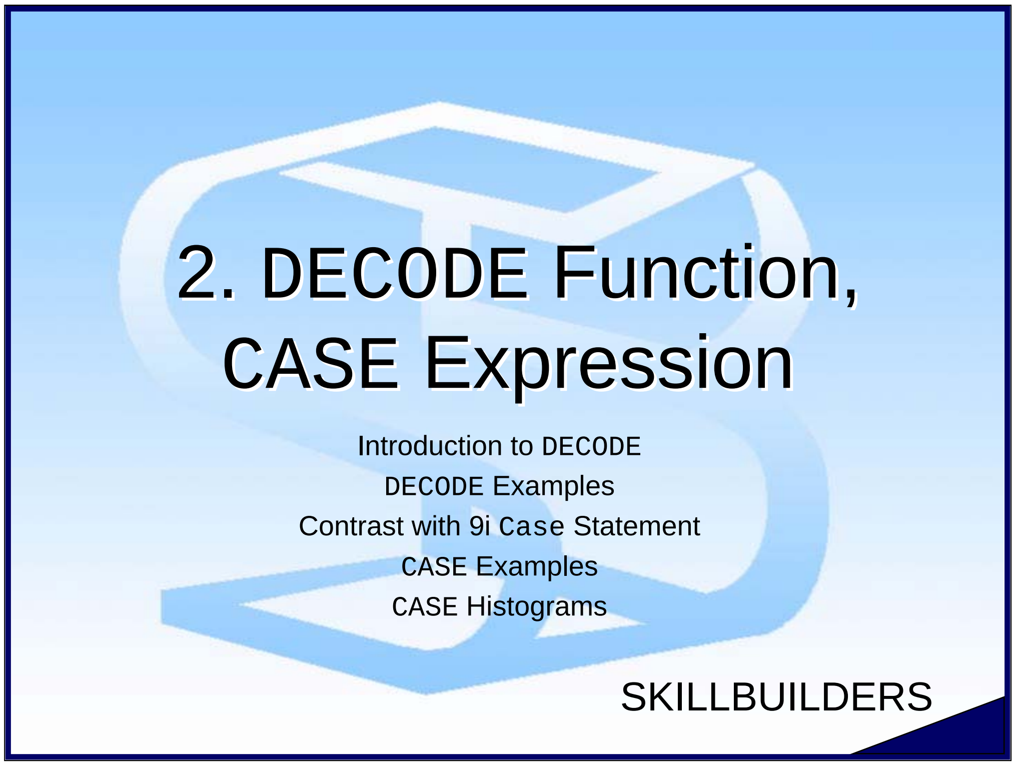
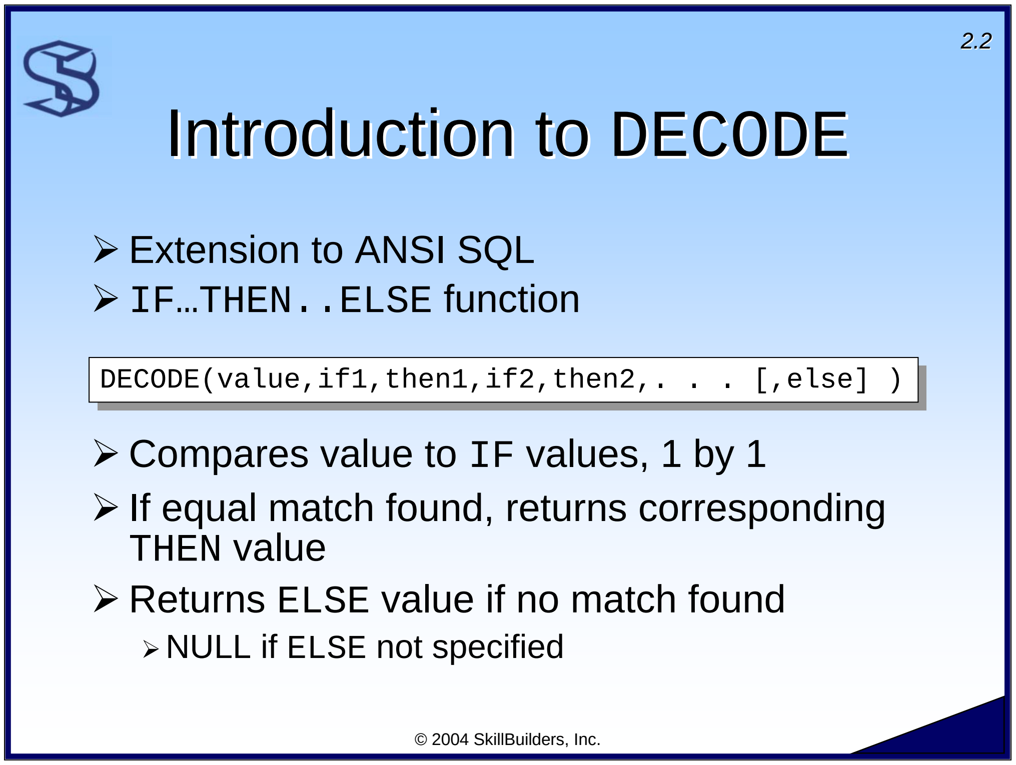


2. DECODE Function, CASE Expression

Introduction to DECODE
DECODE Examples
Contrast with 9i Case Statement
CASE Examples
CASE Histograms

SKILLBUILDERS



2.2

Introduction to DECODE

- Extension to ANSI SQL
- IF…THEN..ELSE function

```
DECODE(value,if1,then1,if2,then2,. . . [,else] )
```

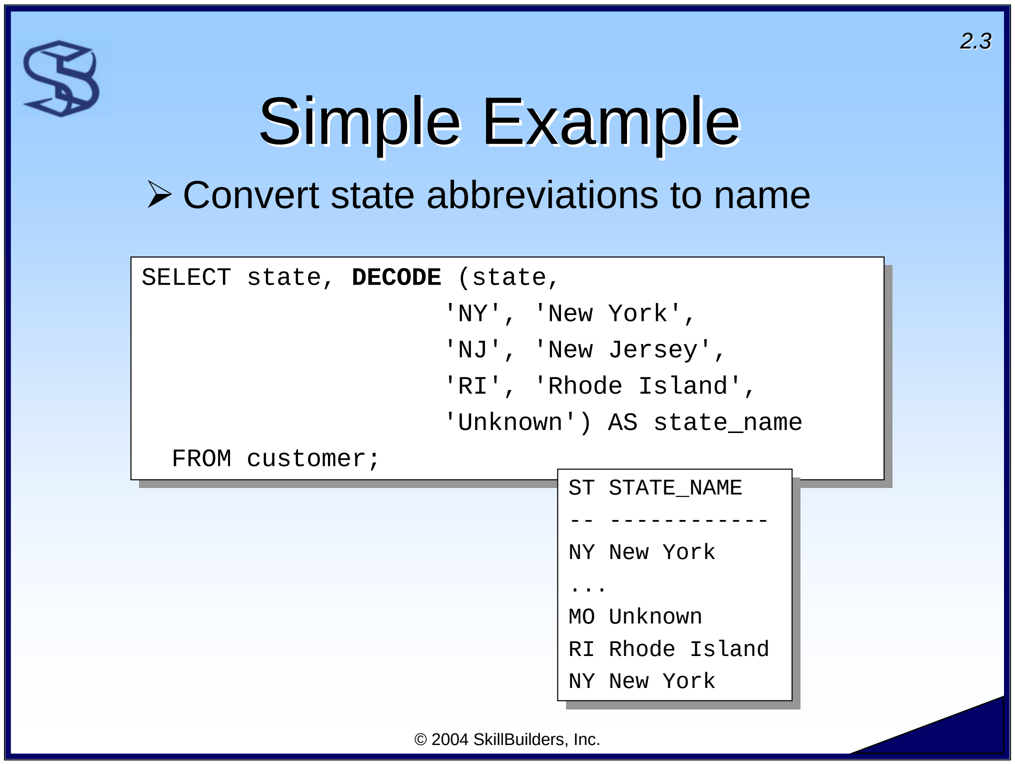
- Compares value to IF values, 1 by 1
- If equal match found, returns corresponding THEN value
- Returns ELSE value if no match found
  - NULL if ELSE not specified

© 2004 SkillBuilders, Inc.

DECODE is an Oracle extension that adds an *IF/THEN/ELSE* function to SQL. The syntax allows the specification of a limited number of *IF/THEN* value pairs. Oracle compares the supplied VALUE to the *IF* values (one by one) and returns the first *THEN* value if an equality match is found. If a match is not found, the *ELSE* value is returned, if coded. If *ELSE* is not coded and a match is not found, NULL is returned.

Notable things about DECODE:

- *VALUE, IF, THEN* and *ELSE* can be expressions. For example, SYSDATE-BIRTHDATE. Functions can also be used. (See examples later in this section.)
- In DECODE, Oracle considers two NULLs to be equivalent.
- The maximum number of items, including *VALUE, IF, THEN* and *ELSE* values is 255.



2.3

Simple Example

- Convert state abbreviations to name

```
SELECT state, DECODE (state,
                      'NY', 'New York',
                      'NJ', 'New Jersey',
                      'RI', 'Rhode Island',
                      'Unknown') AS state_name
  FROM customer;
```

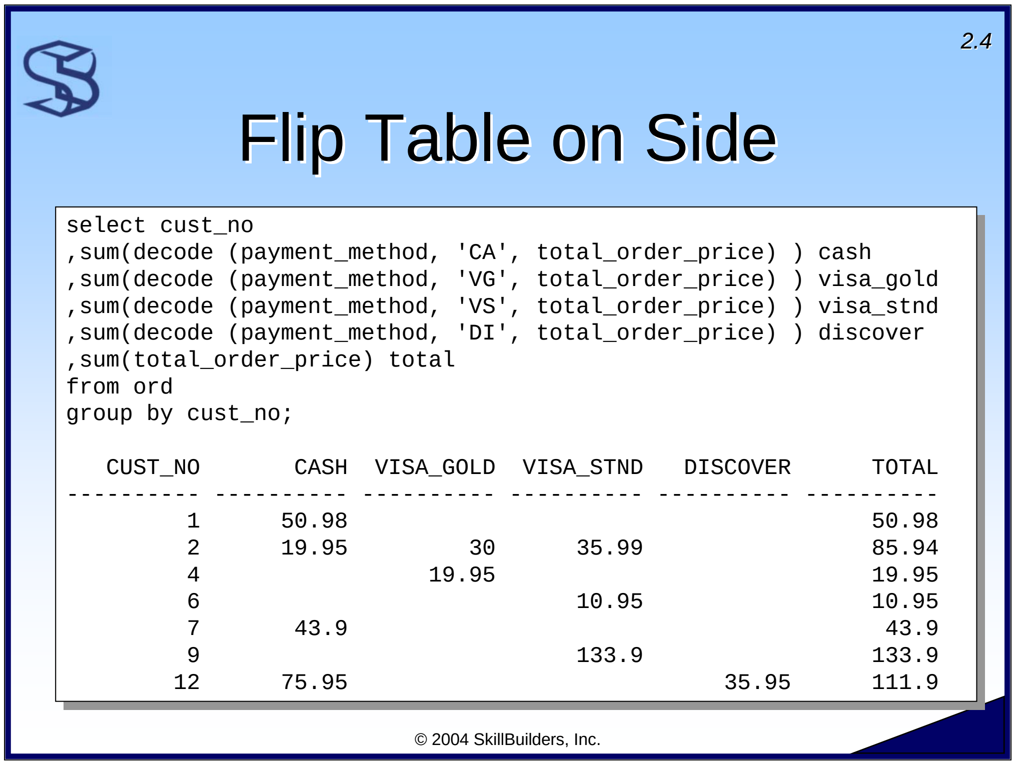
```
ST STATE_NAME
-- ------------
NY New York
...
MO Unknown
RI Rhode Island
NY New York
```

© 2004 SkillBuilders, Inc.

DECODE Function

The example decodes state abbreviations and replaces them with state names. Note that the `DECODE` function has code/value pairs, followed by an optional default value to be used if the code is not found in the set of code/value pairs. `NULL` is used if there is no default specified.

Idea: `DECODE` is an extremely powerful function. It can be used to perform a host of special manipulations such as invoice aging, flipping tables on their side, and calculating columns where the calculation itself varies by row. For example, you could calculate the new salary for all employees where the increase varies based on the employees date of hire. These uses are beyond the scope of this course. **Oracle: The Complete Reference**, the Oracle Press book from Osborne, has an entire chapter dedicated to the various ways that you can use `DECODE`.



2.4

Flip Table on Side

```
select cust_no
,sum(decode (payment_method, 'CA', total_order_price) ) cash
,sum(decode (payment_method, 'VG', total_order_price) ) visa_gold
,sum(decode (payment_method, 'VS', total_order_price) ) visa_stnd
,sum(decode (payment_method, 'DI', total_order_price) ) discover
,sum(total_order_price) total
from ord
group by cust_no;

   CUST_NO       CASH  VISA_GOLD  VISA_STND   DISCOVER      TOTAL
---------- ---------- ---------- ---------- ---------- ----------
         1      50.98                                       50.98
         2      19.95         30      35.99                 85.94
         4                 19.95                            19.95
         6                            10.95                 10.95
         7       43.9                                        43.9
         9                            133.9                 133.9
        12      75.95                            35.95      111.9
```

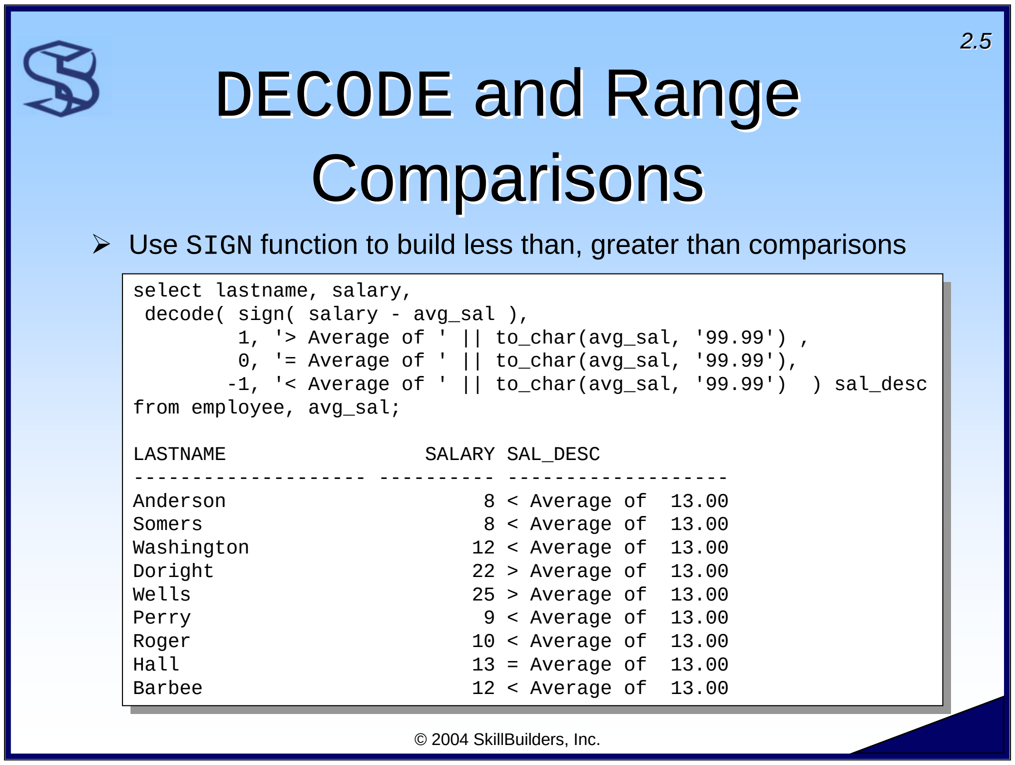
© 2004 SkillBuilders, Inc.

The `DECODE` function is often used to "turn a table on its side." Or, put another way, turn table values into columns. The essence of doing this is to:

- `DECODE` the column you would like to make into columns. In this example, I have decoded `PAYMENT_METHOD`.
- Supply a relevant column for the *THEN* value. In this case I have coded *`TOTAL_ORDER_PRICE`*.
- Code *`NULL`* (or take the default `NULL` as in this example) for the *ELSE* value.
- Supply a meaningful column alias (column name). In this example, I have supplied "`cash`" for value "CA", `visa_gold`, "VG", etc.

In this example I have also use the aggregate function `SUM` to display the total amount spent by a customer via cash payment method. For example, customer number 2 has spent a total of $19.95 in cash, $30 in Visa Gold and $35.99 in Visa Standard.

See the supplied script `DECODE2.SQL` for a working example.



2.5

DECODE and Range Comparisons

- Use `SIGN` function to build less than, greater than comparisons

```
select lastname, salary,
 decode( sign( salary - avg_sal ),
         1, '> Average of ' || to_char(avg_sal, '99.99') ,
         0, '= Average of ' || to_char(avg_sal, '99.99'),
        -1, '< Average of ' || to_char(avg_sal, '99.99')  ) sal_desc
from employee, avg_sal;

LASTNAME                 SALARY SAL_DESC
-------------------- ---------- -------------------
Anderson                      8 < Average of  13.00
Somers                        8 < Average of  13.00
Washington                   12 < Average of  13.00
Doright                      22 > Average of  13.00
Wells                        25 > Average of  13.00
Perry                         9 < Average of  13.00
Roger                        10 < Average of  13.00
Hall                         13 = Average of  13.00
Barbee                       12 < Average of  13.00
```

We can build a range comparison into the `DECODE` by incorporating the `SIGN` function. `SIGN` returns:

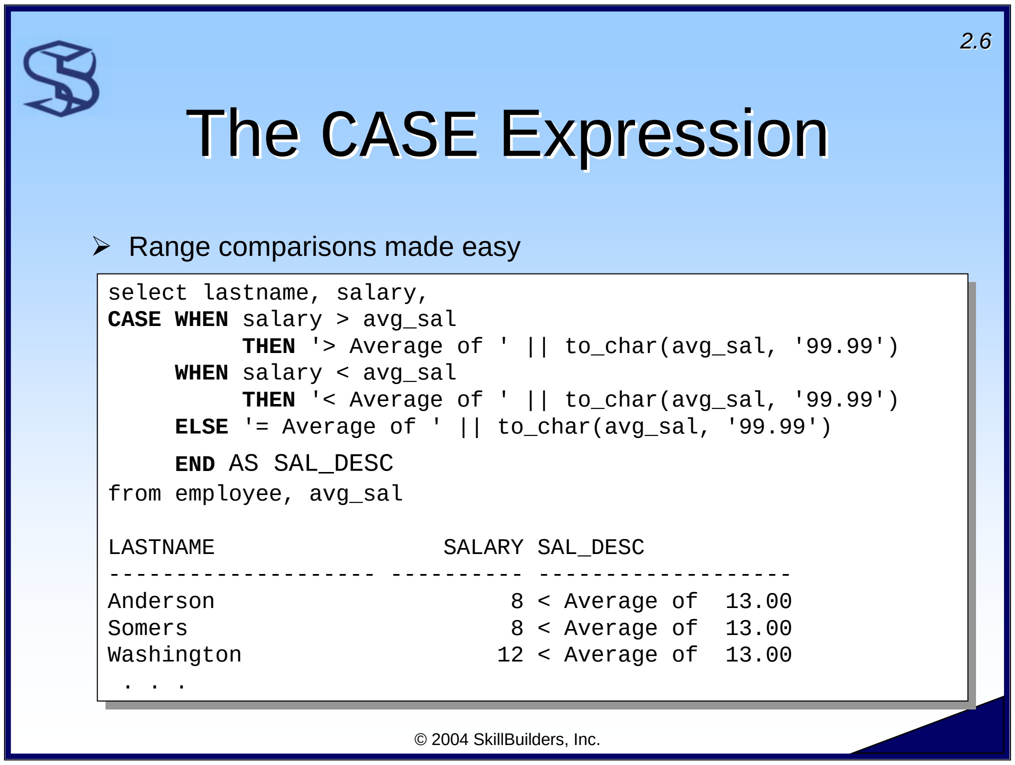
- 1 if the result is positive
- -1 if the result is negative
- 0 if the result is 0

This example assumes the following view has been created:

```
create or replace view avg_Sal
  as select trunc(avg(salary)) avg_sal
  from employee;
```

To help understand this example, divide employee Anderson's salary by the average and subtract 1: 8 - 13 = -5. The `SIGN` function returns –1 (since the result is negative) and `DECODE` translates that into the string "`< Average of ' || to_char(avg_sal, '99.99').`" Try the same for Hall: 13 - 13 = 0. Since the result is 0, the `SIGN` function returns 0.

See the supplied script `DECODE3.SQL` for a working example.



2.6

The CASE Expression

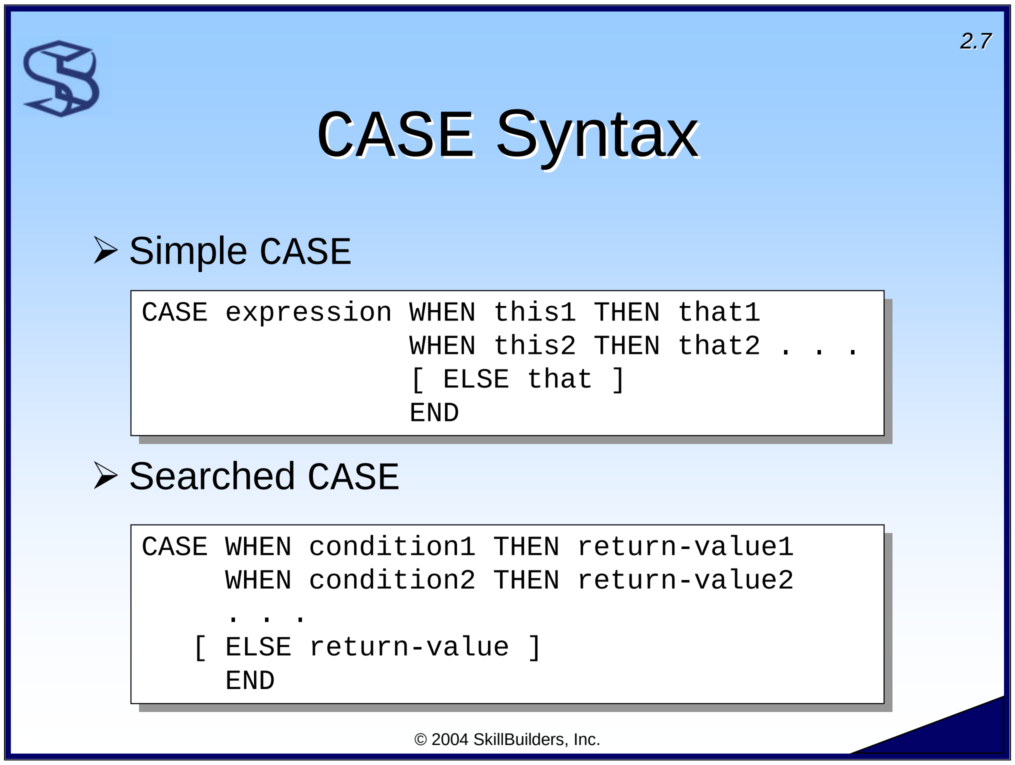
- Range comparisons made easy

```
select lastname, salary,
CASE WHEN salary > avg_sal
          THEN '> Average of ' || to_char(avg_sal, '99.99')
     WHEN salary < avg_sal
          THEN '< Average of ' || to_char(avg_sal, '99.99')
     ELSE '= Average of ' || to_char(avg_sal, '99.99')
     END AS SAL_DESC
from employee, avg_sal

LASTNAME                  SALARY SAL_DESC
-------------------- ---------- -------------------
Anderson                       8 < Average of  13.00
Somers                         8 < Average of  13.00
Washington                    12 < Average of  13.00
 . . .
```

Oracle provided ANSI CASE support in their SQL language in release 8.1.7. It is not supported in PL/SQL until Oracle9i. (Workaround: Include it with dynamic SQL.)

CASE is limited to 128 WHEN/THEN pairs (255 total values). This limit can be overcome by nesting CASE within CASE.



2.7

CASE Syntax

- Simple CASE

```
CASE expression WHEN this1 THEN that1
                WHEN this2 THEN that2 . . .
                [ ELSE that ]
                END
```

- Searched CASE

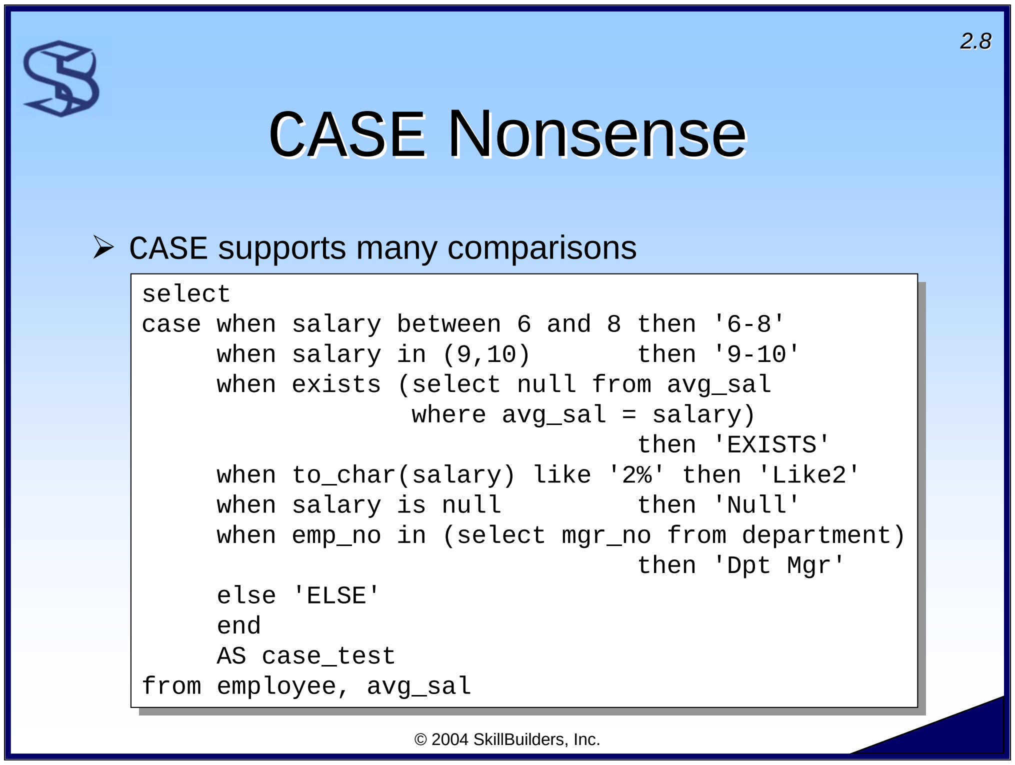
```
CASE WHEN condition1 THEN return-value1
     WHEN condition2 THEN return-value2
     . . .
   [ ELSE return-value ]
     END
```

© 2004 SkillBuilders, Inc.

Oracle supports two flavors of `CASE`, simple and searched.

The simple case expression tests for an equal condition on the supplied value or expression. The first `WHEN` value that is equal causes Oracle to return the corresponding `THEN` value. If none of the `WHEN` values match the supplied expression, the `ELSE` value is returned. If the `ELSE` is not coded, `NULL` is returned.

The searched case (as seen in the previous example) allows multiple comparison expressions (`<, >, <=, >=, BETWEEN, LIKE, IN, IS NULL`, etc.). The first *TRUE* expression causes Oracle to return the corresponding `THEN` value. If none of the `WHEN` values match the supplied expression, the `ELSE` value is returned. If the `ELSE` is not coded, `NULL` is returned.



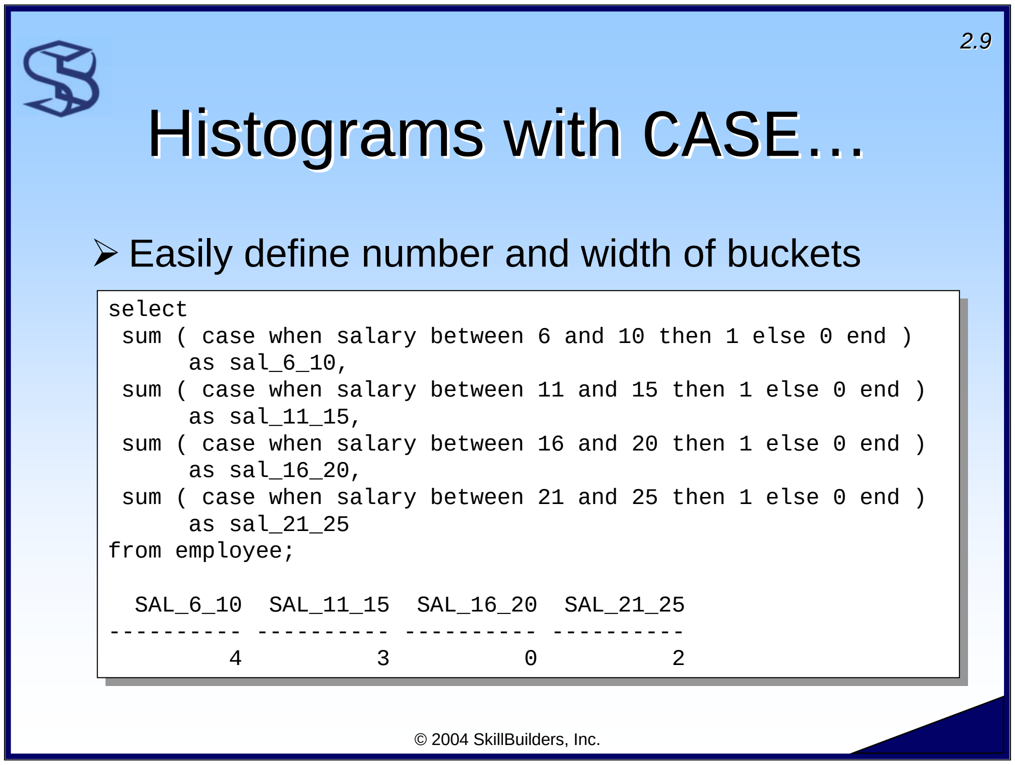
2.8

CASE Nonsense

- CASE supports many comparisons

```
select
case when salary between 6 and 8 then '6-8'
     when salary in (9,10)       then '9-10'
     when exists (select null from avg_sal
                  where avg_sal = salary)
                                 then 'EXISTS'
     when to_char(salary) like '2%' then 'Like2'
     when salary is null         then 'Null'
     when emp_no in (select mgr_no from department)
                                 then 'Dpt Mgr'
     else 'ELSE'
     end
     AS case_test
from employee, avg_sal
```

This example illustrates the breadth of support for the various relational comparison operators supported by Oracle.



2.9

Histograms with CASE…

- Easily define number and width of buckets

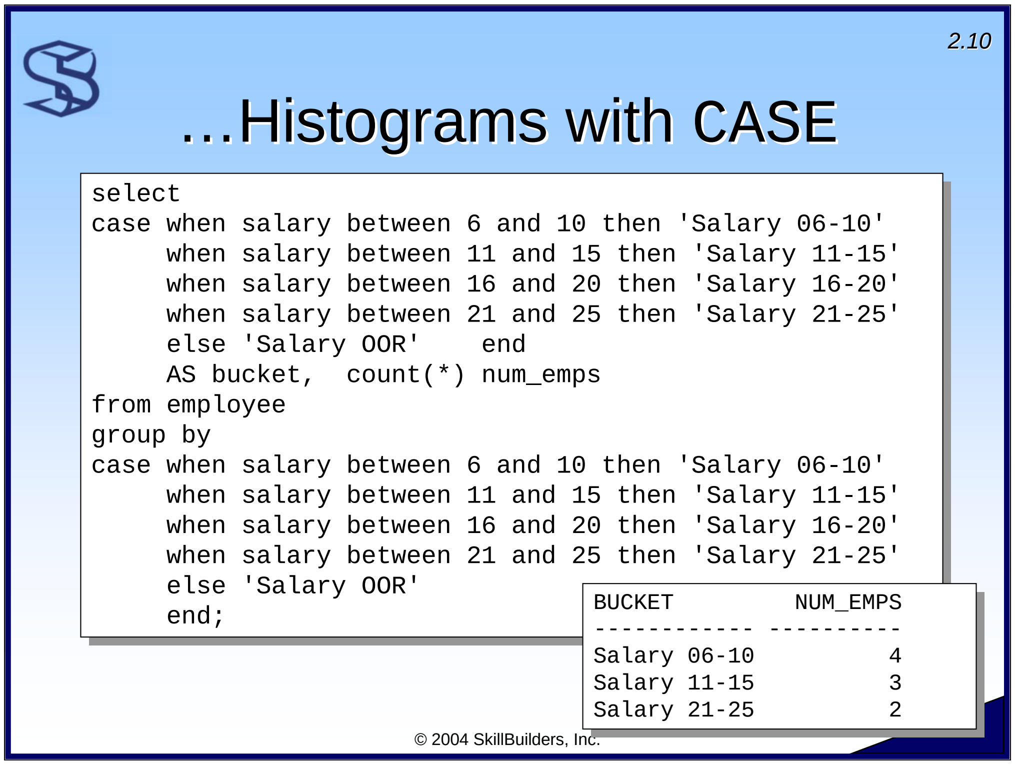
```
select
 sum ( case when salary between 6 and 10 then 1 else 0 end )
      as sal_6_10,
 sum ( case when salary between 11 and 15 then 1 else 0 end )
      as sal_11_15,
 sum ( case when salary between 16 and 20 then 1 else 0 end )
      as sal_16_20,
 sum ( case when salary between 21 and 25 then 1 else 0 end )
      as sal_21_25
from employee;

  SAL_6_10  SAL_11_15  SAL_16_20  SAL_21_25
---------- ---------- ---------- ----------
         4          3          0          2
```

© 2004 SkillBuilders, Inc.

Histograms can be easily created with the CASE expression.

In this example, to show the distribution of salary levels, I have created a histogram containing 4 buckets and each bucket has a width of 5.



2.10

…Histograms with CASE

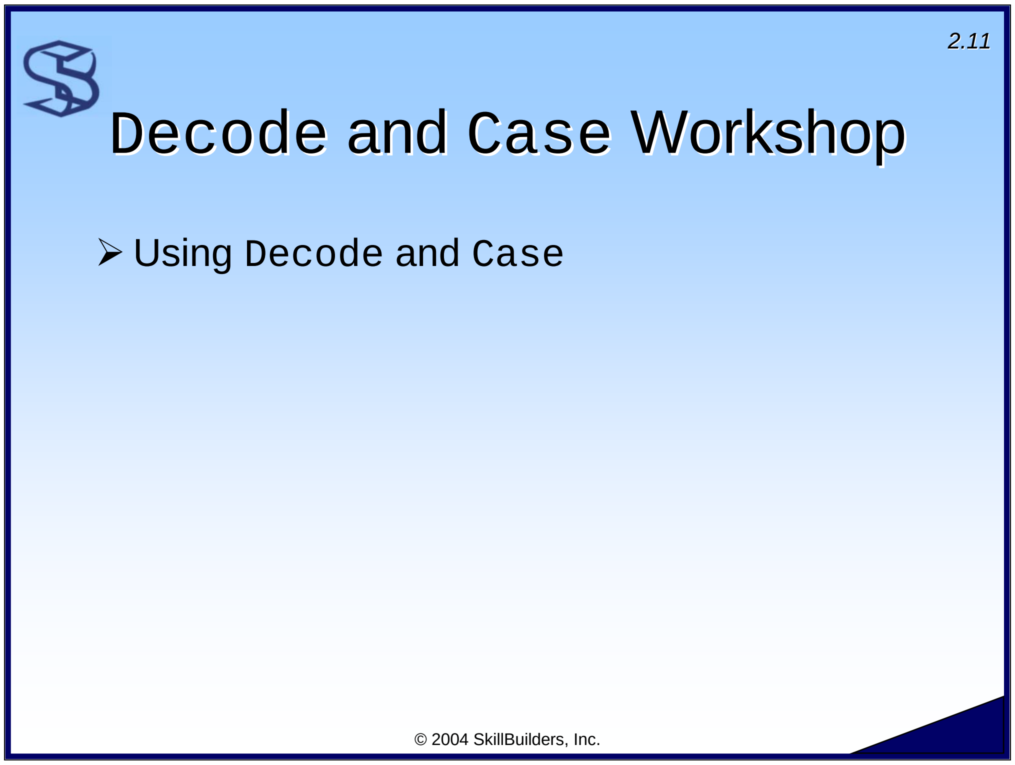
```
select
case when salary between 6 and 10 then 'Salary 06-10'
     when salary between 11 and 15 then 'Salary 11-15'
     when salary between 16 and 20 then 'Salary 16-20'
     when salary between 21 and 25 then 'Salary 21-25'
     else 'Salary OOR'    end
     AS bucket,  count(*) num_emps
from employee
group by
case when salary between 6 and 10 then 'Salary 06-10'
     when salary between 11 and 15 then 'Salary 11-15'
     when salary between 16 and 20 then 'Salary 16-20'
     when salary between 21 and 25 then 'Salary 21-25'
     else 'Salary OOR'
     end;
```

```
BUCKET         NUM_EMPS
------------ ----------
Salary 06-10          4
Salary 11-15          3
Salary 21-25          2
```

© 2004 SkillBuilders, Inc.

This example shows that the CASE expression can be used in the GROUP BY clause.

Like the previous example I am showing the distribution of salary levels. However, in this example, each bucket in the histogram is represented by a row in the result (as opposed to a column), and empty buckets are not included in the result.



2.11

Decode and Case Workshop

- Using Decode and Case

© 2004 SkillBuilders, Inc.

DECODE Workshop

1. Create a report that shows number of customers per state. Use the SUM and DECODE functions to format the report similar to:

```
        NJ         NY         RI
---------- ---------- ----------
         2         10          1
```

2. Add an "OTHER" and "TOTAL" column to the report:

```
        NJ         NY         RI      OTHER      TOTAL
---------- ---------- ---------- ---------- ----------
         2         10          1          1         14
```

3. Use `DECODE` and `SIGN` functions on the `ORD` table to create a report similar to:

```
SIZE_ORDER   NUM_ORDERS
------------ ----------
Large Order           5
Medium Order          1
Small order           6
```

Assume that a $30 order (`total_order_price`) is a "Medium Order".
Hint: You will need to repeat the `DECODE` function in the `GROUP BY` clause.

`CASE` Workshop

1. Use the `CASE` expression and `SUM` function (in the `CASE "THEN"`) to create the following report.

The `SUM_SALARY` column is the sum of the salary of all employees with the same job title by department.

```
   DEPT_NO TITLE                SUM_SALARY
---------- -------------------- ----------
         1 Designer                     59
         2 Sales Clerk                  17
         2 Sales Manager                 8
         3 Accountant                   10
         3 Sales Representative         25
```

Hint: `GROUP BY dept_no, title`

2. Use the CASE expression and SUM function to create an order price report similar to :

```
PAY_METHOD    SMALL_ORDER MEDIUM_ORDER LARGE_ORDER
------------- ----------- ------------ -----------
CASH                    3            0           2
DISCOVER                0            0           1
VISA GOLD               1            1           0
VISA STANDARD           2            0           2
```

Hints: SUM the result of the CASE expressions. GROUP BY the payment_method column.