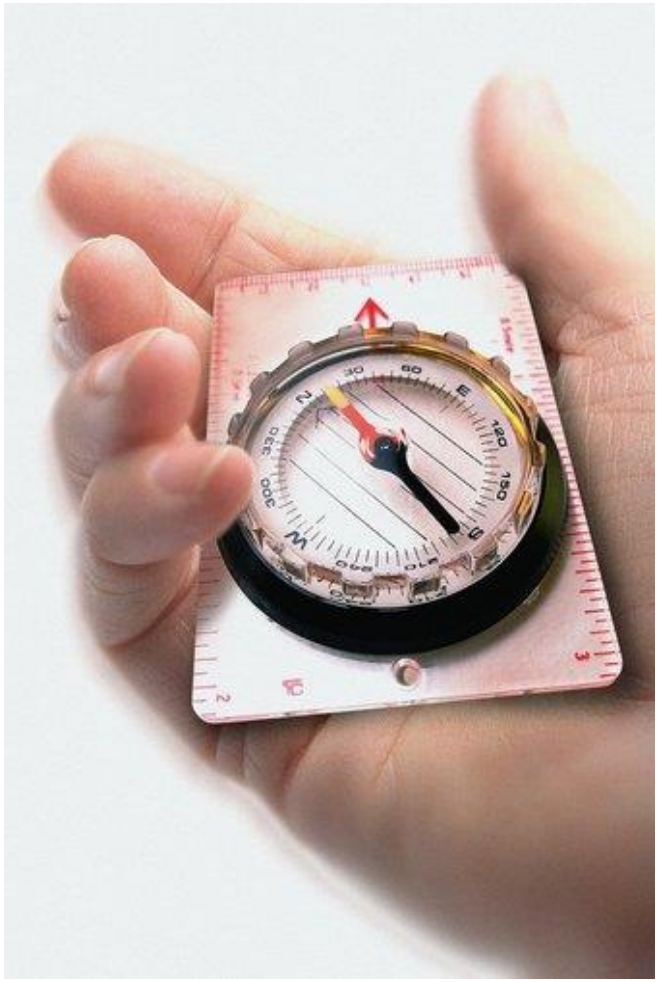
Functions in SQL

Course objectives

By completing this course, you will be able to:



- **Describe various types of functions that are available in SQL**
- **Describe the use of group functions**

Course topics

Course's plan:

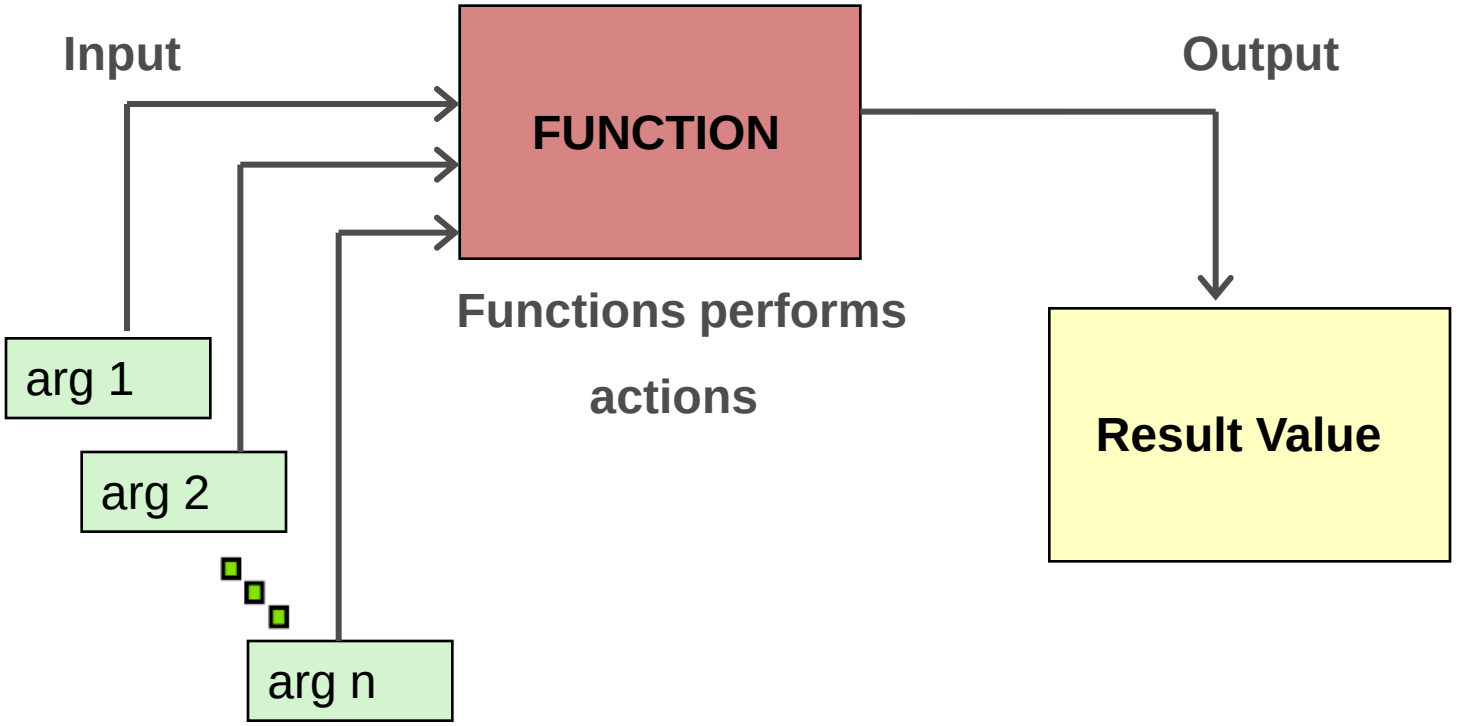
- Using Single-Row Functions to Customize Output
- Reporting Aggregated Data Using the Group Functions

Single-Row Functions

Preview

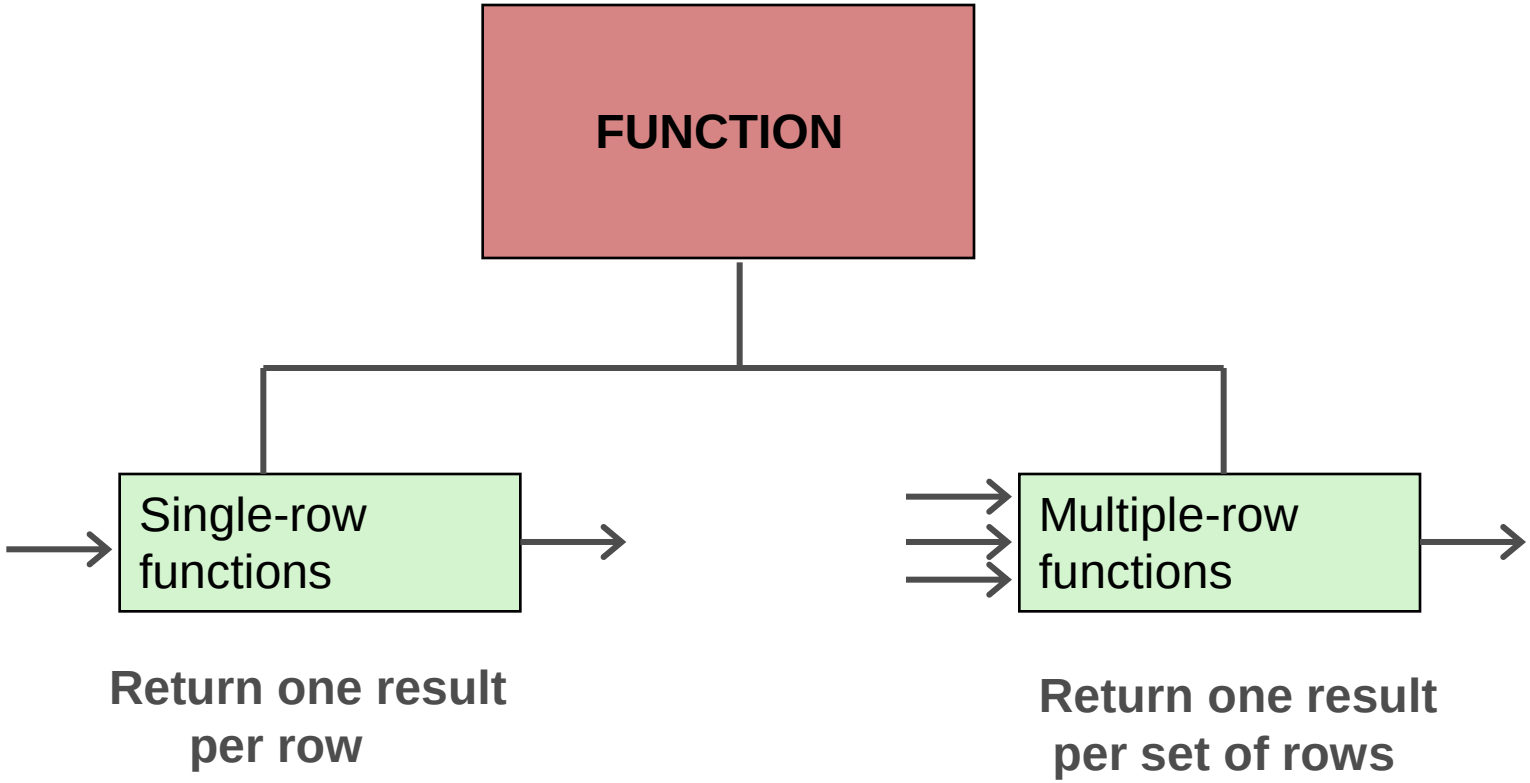
- SQL Functions.
- Character Functions.
- Number Functions.
- Date Functions.
- Conversion Functions.
- General Functions.
- Conditional Expressions

SQL Functions



SQL Functions

Two Types of SQL Functions



SQL Functions

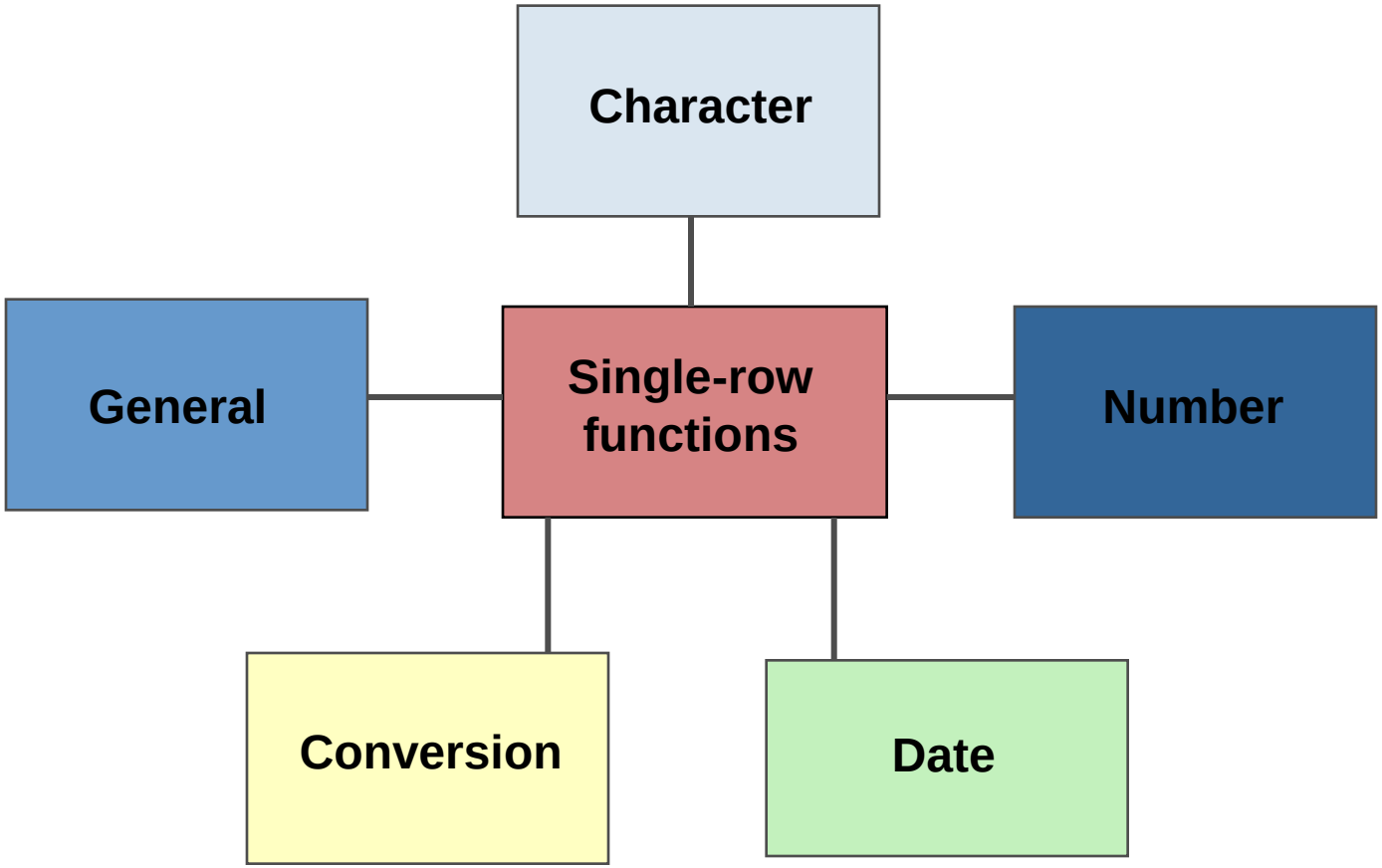
Single-row functions:

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

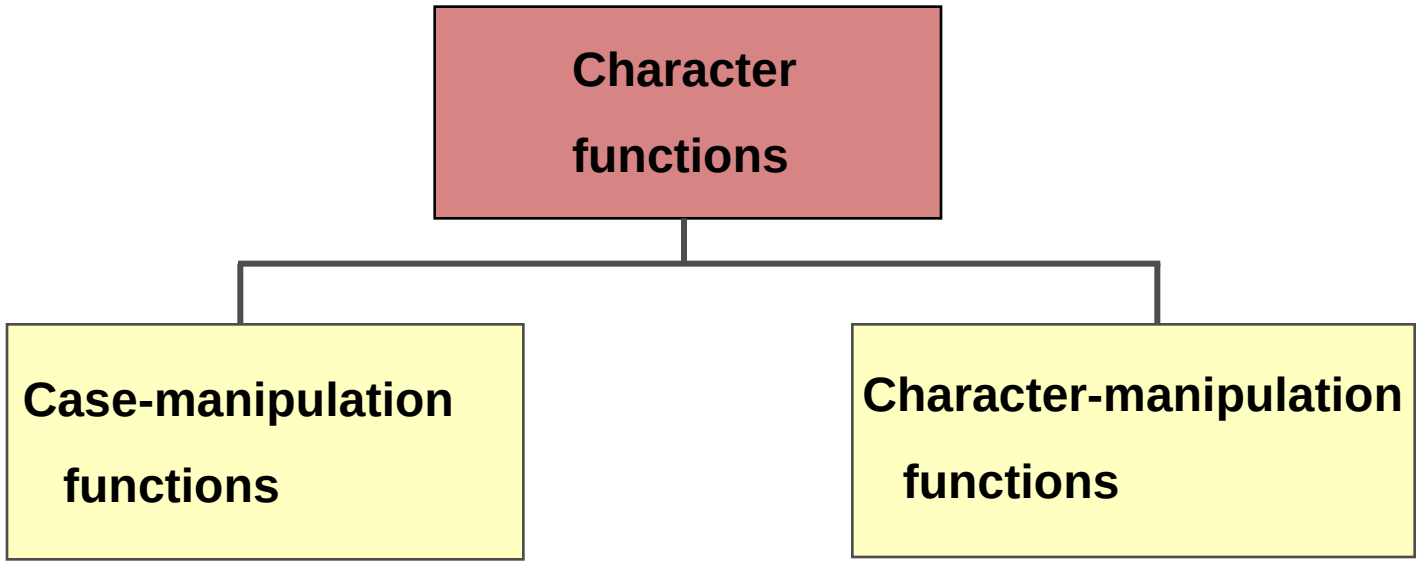
```
function_name [(arg1, arg2,...)]
```

SQL Functions

Single-row functions:



Character Functions



LOWER

UPPER

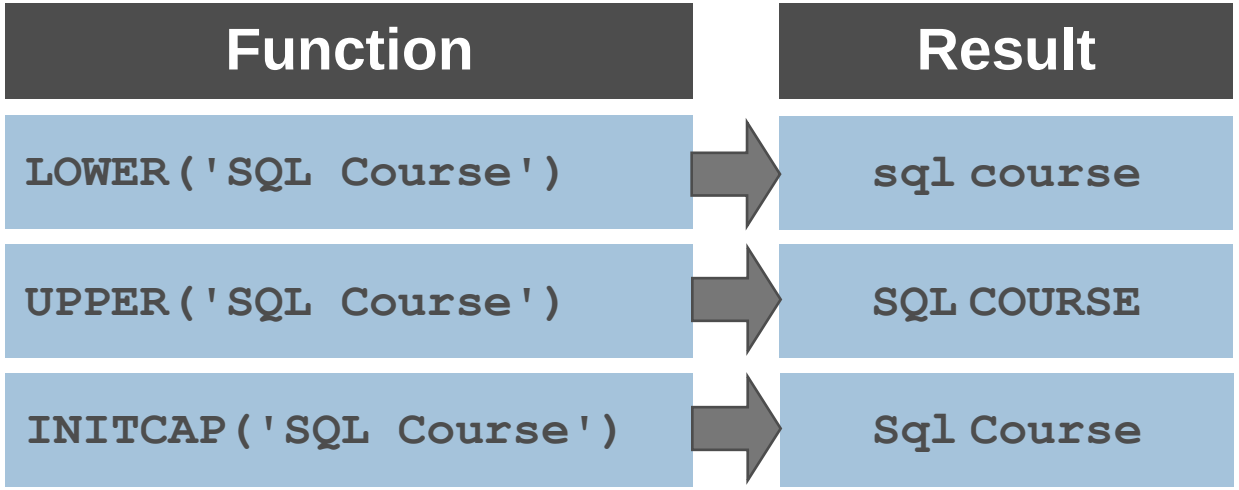
INITCAP

CONCAT
SUBSTR
LENGTH
INSTR
LPAD | RPAD
TRIM
REPLACE

Character Functions

Using Case-Manipulation Functions

- These functions convert case for character strings:



Character Functions

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
```

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  LOWER(last_name) = 'higgins';
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
205	Higgins	110

Character Functions

These functions manipulate character strings:

Function	Result
`CONCAT('Hello', 'World')`	HelloWorld
`SUBSTR('HelloWorld',6,5)`	World
`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

Character Functions

Using the Character-Manipulation Functions

```
SELECT employee_id, CONCAT(first_name, last_name)
       NAME,job_id, LENGTH (last_name),
       INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(job_id, 4) = 'REP';
```

EMPLOYEE_ID	NAME	JOB_ID	LENGTH(LAST_NAME)	Contains 'a'?
174	EllenAbel	SA_REP	4	0
176	JonathonTaylor	SA_REP	6	2
178	KimberelyGrant	SA_REP	5	3
202	PatFay	MK_REP	3	2

Character Functions

Using SUBSTR Function with different arguments:

```
SELECT SUBSTR('Hello World', 4)
FROM DUAL;
```

SUBSTR('HELLOWORLD',4)
lo World

```
SELECT SUBSTR('Hello World', -4)
FROM DUAL;
```

SUBSTR('HELL
orld

```
SELECT SUBSTR('Hello World', -4,3)
FROM DUAL;
```

SUBSTR('H
orl

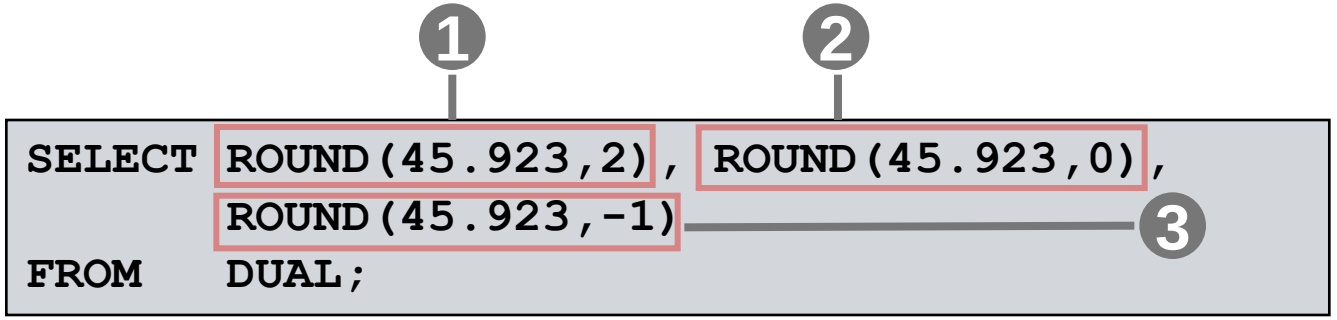
Number Functions

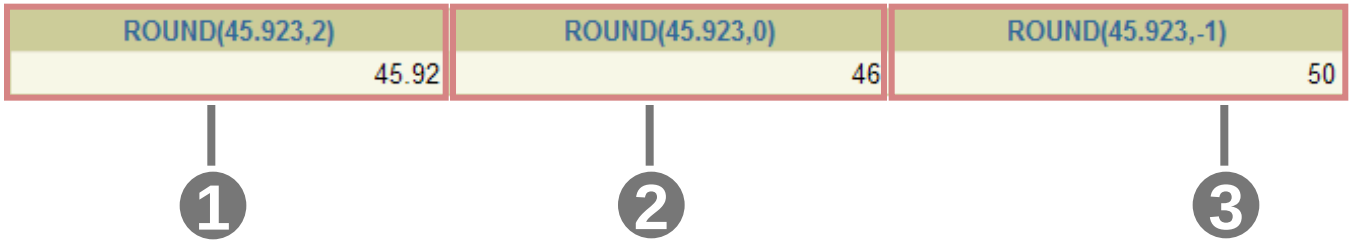
- **ROUND**: Rounds value to specified decimal
- **TRUNC**: Truncates value to specified decimal
- **MOD**: Returns remainder of division

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

Number Functions

Using the ROUND Function

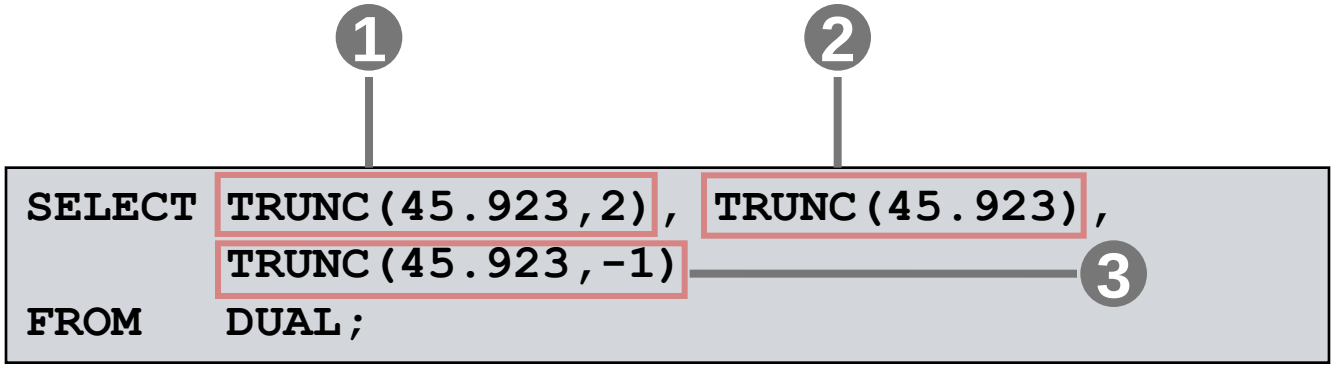




- **DUAL** is a dummy table that you can use to view results from functions and calculations.

Number Functions

Using the TRUNC Function



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

1 2 3

Number Functions

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

Date Functions

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

Date Functions

Working with Dates

- **`SYSDATE`** is a function that returns:
  - Date
  - Time

Date Functions

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.

Date Functions

Using Arithmetic Operators with Dates

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

LAST_NAME	WEEKS
King	1003.82454
Kochhar	885.681678
De Haan	712.824535

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

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'`

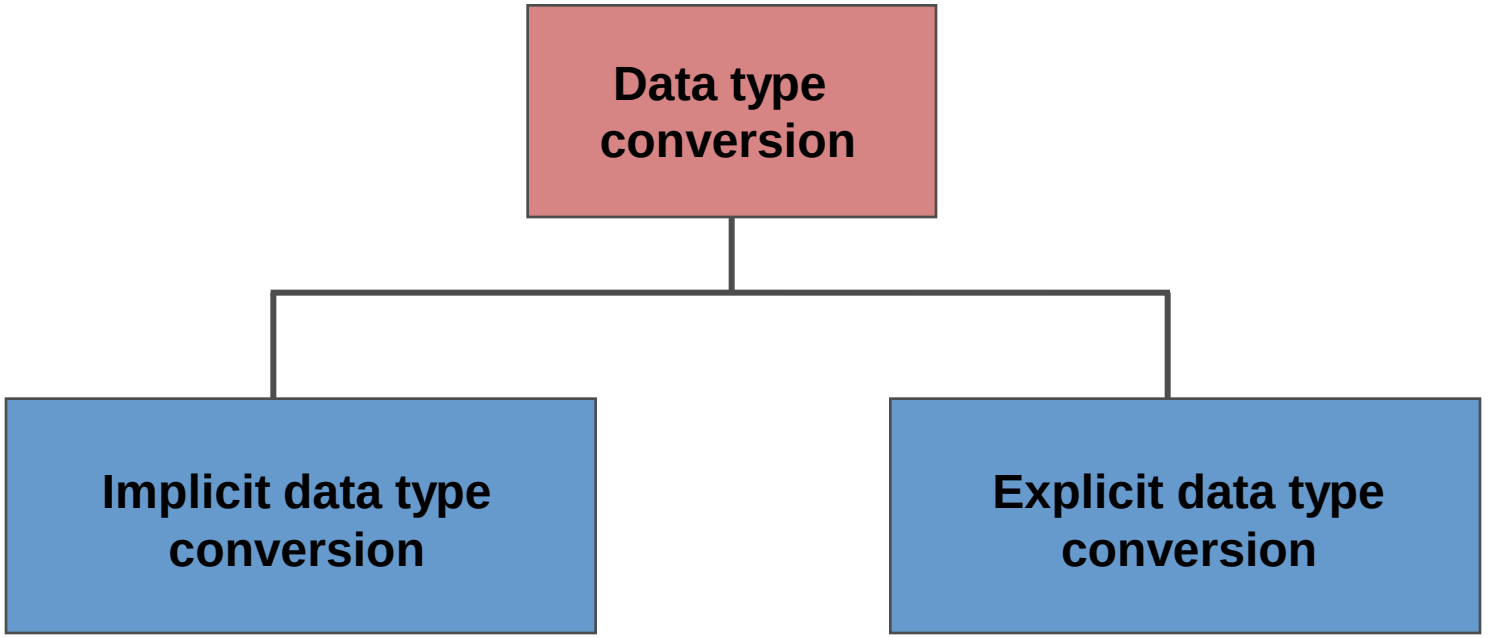
Date Functions

Example

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`

Conversion Functions



Conversion Functions

Implicit Data Type Conversion

- For assignments, the Oracle server can automatically convert the following:

From	To
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

Conversion Functions

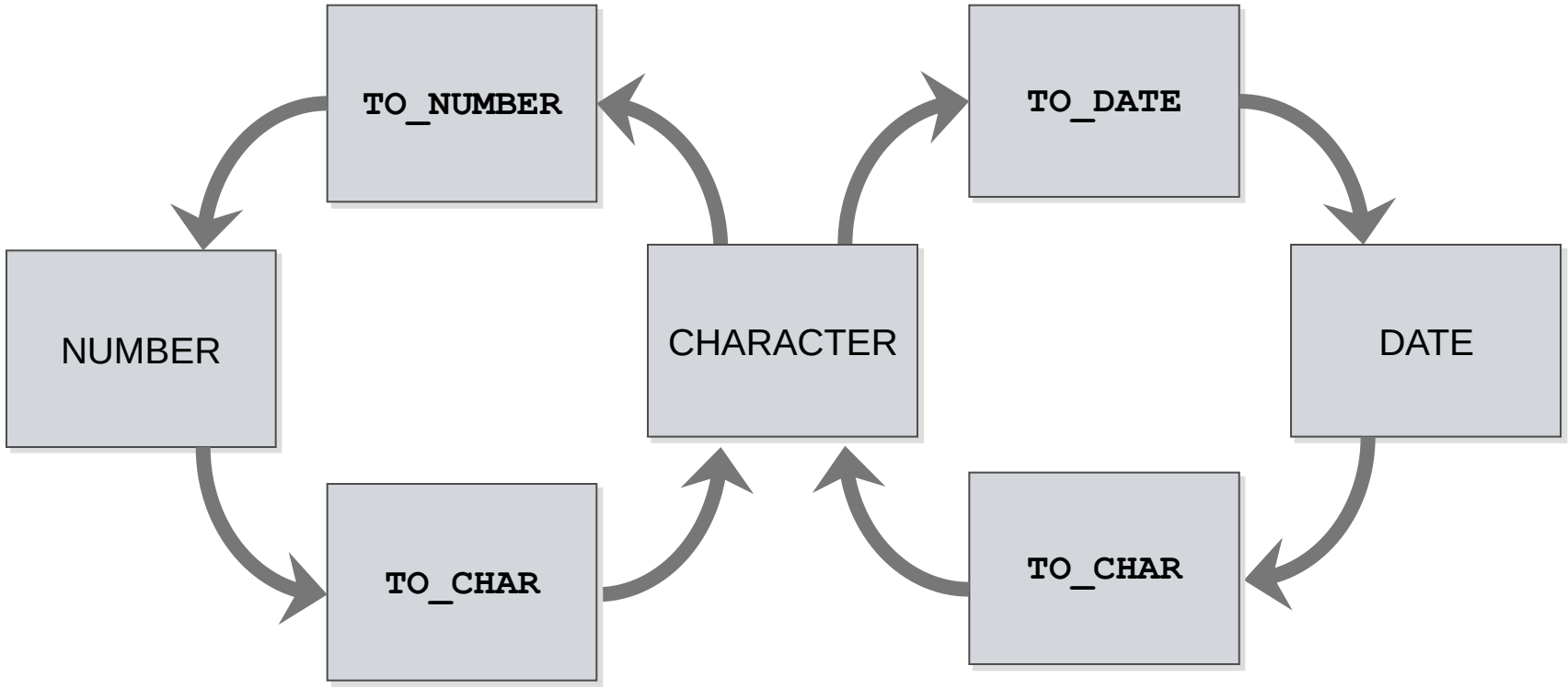
Implicit Data Type Conversion

- For expression evaluation, the Oracle Server can automatically convert the following:

From	To
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE

Conversion Functions

Explicit Data Type Conversion



Conversion Functions

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

Conversion Functions

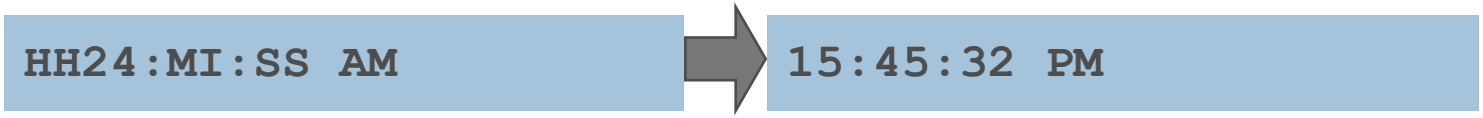
Elements of the Date Format Model

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
D	Numeric day of the week

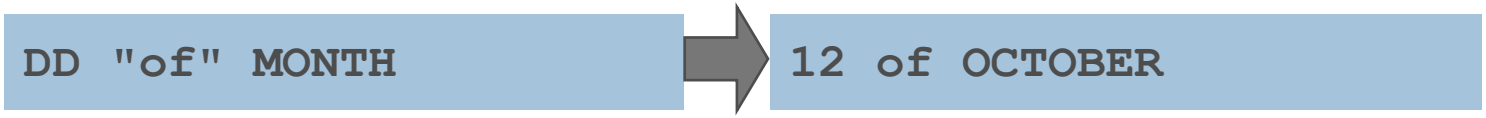
Conversion Functions

Elements of the Date Format Model

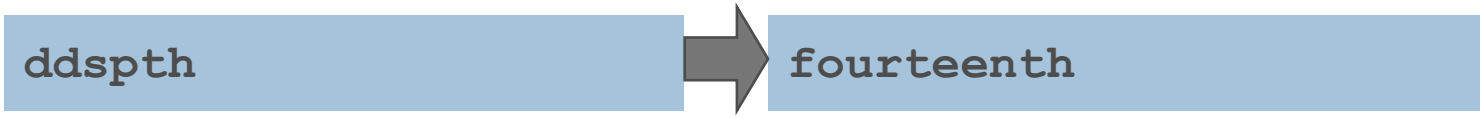
- Time elements format the time portion of the date:



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



- Number suffixes spell out numbers:



Conversion Functions

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.

Conversion Functions

Using the TO_CHAR Function with Numbers

```
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 thousands indicator

Conversion Functions

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

Conversion Functions

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'])
```

- These functions have an **fx** modifier. This modifier specifies the exact matching for the character argument and date format model of a **TO_DATE** function.

Conversion Functions

RR Date Format

		If the specified two-digit year is:	
		0-49	50-99
If two digits of the current year are:	0-49	The return date is in the current century	The return date is in the century before the current one
	50-99	The return date is in the century after the current one	The return date is in the current century

Conversion Functions

RR Date Format

Current Year	Specified Date	RR Format	YY Format
1995	27-OCT-95	1995	1995
1995	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017
2001	27-OCT-95	1995	2095

Conversion Functions

Example of RR Date Format

- To find employees hired prior to 1990, use the **RR** date format, which produces the same results whether the command is run in 1999 or now:

```
SELECT last_name,TO_CHAR(hire_date,'DD-Mon-YYYY')
FROM   employees
WHERE  hire_date < TO_DATE('01-Jan-90','DD-Mon-RR');
```

LAST_NAME	TO_CHAR(HIRE_DATE,'DD-MON-YYYY')
King	17-Jun-1987
Kochhar	21-Sep-1989
Whalen	17-Sep-1987

Conversion Functions

Nesting Functions

- Single-row functions can be nested to any level.
- Nested functions are evaluated from deepest level to the least deep level.

Conversion Functions

Nesting Functions

```
SELECT last_name,
       UPPER(CONCAT(SUBSTR(LAST_NAME,1,8),'_US'))
FROM   employees
WHERE  department_id = 60;
```

LAST_NAME	UPPER(CONCAT(SUBSTR(LAST_NAME,1,8),
Hunold	HUNOLD_US
Ernst	ERNST_US
Austin	AUSTIN_US
Pataballa	PATABALL_US
Lorentz	LORENTZ_US

General Functions

- The following functions work with any data type and pertain to using nulls:
  - **`NVL (expr1, expr2)`**
  - **`NVL2 (expr1, expr2, expr3)`**

General Functions

NVL Function

- Converts a null value 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')`

General Functions

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
Russell	14000	.4	235200
Partners	13500	.3	210600
Errazuriz	12000	.3	187200
Cambrault	11000	.3	171600

General Functions

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

Conditional Expressions

Conditional Expressions

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

Conditional Expressions

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
```

Conditional Expressions

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 "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.

Conditional Expressions

DECODE Function

- 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])
```

Conditional Expressions

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.

Conditional Expressions

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;
```

Part 1 Summary

SQL Functions

Number Function

Character Function

Conversion Function

Global Function

Expression of condition

Group Functions

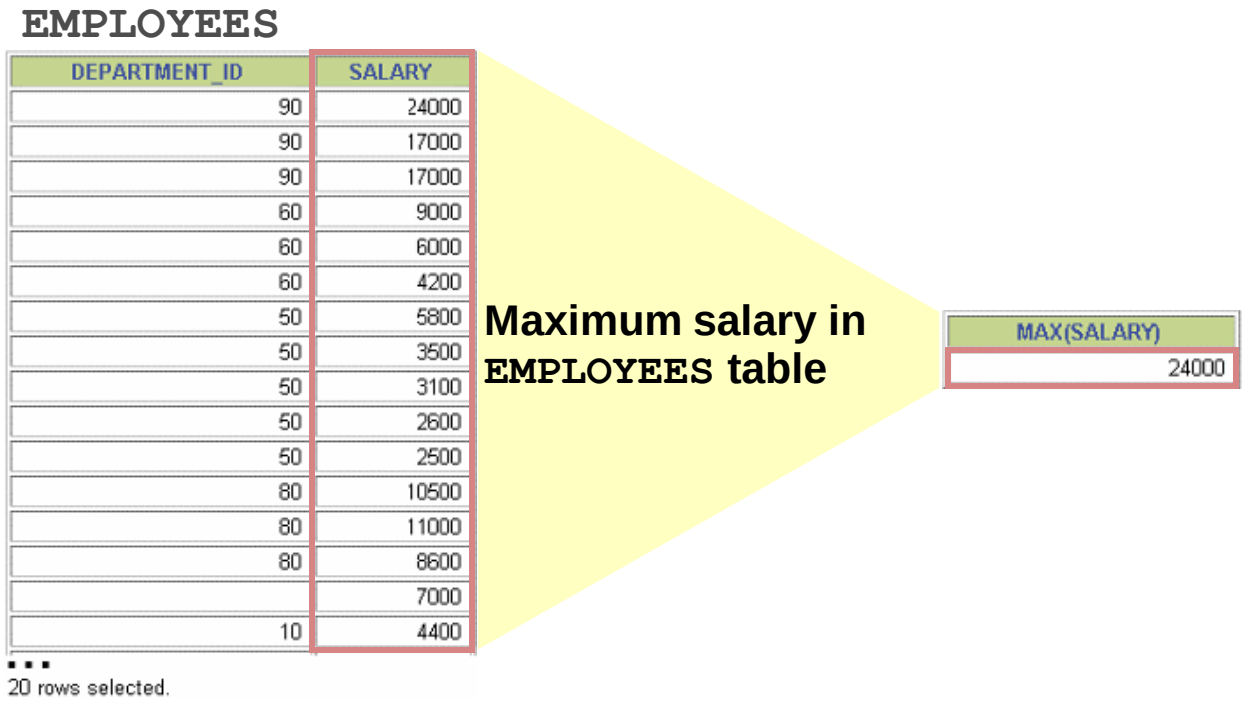
Preview

- Presentation
- Creating groups
- Restricting Group Results

Presentation

What Are Group Functions?

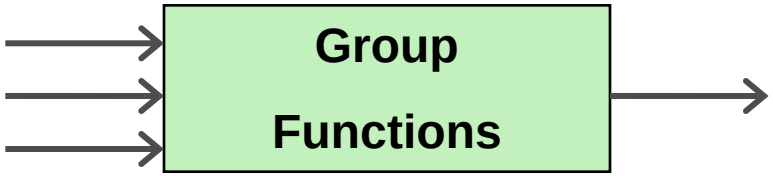
- Group functions operate on sets of rows to give one result per group.



Presentation

Types of Group Functions

- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE



Presentation

Group Functions: Syntax

```
 SELECT   [column,] group_function(column), ...
 FROM      table
[WHERE     condition]
[GROUP BY column]
[ORDER BY column];
```

Presentation

You can use AVG and SUM for numeric data.

```
SELECT  AVG(salary), MAX(salary),
        MIN(salary), SUM(salary)
FROM    employees
WHERE   job_id LIKE '%REP%';
```

AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
8150	11000	6000	32600

Presentation

You can use `MIN` and `MAX` for numeric, character, and date data types.

```
SELECT MIN(hire_date), MAX(hire_date)
FROM   employees;
```

MIN(HIRE_DATE)	MAX(HIRE_DATE)
17-JUN-87	29-JAN-00

Presentation

Using the COUNT Function

- **COUNT(*)** returns the number of rows in a table:

```
SELECT COUNT(*)
FROM   employees
WHERE  department_id = 50;
```

COUNT(*)
5

- **COUNT(expr)** returns the number of rows with non null values for the **expr**:

```
SELECT COUNT(commission_pct)
FROM   employees
WHERE  department_id = 80;
```

COUNT(COMMISSION_PCT)
3

Presentation

Using the DISTINCT Keyword

- COUNT(DISTINCT expr) returns the number of distinct non-null values of the expr.
- To display the number of distinct department values in the EMPLOYEES table:

```
SELECT COUNT(DISTINCT department_id)
FROM   employees;
```

COUNT(DISTINCTDEPARTMENT_ID)
7

Presentation

Group Functions and Null Values

- Group functions ignore null values in the column:

```
SELECT AVG(commission_pct)
FROM   employees;
```

AVG(COMMISSION_PCT)
.2125

- The **NVL** function forces group functions to include null values:

```
SELECT AVG(NVL(commission_pct, 0))
FROM   employees;
```

AVG(NVL(COMMISSION_PCT,0))
.0425

Creating Groups

Creating Groups of Data

EMPLOYEES

DEPARTMENT_ID	SALARY
10	4400
20	13000
20	6000
50	5800
50	3500
50	3100
50	2500
50	2600
60	9000
60	6000
60	4200
80	10500
80	8600
80	11000
90	24000
90	17000

. . .

20 rows selected.

4400

9500

3500

6400

10033

Average salary in EMPLOYEES table for each department

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333
90	19333.3333
110	10150
	7000

Creating Groups

GROUP BY Clause Syntax

```
 SELECT    column, group_function(column)
 FROM      table
[WHERE     condition]
[GROUP BY group_by_expression]
[ORDER BY column];
```

- You can divide rows in a table into smaller groups by using the **GROUP BY** clause.

Creating Groups

Using the GROUP BY Clause

- All columns in the **SELECT** list that are not in group functions must be in the **GROUP BY** clause.

```
SELECT    department_id , AVG(salary)
FROM      employees
GROUP BY department_id;
```

DEPARTMENT_ID	AVG(SALARY)
	7000
90	19333.3333
20	9500
110	10150
50	3500
80	10033.3333
60	6400
10	4400

Creating Groups

Using the GROUP BY Clause

- The **GROUP BY** column does not have to be in the **SELECT** list.

```
SELECT   AVG(salary)
FROM     employees
GROUP BY department_id ;
```

AVG(SALARY)
7000
19333.3333
9500
10150
3500
10033.3333
6400
4400

Creating Groups

Grouping by More Than One Column

EMPLOYEES

DEPARTMENT_ID	JOB_ID	SALARY
90	AD_PRES	24000
90	AD_VP	17000
90	AD_VP	17000
60	IT_PROG	9000
60	IT_PROG	6000
60	IT_PROG	4200
50	ST_MAN	5800
50	ST_CLERK	3500
50	ST_CLERK	3100
50	ST_CLERK	2600
50	ST_CLERK	2500
80	SA_MAN	10500
80	SA_REP	11000
80	SA_REP	8600
...		
20	MK_REP	6000
110	AC_MGR	12000
110	AC_ACCOUNT	8300

20 rows selected.

Add the salaries in the EMPLOYEES table for each job, grouped by department.

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
20	MK_MAN	13000
20	MK_REP	6000
50	ST_CLERK	11700
50	ST_MAN	5800
60	IT_PROG	19200
80	SA_MAN	10500
80	SA_REP	19600
90	AD_PRES	24000
90	AD_VP	34000
110	AC_ACCOUNT	8300
110	AC_MGR	12000
	SA_REP	7000

13 rows selected.

Creating Groups

Using the GROUP BY Clause on Multiple Columns

```
SELECT   department_id dept_id, job_id, SUM(salary)
FROM     employees
GROUP BY department_id, job_id ;
```

DEPT_ID	JOB_ID	SUM(SALARY)
110	AC_ACCOUNT	8300
90	AD_VP	34000
50	ST_CLERK	11700
80	SA_REP	19600
50	ST_MAN	5800
80	SA_MAN	10500
110	AC_MGR	12000
90	AD_PRES	24000
60	IT_PROG	19200
20	MK_MAN	13000
	SA_REP	7000
10	AD_ASST	4400
20	MK_REP	6000

Creating Groups

Illegal Queries Using Group Functions

- Any column or expression in the **SELECT** list that is not an aggregate function must be in the **GROUP BY** clause:

```
SELECT department_id, COUNT(last_name)
FROM   employees;
```

```
SELECT department_id, COUNT(last_name)
       *
ERROR at line 1:
ORA-00937: not a single-group group function
```

Column missing in the GROUP BY clause

Creating Groups

Illegal Queries Using Group Functions

- You cannot use the **WHERE** clause to restrict groups.
- You use the **HAVING** clause to restrict groups.
- You cannot use group functions in the **WHERE** clause.

```
SELECT   department_id, AVG(salary)
FROM     employees
WHERE    AVG(salary) > 8000
GROUP BY department_id;
```

```
WHERE AVG(salary) > 8000
      *
ERROR at line 3:
ORA-00934: group function is not allowed
here
```

Cannot use the WHERE clause to restrict groups

Restricting Group Results

EMPLOYEES

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	9000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
80	11000
80	10500
80	8600
...	
20	6000
110	12000
110	8300

20 rows selected.

The maximum salary per department whien it is greater than $10,000

DEPARTMENT_ID	MAX(SALARY)
20	13000
80	11000
90	24000
110	12000

Restricting Group Results

Restricting Group Results with the HAVING Clause

- When you use the **HAVING** clause, the Oracle server restricts groups as follows:
  - Rows are grouped.
  - The group function is applied.
  - Groups matching the **HAVING** clause are displayed.

```
 SELECT    column, group_function
 FROM      table
[WHERE     condition]
[GROUP BY  group_by_expression]
[HAVING    group_condition]
[ORDER BY  column];
```

Restricting Group Results

Using the HAVING Clause

```
SELECT    department_id, MAX(salary)
FROM      employees
GROUP BY  department_id
HAVING    MAX(salary)>10000;
```

DEPARTMENT_ID	MAX(SALARY)
90	24000
20	13000
110	12000
80	11000

Restricting Group Results

Using the HAVING Clause

```
SELECT   job_id, SUM(salary) PAYROLL
FROM     employees
WHERE    job_id NOT LIKE '%REP%'
GROUP BY job_id
HAVING   SUM(salary) > 13000
ORDER BY SUM(salary);
```

JOB_ID	PAYROLL
IT_PROG	19200
AD_PRES	24000
AD_VP	34000

Restricting Group Results

Display the maximum average salary:

```
SELECT   MAX(AVG(salary))
FROM     employees
GROUP BY department_id;
```

MAX(AVG(SALARY))
19333.3333

Part 2 Summary

COUNT, MAX, MIN and AVG

Use the GROUP BY clause

Use the HAVING clause

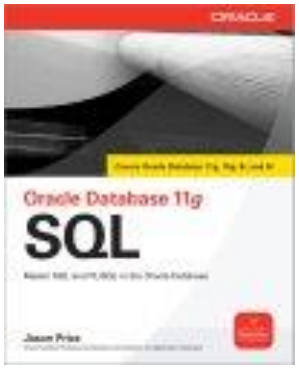
Part 3 Stop-and-think

Do you have any questions ?

For more

If you want to go into these subjects more deeply, …

Publications



http://www.oracle.../bookstore/

Courses

Cursus: Merise & SQL

Cursus: PL/SQL

Cursus: DBA1 & DBA2

Cursus: DWH & BIS

Web sites

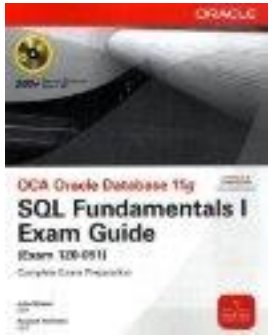
http://www.labo-oracle.com

http://www.oracle.com

http://otn.oracle.com

Certifications

1Z0-007



The end