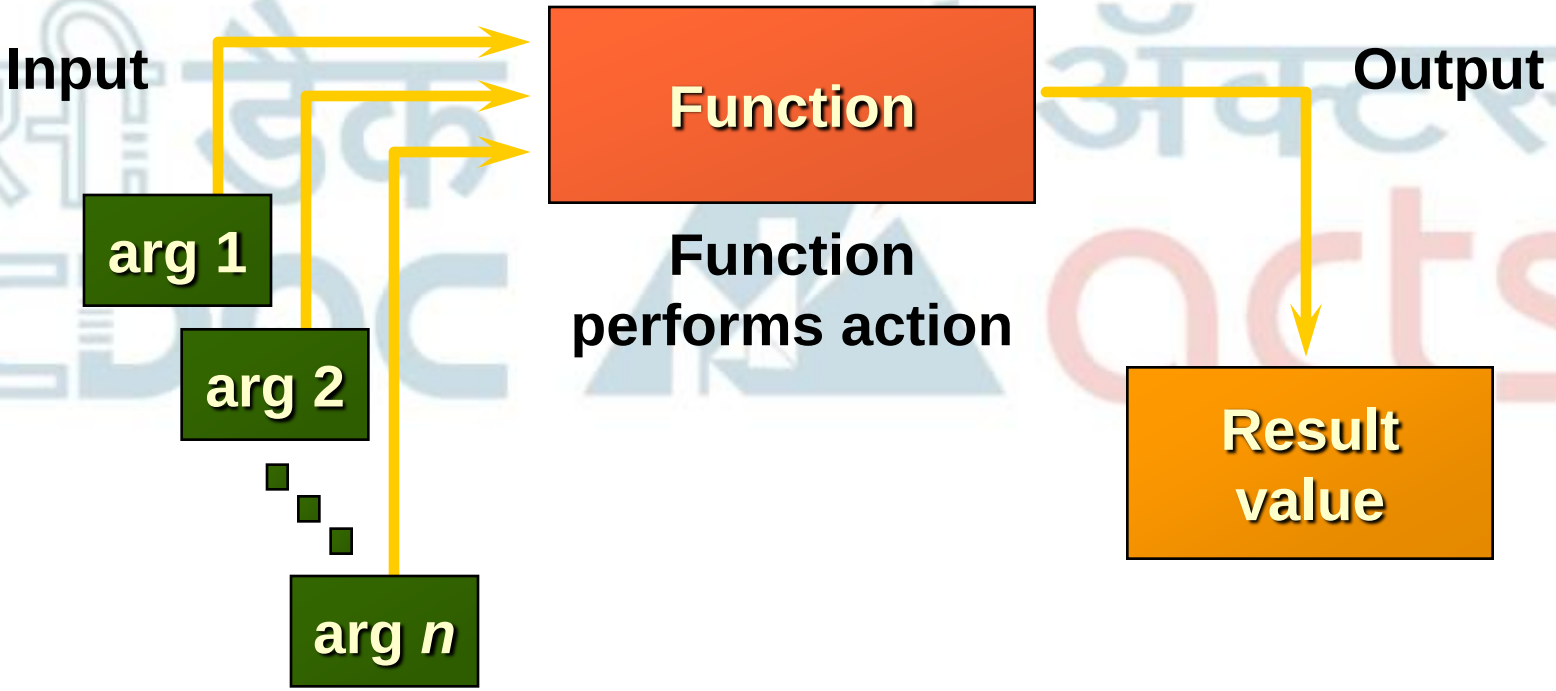
Single Row functions

- Jayendra Khatod

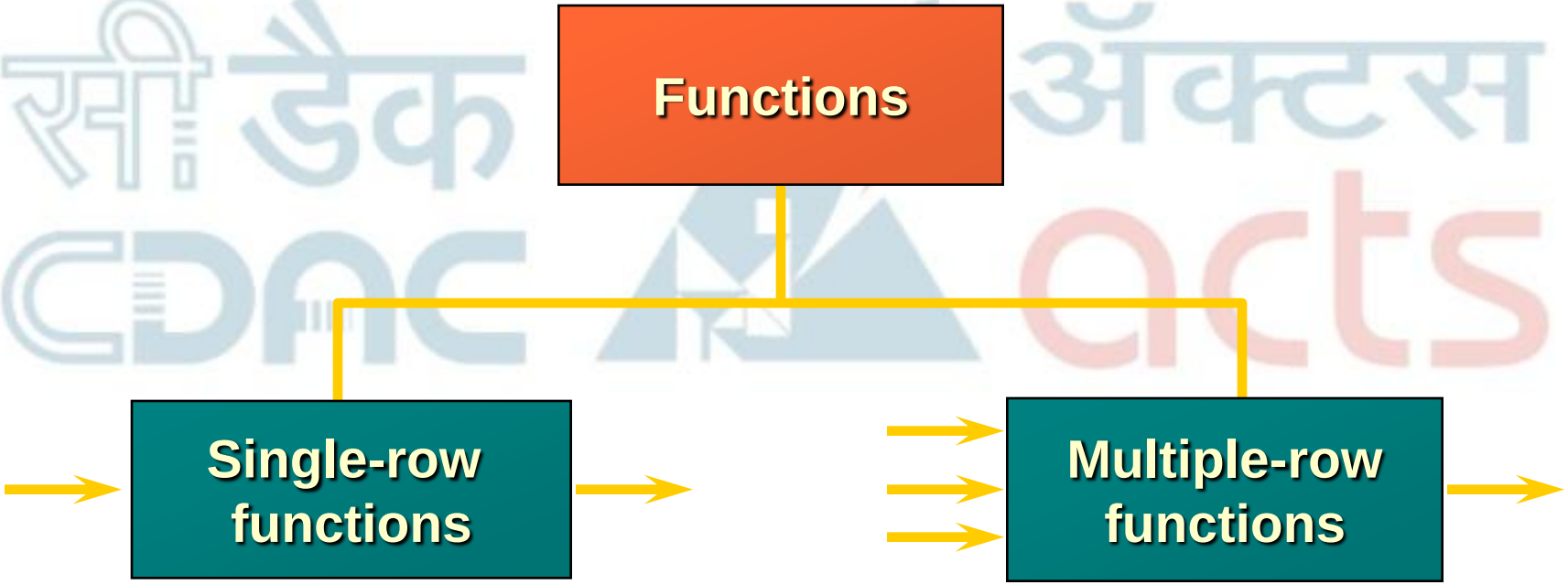
Objectives

- **Describe various types of functions available in SQL**
- **Use character, number, and date functions in `SELECT` statements**
- **Describe the use of conversion functions**

SQL Functions



Two Types of SQL Functions



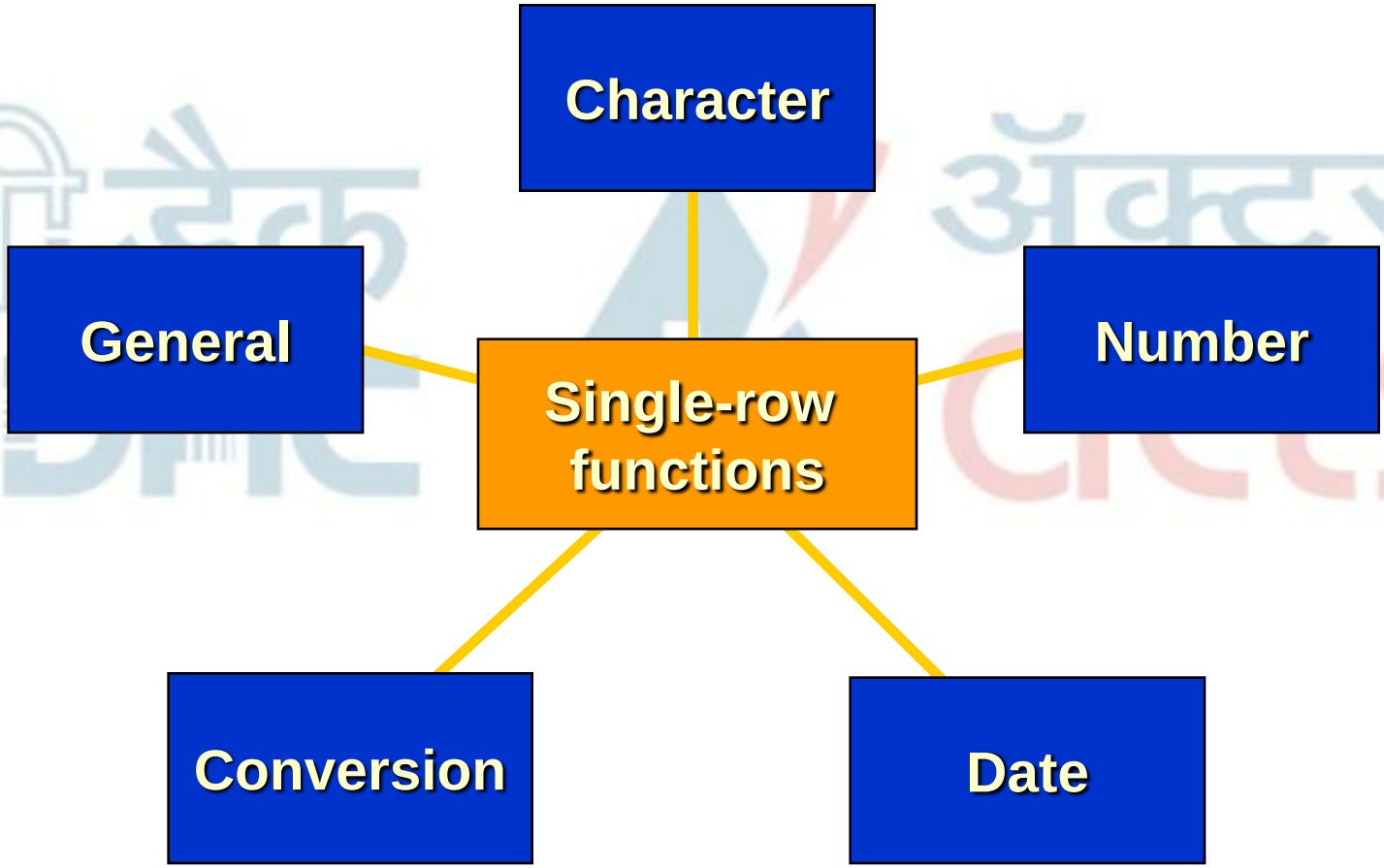
Single-Row Functions

Single row functions:

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

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

Single-Row Functions



Character Functions

Character functions

Case-manipulation functions

LOWER
UPPER
INITCAP

Character-manipulation functions

CONCAT
SUBSTR
LENGTH
INSTR
LPAD | RPAD
TRIM
REPLACE

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`

Character-Manipulation Functions

These functions manipulate character strings:

Function	Result
CONCAT('Raj', 'Menon')	RajMenon
SUBSTR('Salesman',1,5)	Sales
LENGTH('RajMenon')	10
INSTR('RajMenon', 'M')	4
LPAD(salary,10,'*')	*****27000
RPAD(salary, 10, '*')	27000*****
TRIM('H' FROM 'HelloWorld')	elloWorld

Number Functions

- **ROUND: Rounds value to specified decimal**

  ROUND(45.926, 2) → 45.93

- **TRUNC: Truncates value to specified decimal**

  TRUNC(45.926, 2) → 45.92

- **MOD: Returns remainder of division**

  MOD(1600, 300) → 100

Using the MOD Function

Calculate the remainder of a salary after it is divided by 5000 for all employees whose job title is sales representative.

```
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

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.**

Using Arithmetic Operators with Dates

```
SELECT last_name, (SYSDATE-hire_date)/7 AS WEEKS
FROM   employees
WHERE  department_id = 90;
```

LAST_NAME	WEEKS
King	744.245395
Kochhar	626.102538
De Haan	453.245395

Date Functions

Function	Description
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

Using Date Functions

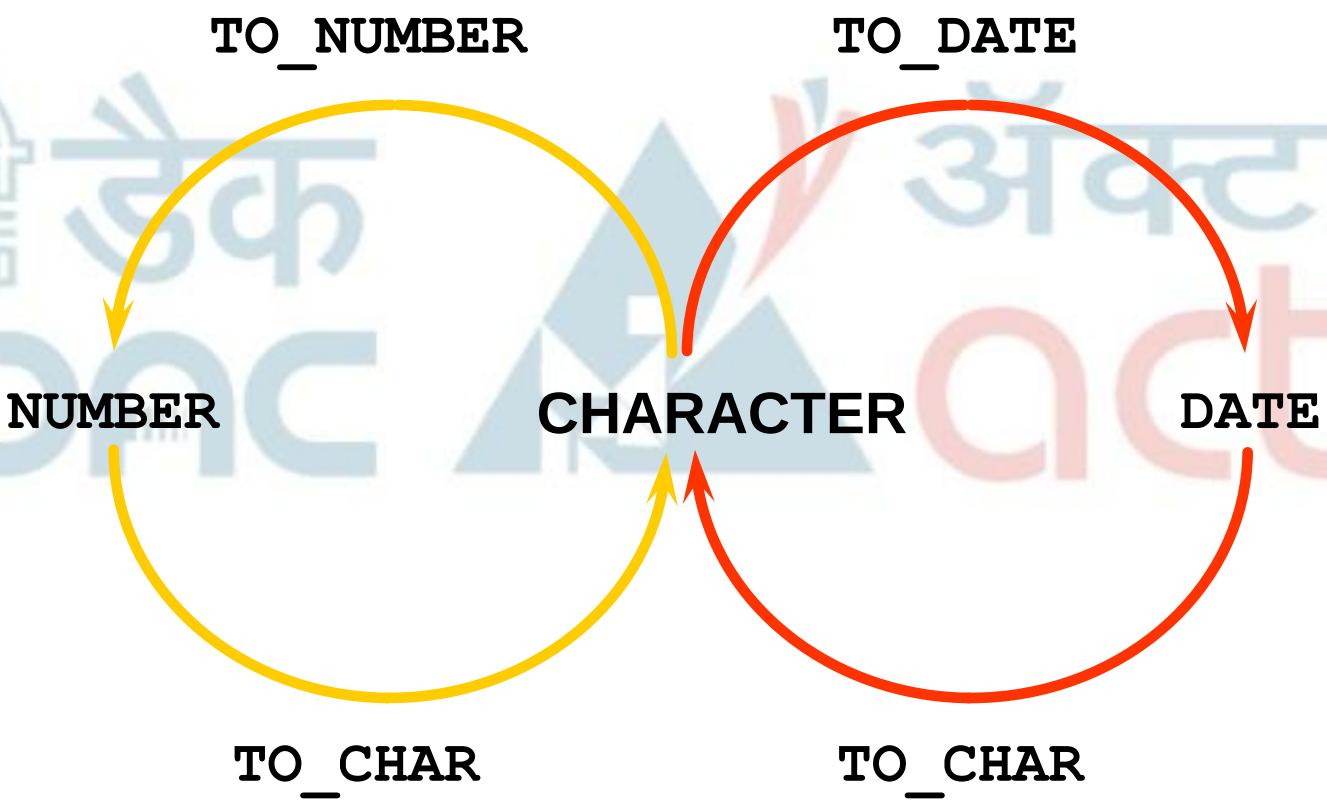
- MONTHS_BETWEEN ('01-SEP-95','11-JAN-94') → 19.6774194
- ADD_MONTHS ('11-JAN-94',6) → '11-JUL-94'
- NEXT_DAY ('01-SEP-95','FRIDAY') → '08-SEP-95'
- LAST_DAY('01-FEB-95') → '28-FEB-95'

Using Date Functions

Assume `SYSDATE = '25-JUL-95'`:

- `ROUND(SYSDATE,'MONTH')` → `01-AUG-95`
- `ROUND(SYSDATE ,'YEAR')` → `01-JAN-96`
- `TRUNC(SYSDATE ,'MONTH')` → `01-JUL-95`
- `TRUNC(SYSDATE ,'YEAR')` → `01-JAN-95`

Conversion Functions



Using the TO_CHAR Function with Dates

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

YYYY	Full year in numbers
YEAR	Year spelled out
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

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

Using the TO_CHAR Function with Dates

```
SELECT last_name,
       TO_CHAR(hire_date, 'DD 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.

Using the TO_CHAR Function with Numbers

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

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

9	Represents a number
0	Forces a zero to be displayed
$	Places a floating dollar sign
.	Prints a decimal point
,	Prints a thousand indicator

Using the TO_CHAR Function with Numbers

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

SALARY
$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'])
```

General Functions

These 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)`

NVL Function

Converts a null to an actual value.

- **Data types that can be used are date, character, and number.**
- **Data types must match:**
  - `NVL(commission_pct,0)`
  - `NVL(hire_date,'01-JAN-97')`
  - `NVL(job_id,'No Job Yet')`

Using the NVL Function

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

LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
King	24000	0	288000
Kochhar	17000	0	204000
De Haan	17000	0	204000
Hunold	9000	0	108000
Ernst	6000	0	72000
Lorentz	4200	0	50400
Mourgos	5800	0	69600
Rajs	3500	0	42000

...

20 rows selected.

Using the NVL2 Function

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

LAST_NAME	SALARY	COMMISSION_PCT	INCOME
Zlotkey	10500	.2	SAL+COMM
Abel	11000	.3	SAL+COMM
Taylor	8600	.2	SAL+COMM
Mourgos	5800		SAL
Rajs	3500		SAL
Davies	3100		SAL
Matos	2600		SAL
Vargas	2500		SAL

8 rows selected.

Using the NULLIF Function

```
SELECT first_name, LENGTH(first_name) "expr1",
       last_name,  LENGTH(last_name)  "expr2",
       NULLIF(LENGTH(first_name), LENGTH(last_name)) result
FROM   employees;
```

FIRST_NAME	expr1	LAST_NAME	expr2	RESULT
Steven	6	King	4	6
Neena	5	Kochhar	7	5
Lex	3	De Haan	7	3
Alexander	9	Hunold	6	9
Bruce	5	Ernst	5	
Diana	5	Lorentz	7	5
Kevin	5	Mourgos	7	5
Trenna	6	Rajs	4	6
Curtis	6	Davies	6	

...

20 rows selected.

Using the COALESCE Function

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

Using the COALESCE Function

```
SELECT    last_name,
          COALESCE(commission_pct, salary, 10) comm
FROM      employees
ORDER BY  commission_pct;
```

LAST_NAME	COMM
Grant	.15
Zlotkey	.2
Taylor	.2
Abel	.3
King	24000
Kochhar	17000
De Haan	17000
Hunold	9000

...

20 rows selected.

Conditional Expressions

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

The 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

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
...			
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
...			
Gietz	AC_ACCOUNT	8300	8300

20 rows selected.

The DECODE Function

Facilitates conditional inquiries by doing the work of a CASE or 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
...			
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
...			
Gietz	AC_ACCOUNT	8300	8300

20 rows selected.

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;
```

Summary

In this lesson, you should have learned how to:

- **Perform calculations on data using functions**
- **Modify individual data items using functions**
- **Manipulate output for groups of rows using functions**
- **Alter date formats for display using functions**
- **Convert column data types using functions**
- **Use NVL functions**
- **Use IF-THEN-ELSE logic**

Thank You !