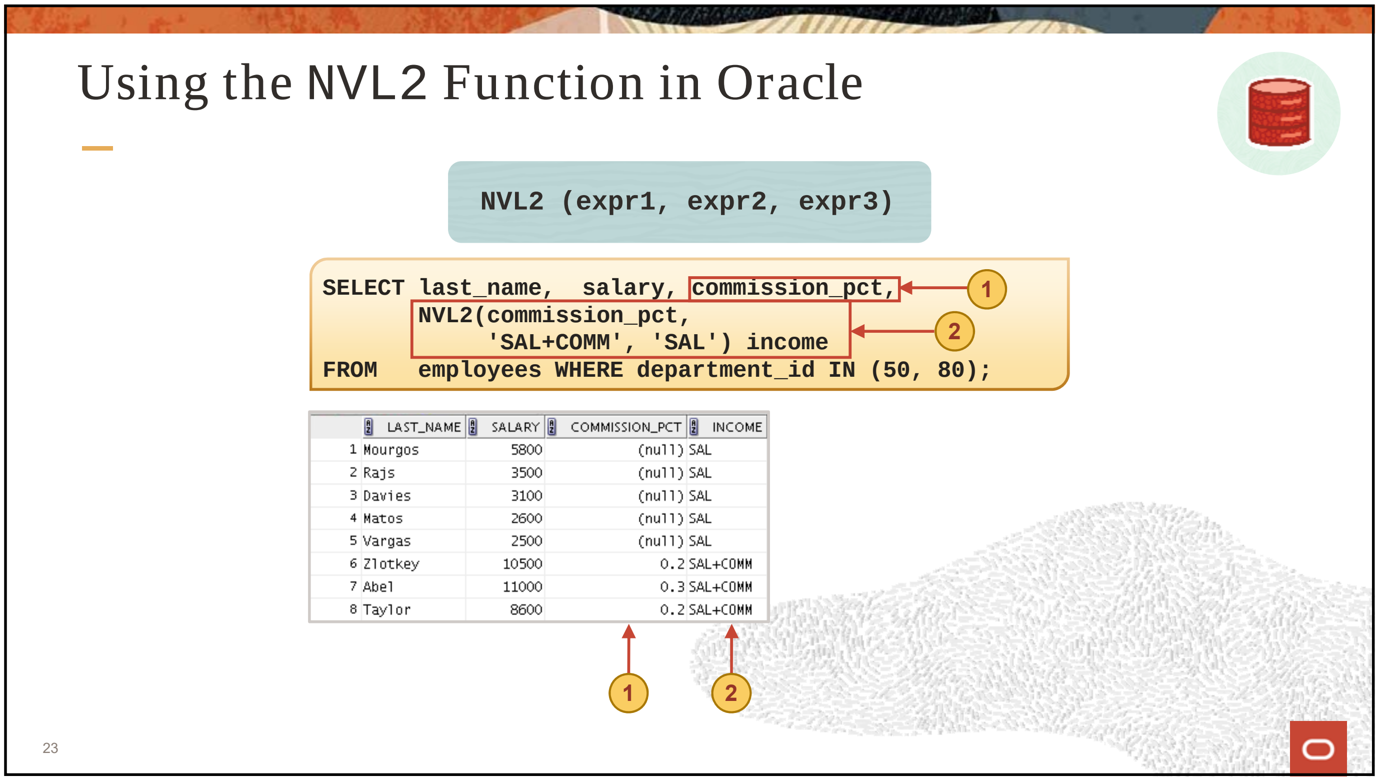
2



	LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
1	King	24000	0	288000
2	Kochhar	17000	0	204000
3	De Haan	17000	0	204000
4	Hunold	9000	0	108000
5	Ernst	6000	0	72000
6	Lorentz	4200	0	50400
7	Mourgos	5800	0	69600
8	Rajs	3500	0	42000
9	Davies	3100	0	37200
10	Matos	2600	0	31200

1

2



Using the NVL2 Function in Oracle

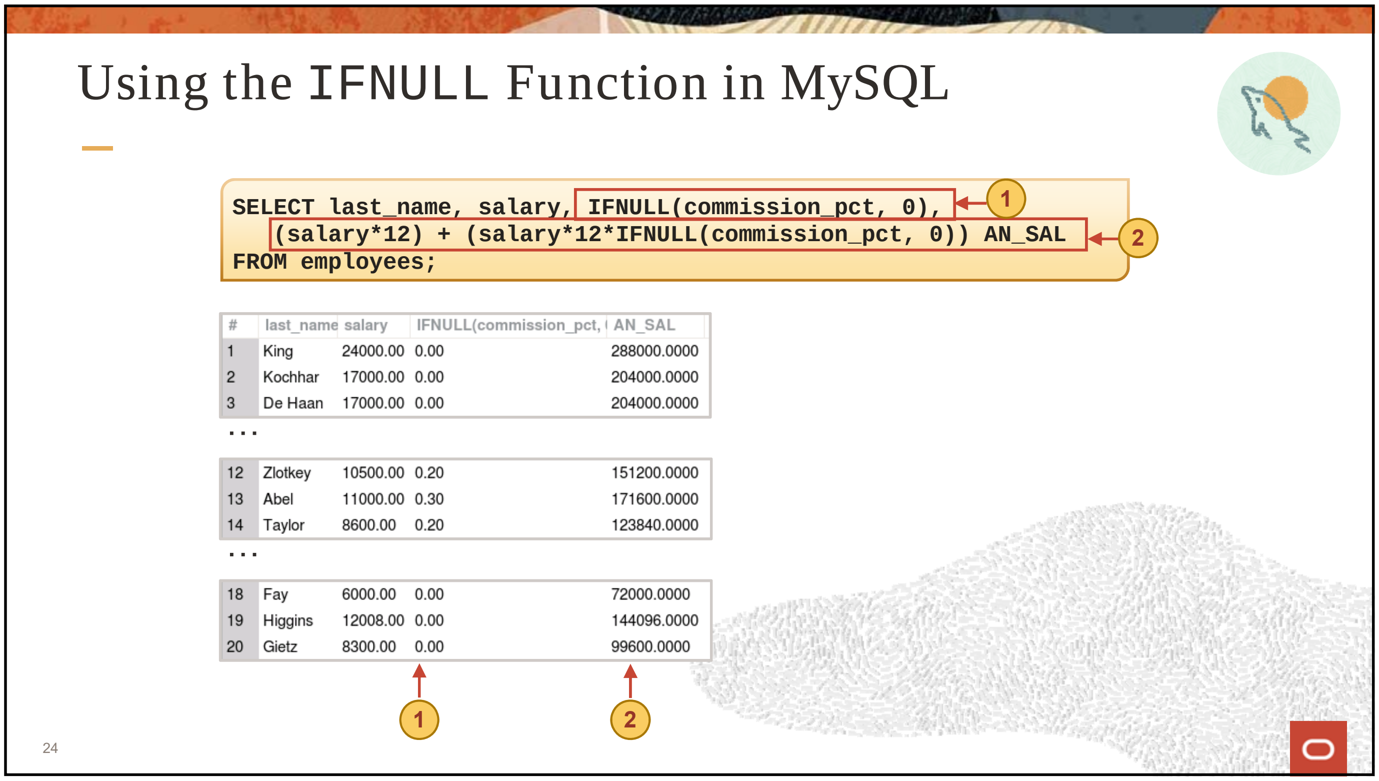
```
NVL2 (expr1, expr2, expr3)
```

```
SELECT last_name,  salary, commission_pct,        ← 1
       NVL2(commission_pct,                        ← 2
            'SAL+COMM', 'SAL') income
FROM   employees WHERE department_id IN (50, 80);
```

	LAST_NAME	SALARY	COMMISSION_PCT	INCOME
1	Mourgos	5800	(null)	SAL
2	Rajs	3500	(null)	SAL
3	Davies	3100	(null)	SAL
4	Matos	2600	(null)	SAL
5	Vargas	2500	(null)	SAL
6	Zlotkey	10500	0.2	SAL+COMM
7	Abel	11000	0.3	SAL+COMM
8	Taylor	8600	0.2	SAL+COMM

1 2

23



Using the IFNULL Function in MySQL

```
SELECT last_name, salary, IFNULL(commission_pct, 0),                ← 1
   (salary*12) + (salary*12*IFNULL(commission_pct, 0)) AN_SAL       ← 2
FROM employees;
```

#	last_name	salary	IFNULL(commission_pct, (	AN_SAL
1	King	24000.00	0.00	288000.0000
2	Kochhar	17000.00	0.00	204000.0000
3	De Haan	17000.00	0.00	204000.0000
...				
12	Zlotkey	10500.00	0.20	151200.0000
13	Abel	11000.00	0.30	171600.0000
14	Taylor	8600.00	0.20	123840.0000
...				
18	Fay	6000.00	0.00	72000.0000
19	Higgins	12008.00	0.00	144096.0000
20	Gietz	8300.00	0.00	99600.0000

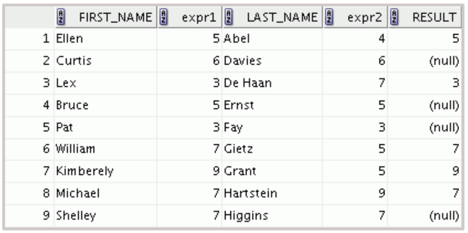
1 2

24

Using the NULLIF Function

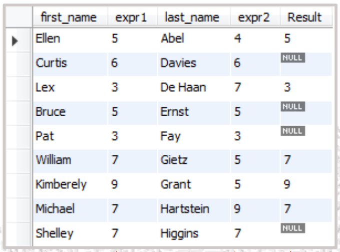
```
NULLIF (expr1, expr2)
```

```
SELECT first_name, LENGTH(first_name) AS expr1,
       last_name,  LENGTH(last_name)  AS expr2,
       NULLIF(LENGTH(first_name), LENGTH(last_name)) AS Result
FROM   employees;
```



	FIRST_NAME	expr1	LAST_NAME	expr2	RESULT
1	Ellen	5	Abel	4	5
2	Curtis	6	Davies	6	(null)
3	Lex	3	De Haan	7	3
4	Bruce	5	Ernst	5	(null)
5	Pat	3	Fay	3	(null)
6	William	7	Gietz	5	7
7	Kimberely	9	Grant	5	9
8	Michael	7	Hartstein	9	7
9	Shelley	7	Higgins	7	(null)

...



first_name	expr1	last_name	expr2	Result
Ellen	5	Abel	4	5
Curtis	6	Davies	6	NULL
Lex	3	De Haan	7	3
Bruce	5	Ernst	5	NULL
Pat	3	Fay	3	NULL
William	7	Gietz	5	7
Kimberely	9	Grant	5	9
Michael	7	Hartstein	9	7
Shelley	7	Higgins	7	NULL

...

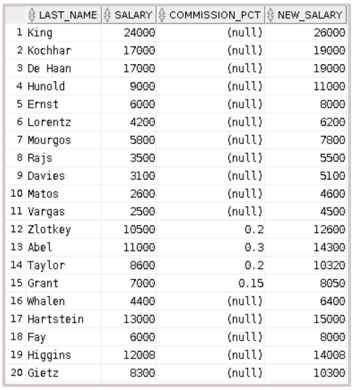
Using the COALESCE Function

- The advantage of the `COALESCE` function over the `NVL` or `IFNULL` functions is that the `COALESCE` function can take multiple alternative values.
- If the first expression is not null, the `COALESCE` function returns that expression; otherwise, it does a `COALESCE` of the remaining expressions.

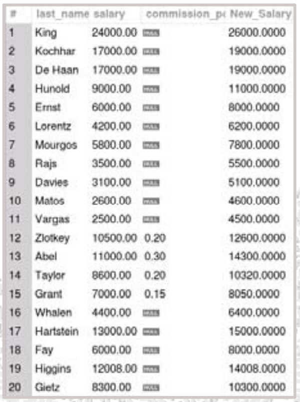
```
COALESCE (expr1, expr2, ..., exprn)
```

Using the COALESCE Function

```
SELECT last_name, salary, commission_pct,
COALESCE((salary+(commission_pct*salary)), salary+2000)
AS New_Salary
FROM   employees;
```



	LAST_NAME	SALARY	COMMISSION_PCT	NEW_SALARY
1	King	24000	(null)	26000
2	Kochhar	17000	(null)	19000
3	De Haan	17000	(null)	19000
4	Hunold	9000	(null)	11000
5	Ernst	6000	(null)	8000
6	Lorentz	4200	(null)	6200
7	Mourgos	5800	(null)	7800
8	Rajs	3500	(null)	5500
9	Davies	3100	(null)	5100
10	Matos	2600	(null)	4600
11	Vargas	2500	(null)	4500
12	Zlotkey	10500	0.2	12600
13	Abel	11000	0.3	14300
14	Taylor	8600	0.2	10320
15	Grant	7000	0.15	8050
16	Whalen	4400	(null)	6400
17	Hartstein	13000	(null)	15000
18	Fay	6000	(null)	8000
19	Higgins	12008	(null)	14008
20	Gietz	8300	(null)	10300



#	last_name	salary	commission_pc	New_Salary
1	King	24000.00	NULL	26000.0000
2	Kochhar	17000.00	NULL	19000.0000
3	De Haan	17000.00	NULL	19000.0000
4	Hunold	9000.00	NULL	11000.0000
5	Ernst	6000.00	NULL	8000.0000
6	Lorentz	4200.00	NULL	6200.0000
7	Mourgos	5800.00	NULL	7800.0000
8	Rajs	3500.00	NULL	5500.0000
9	Davies	3100.00	NULL	5100.0000
10	Matos	2600.00	NULL	4600.0000
11	Vargas	2500.00	NULL	4500.0000
12	Zlotkey	10500.00	0.20	12600.0000
13	Abel	11000.00	0.30	14300.0000
14	Taylor	8600.00	0.20	10320.0000
15	Grant	7000.00	0.15	8050.0000
16	Whalen	4400.00	NULL	6400.0000
17	Hartstein	13000.00	NULL	15000.0000
18	Fay	6000.00	NULL	8000.0000
19	Higgins	12008.00	NULL	14008.0000
20	Gietz	8300.00	NULL	10300.0000

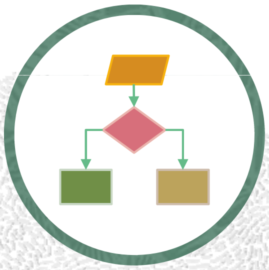
Lesson Agenda

- Implicit and explicit data type conversion
- TO_CHAR, TO_DATE, TO_NUMBER functions in Oracle
- Using the CAST( ) function in MySQL
- General functions
- Conditional expressions
- JSON functions

Conditional Expressions

Help provide the use of `IF-THEN-ELSE` logic within a SQL statement

- You can use the following methods:
  - `CASE` expression
  - Searched `CASE` expression
  - `DECODE` function



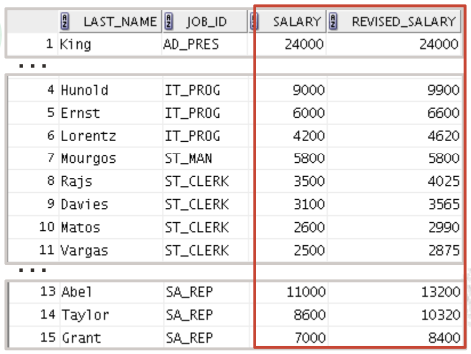
CASE Expression

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

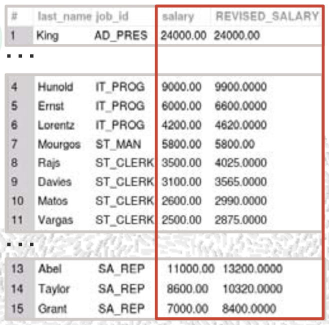
```
CASE expr WHEN comparison_expr1 THEN return_expr1
         [WHEN comparison_expr2 THEN return_expr2
          WHEN comparison_exprn THEN return_exprn
          ELSE else_expr]
END
```

Using the CASE Expression

```
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      AS REVISED_SALARY
FROM   employees;
```



	LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
1	King	AD_PRES	24000	24000
...				
4	Hunold	IT_PROG	9000	9900
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
10	Matos	ST_CLERK	2600	2990
11	Vargas	ST_CLERK	2500	2875
...				
13	Abel	SA_REP	11000	13200
14	Taylor	SA_REP	8600	10320
15	Grant	SA_REP	7000	8400



#	last_name	job_id	salary	REVISED_SALARY
1	King	AD_PRES	24000.00	24000.00
...				
4	Hunold	IT_PROG	9000.00	9900.0000
5	Ernst	IT_PROG	6000.00	6600.0000
6	Lorentz	IT_PROG	4200.00	4620.0000
7	Mourgos	ST_MAN	5800.00	5800.00
8	Rajs	ST_CLERK	3500.00	4025.0000
9	Davies	ST_CLERK	3100.00	3565.0000
10	Matos	ST_CLERK	2600.00	2990.0000
11	Vargas	ST_CLERK	2500.00	2875.0000
...				
13	Abel	SA_REP	11000.00	13200.0000
14	Taylor	SA_REP	8600.00	10320.0000
15	Grant	SA_REP	7000.00	8400.0000

Searched CASE Expression

```
CASE
     WHEN condition1 THEN use_expression1
     WHEN condition2 THEN use_expression2
     WHEN condition3 THEN use_expression3
     ELSE default_use_expression
END
```

```
SELECT last_name,salary,
(CASE WHEN salary<5000 THEN 'Low'
      WHEN salary<10000 THEN 'Medium'
      WHEN salary<20000 THEN 'Good'
      ELSE 'Excellent'
END) AS qualified_salary
FROM employees;
```

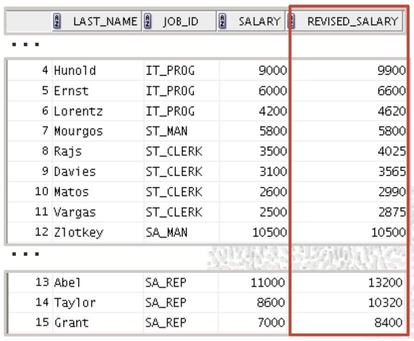
DECODE Function in Oracle

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

```
DECODE(col|expression, search1, result1
                       [, search2, result2,...,]
                       [, default])
```

Using the DECODE Function

```
SELECT last_name, job_id, salary,
       DECODE(job_id, 'IT_PROG',  1.10*salary,
                      'ST_CLERK', 1.15*salary,
                      'SA_REP',   1.20*salary,
              salary)
       REVISED_SALARY
FROM   employees;
```



	LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...				
4	Hunold	IT_PROG	9000	9900
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
10	Matos	ST_CLERK	2600	2990
11	Vargas	ST_CLERK	2500	2875
12	Zlotkey	SA_MAN	10500	10500
...				
13	Abel	SA_REP	11000	13200
14	Taylor	SA_REP	8600	10320
15	Grant	SA_REP	7000	8400

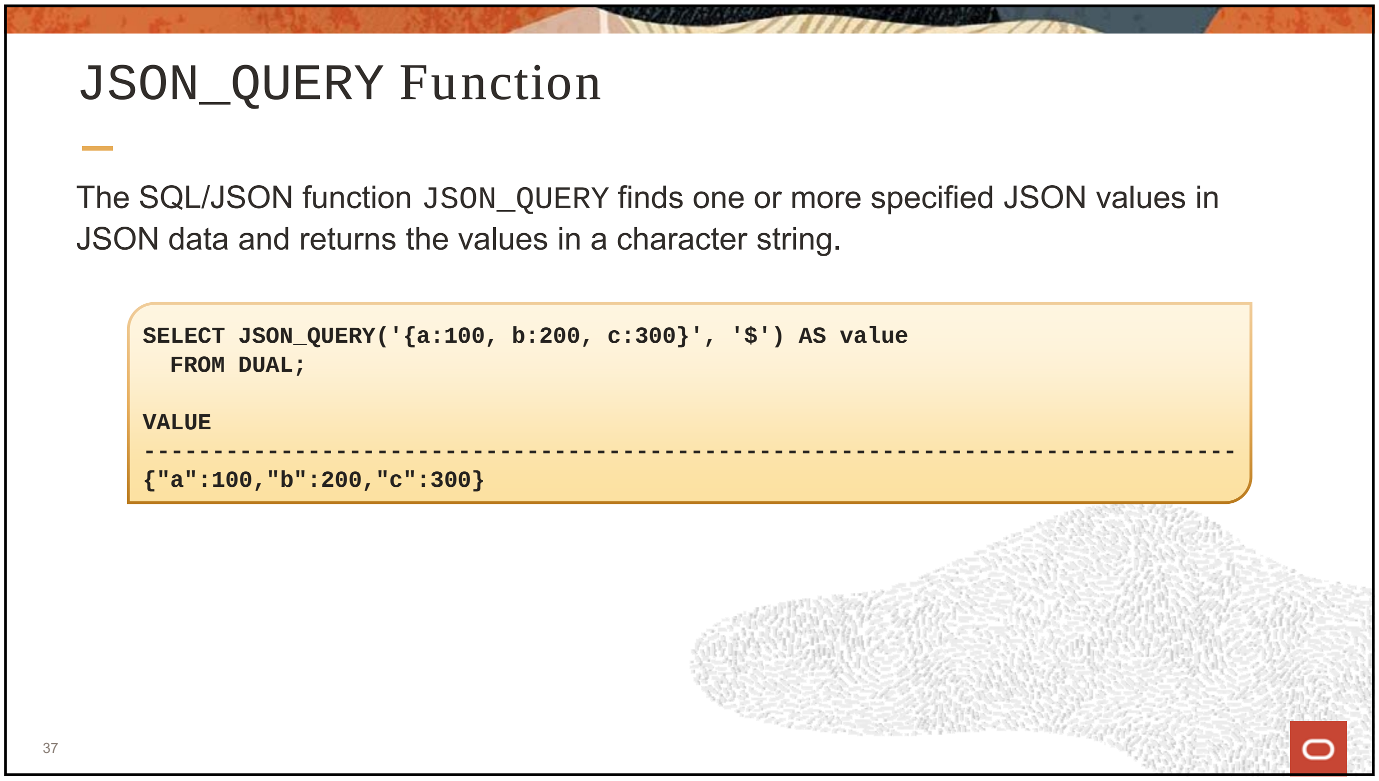
Using the DECODE Function

Display the applicable tax rate for each employee in department 80:

```
SELECT last_name, salary,
       DECODE (TRUNC(salary/2000, 0),
                         0, 0.00,
                         1, 0.09,
                         2, 0.20,
                         3, 0.30,
                         4, 0.40,
                         5, 0.42,
                         6, 0.44,
                            0.45) TAX_RATE
FROM   employees
WHERE  department_id = 80;
```

Lesson Agenda

- Implicit and explicit data type conversion
- TO_CHAR, TO_DATE, TO_NUMBER functions in Oracle
- Using the CAST( ) function in MySQL
- General functions
- Conditional expressions
- JSON functions

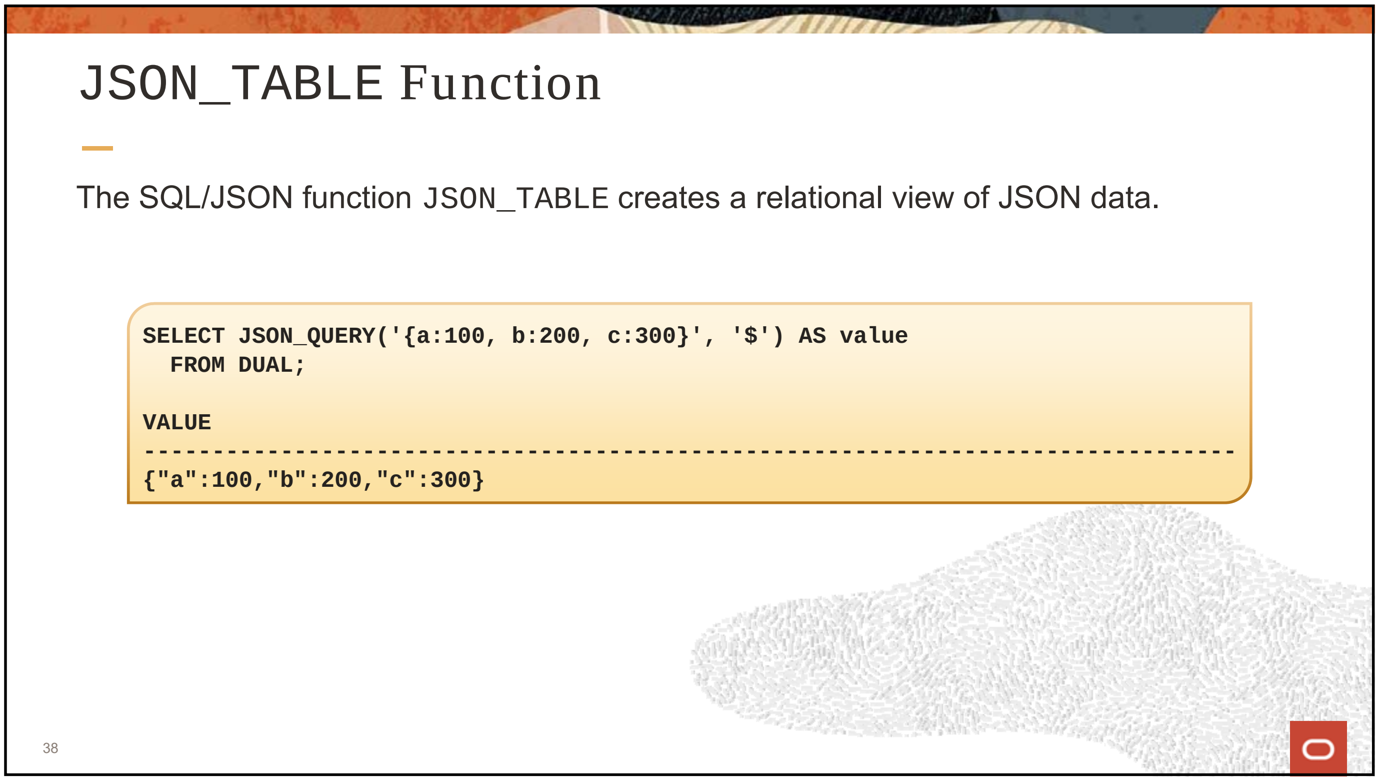


JSON_QUERY Function

The SQL/JSON function `JSON_QUERY` finds one or more specified JSON values in JSON data and returns the values in a character string.

```
SELECT JSON_QUERY('{a:100, b:200, c:300}', '$') AS value
  FROM DUAL;

VALUE
--------------------------------------------------------------------------------
{"a":100,"b":200,"c":300}
```



JSON_TABLE Function

The SQL/JSON function `JSON_TABLE` creates a relational view of JSON data.

```
SELECT JSON_QUERY('{a:100, b:200, c:300}', '$') AS value
  FROM DUAL;

VALUE
--------------------------------------------------------------------------------
{"a":100,"b":200,"c":300}
```

JSON_VALUE Function

The SQL/JSON function JSON_QUERY finds one or more specified JSON values in JSON data and returns the values in a character string.

```
SELECT JSON_VALUE('{a:100}', '$.a') AS value
  FROM DUAL;

VALUE
-----
100
```

Summary

In this lesson, you should have learned how to:

- Alter date formats for display using functions
- Convert column data types using functions
- Use NVL functions
- Use IF-THEN-ELSE logic and other conditional expressions in a SELECT statement