Lab Week 02
SQL Functions and Group Functions

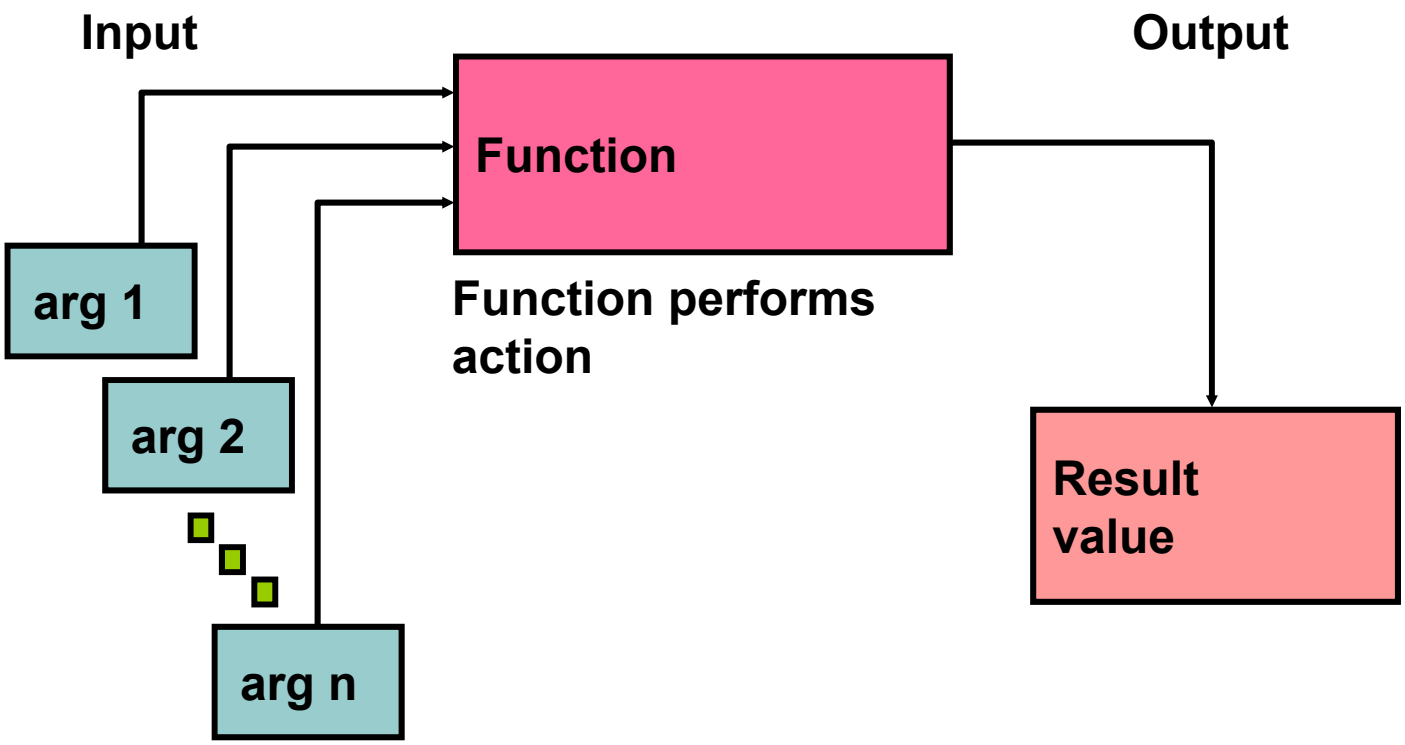
AUT
UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Objectives

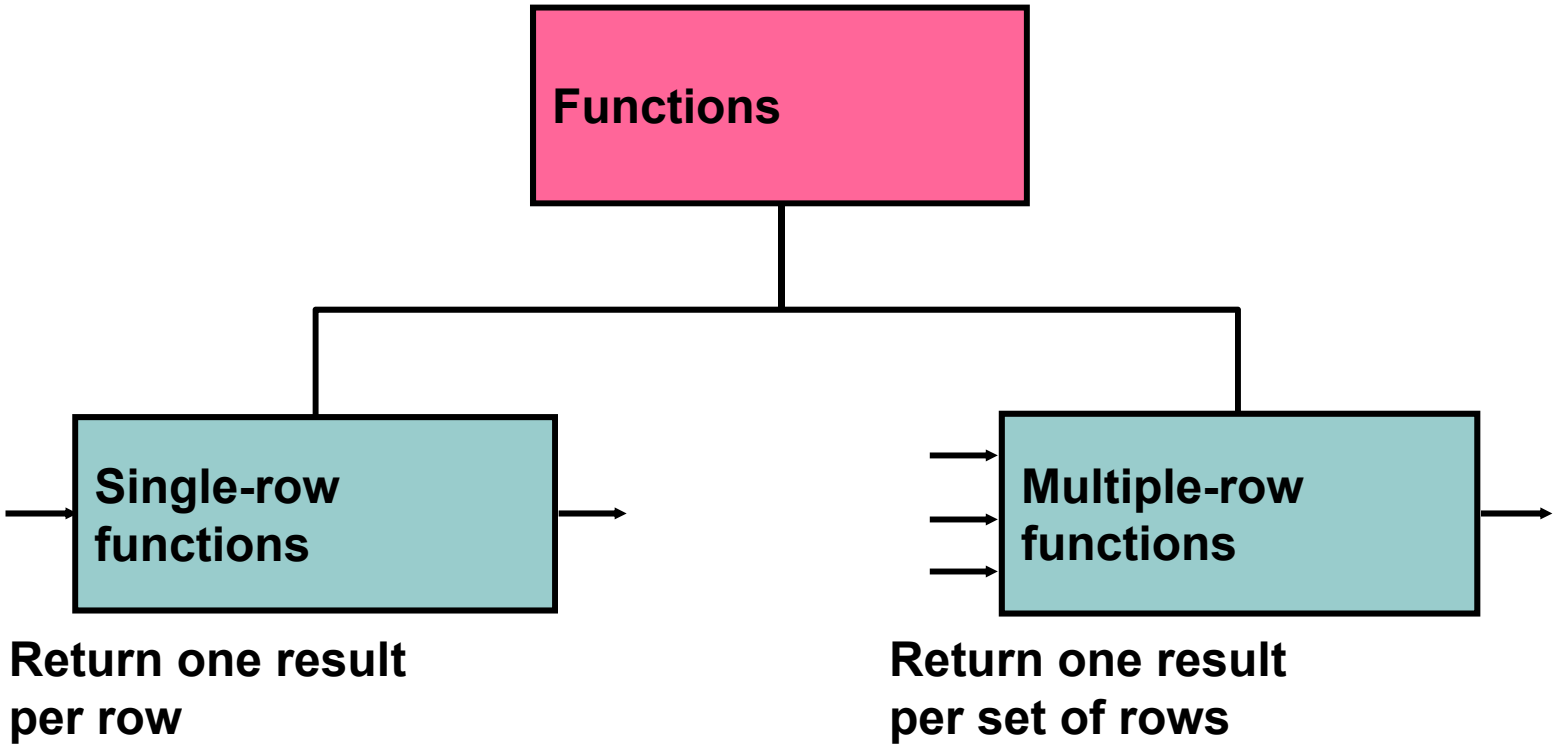
After completing this lesson, you should be able to do the following:

- Describe various types of functions that are 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



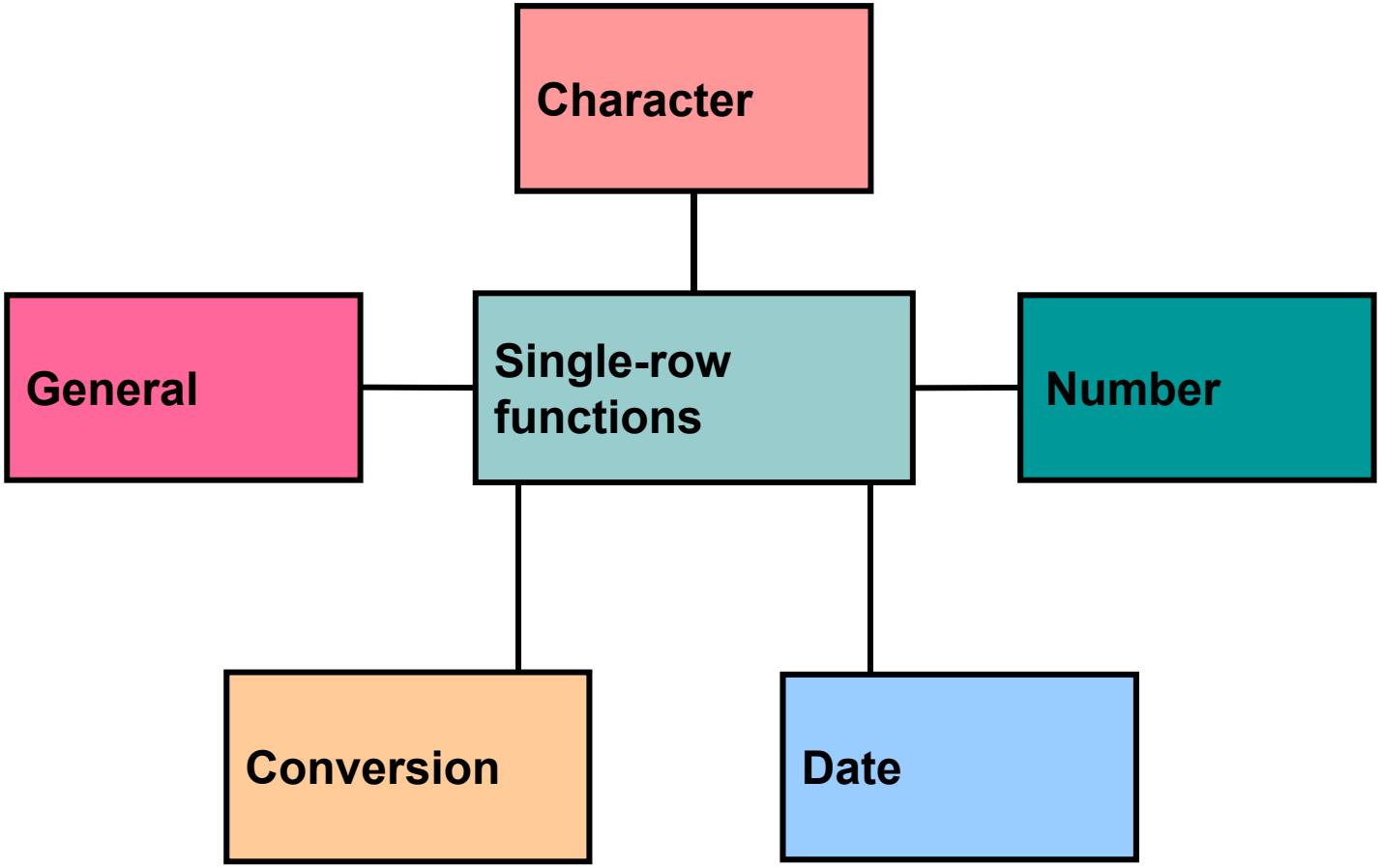
Single-Row Functions

Single-row functions:

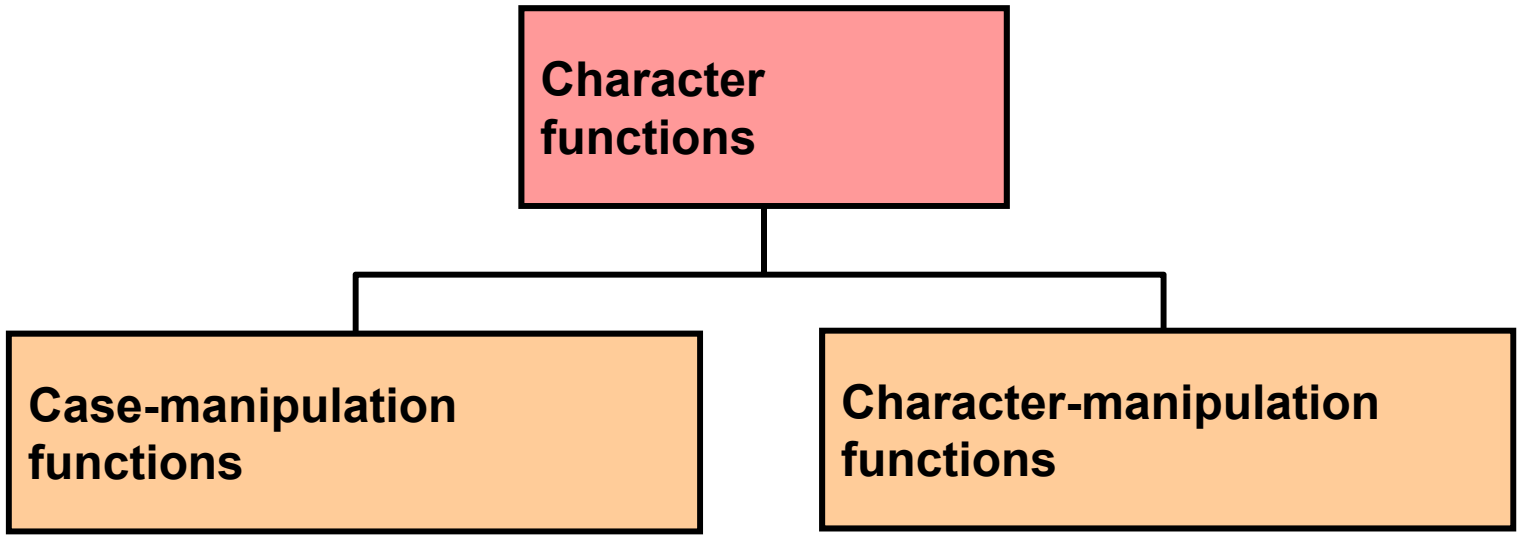
- Manipulate data items
- Accept arguments and return one value
- Act on each row that is returned
- Return one result per row
- May modify the data type
- Can be nested
- Accept arguments that can be a column or an expression.

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

Single-Row Functions



Character Functions



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`

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

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

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`

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`

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.

```
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

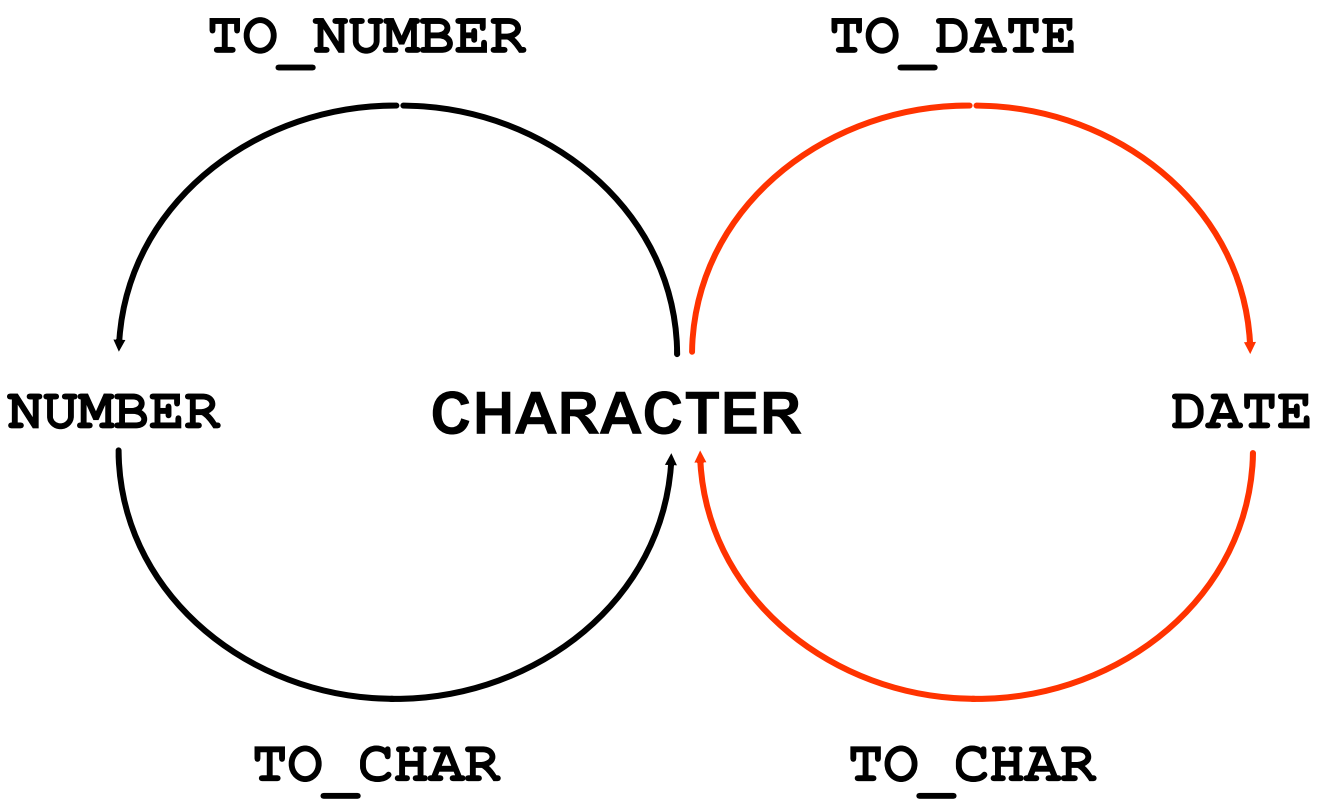
AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Using 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

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'

Data Type Conversion



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
- Is separated from the date value by a comma

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

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.

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

Using the TO_CHAR Function with Numbers

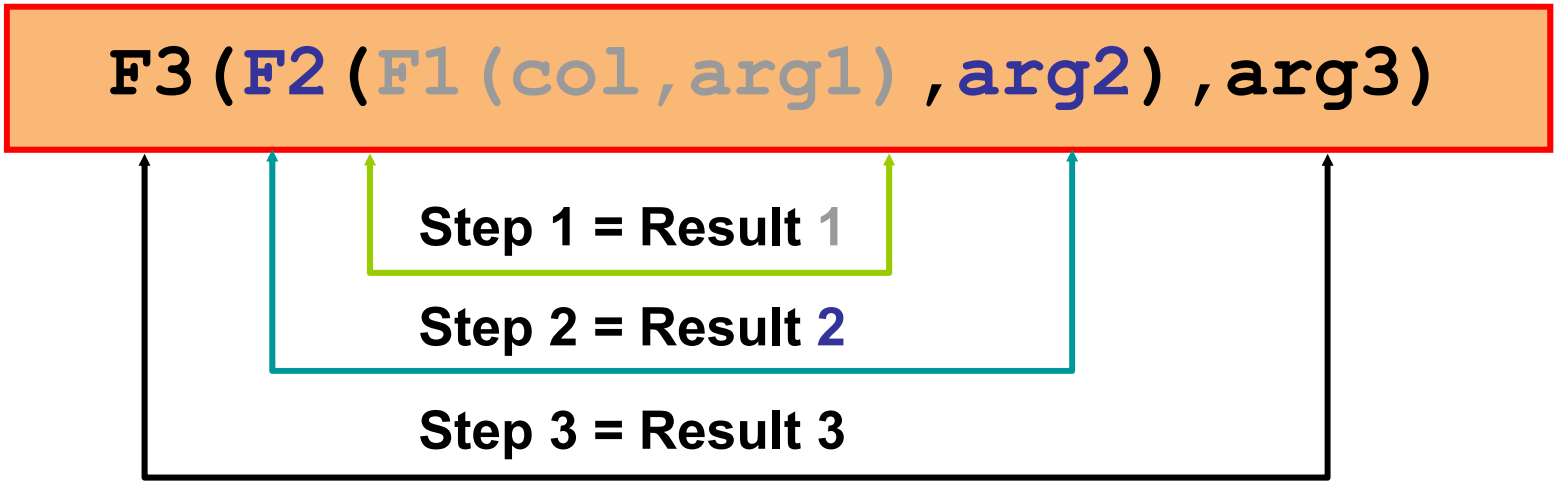
```
SELECT TO_CHAR(salary, '$99,999.00') SALARY
FROM   employees
WHERE  last_name = 'Ernst';
```

SALARY
$6,000.00

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Nesting Functions

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



AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

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
Lorentz	LORENTZ_US

What Are Group Functions?

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

EMPLOYEES

DEPTNO	SAL
10	2450
10	5000
10	1300
20	800
20	1100
20	3000
20	3000
20	2975
30	1600
30	2850
30	1250
30	950
30	1500
30	1250

"maximum salary in the EMP table"

MAX(SALARY)
5000

Types of SQL Group Functions

- COUNT(*)
- COUNT(Cust_Area_Code)
- AVG (Prod_Amt)
- SUM(Prod_Amt)
- MIN (Prod_Amt)
- MAX (Prod_Amt)

Using the COUNT Function

- COUNT(*) returns the number of rows in a table
  - Includes duplicates & nulls

```
SELECT COUNT(*)
FROM EMPLOYEES
WHERE DEPARTMENT_NO = 30;
```

```
 COUNT(*)
---------
         6
```

Using the COUNT Function

- COUNT(*expr*) returns the number of non-null rows.
  - Includes duplicates but not nulls

```
SELECT COUNT(commission_pct)
FROM EMPLOYEES
WHERE deptno = 30;
```

```
COUNT(COMM)
-----------
          4
```

Using the COUNT Function

- COUNT(*expr*) includes duplicates
- Use DISTINCT to eliminate duplicates

```
SELECT COUNT(department_no)
FROM EMPLOYEES;

     COUNT(DEPARTMENT_NO)
     --------------------
                14
```

```
SELECT COUNT(DISTINCT DEPARTMENT_NO)
FROM EMPLOYEES;

     COUNT(DISTINCTDEPTNO)
     ---------------------
                3
```

Using AVG and SUM Functions

- You can use AVG and SUM for numeric data.

```
SELECT  AVG(salary), MAX(salary),
        MIN(salary), SUM(salary)
FROM EMPLOYEES
WHERE job LIKE 'SALES%';
```

AVG(SAL)	MAX(SAL)	MIN(SAL)	SUM(SAL)
1400	1600	1250	5600

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Using MIN and MAX Functions

- You can use MIN and MAX for any datatype.

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

```
MIN(HIRED MAX(HIRED
--------- ---------
17-DEC-80 12-JAN-83
```

Group Functions and Null Values

- All Group functions, except Count(*), ignore null values in the column.

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

```
 AVG(COMM)
----------
       550
```

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Using the NVL Function with Group Functions

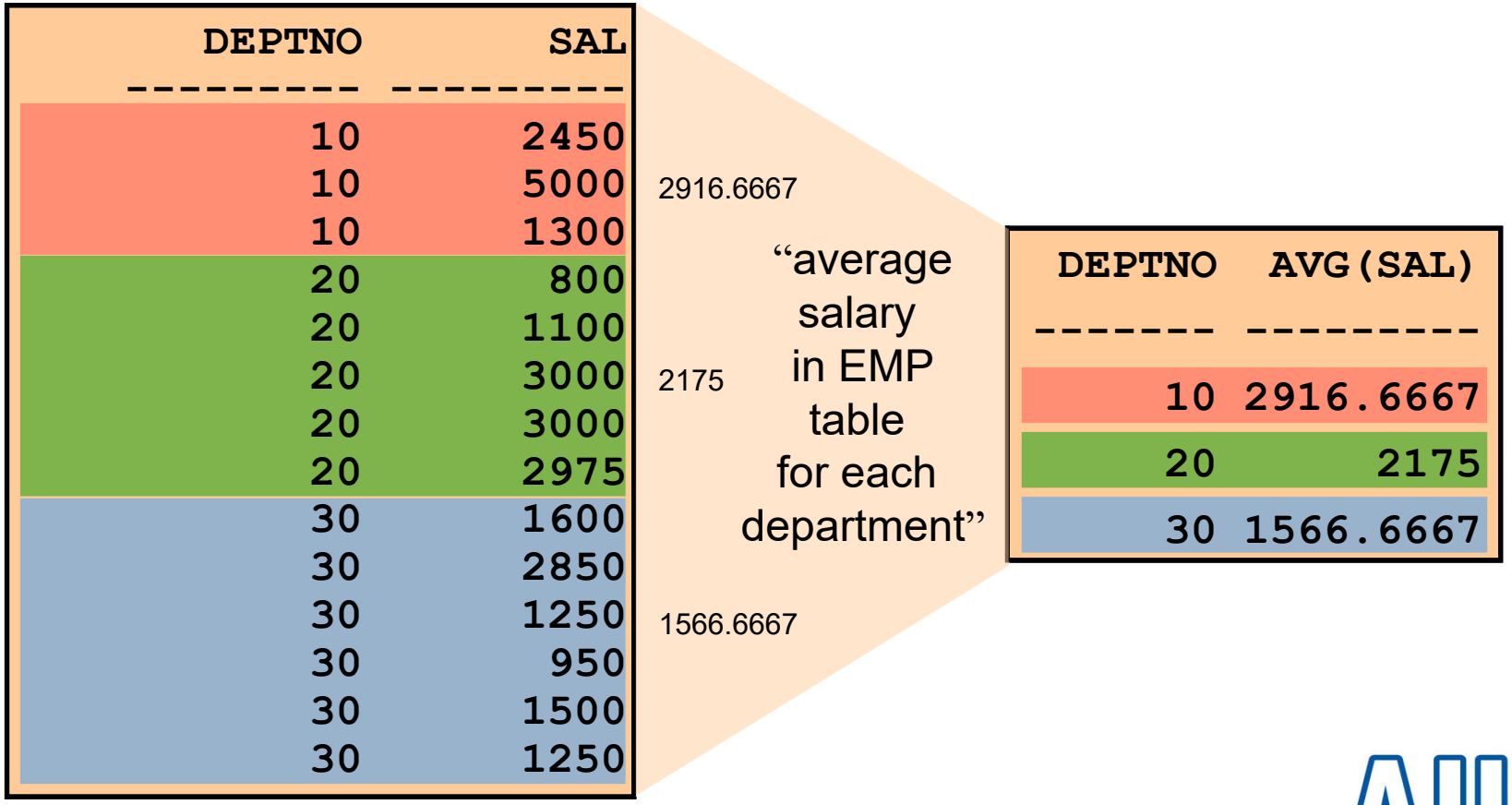
- The NVL function forces group functions to include null values.

```
SELECT AVG(NVL(commission_oct,0))
FROM   EMPLOYEES;
```

```
AVG(NVL(COMM,0))
----------------
      157.14286
```

Creating Groups of Data

EMPLOYEES



AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Creating Groups of Data: GROUP BY Clause

- Divide rows in a table into smaller groups by using the GROUP BY clause.

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

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_no, AVG(salary)
FROM      EMPLOYEES
GROUP BY  department_no;
```

```
   DEPTNO  AVG(SAL)
--------- ---------
        10 2916.6667
        20       2175
        30 1566.6667
```

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_no;
```

```
 AVG(SAL)
---------
2916.6667
     2175
1566.6667
```

Grouping by More Than One Column

EMPLOYEES

DEPTNO	JOB	SAL
10	MANAGER	2450
10	PRESIDENT	5000
10	CLERK	1300
20	CLERK	800
20	CLERK	1100
20	ANALYST	3000
20	ANALYST	3000
20	MANAGER	2975
30	SALESMAN	1600
30	MANAGER	2850
30	SALESMAN	1250
30	CLERK	950
30	SALESMAN	1500
30	SALESMAN	1250

"sum salaries in the EMP table for each job, grouped by department"

DEPTNO	JOB	SUM(SAL)
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
20	ANALYST	6000
20	CLERK	1900
20	MANAGER	2975
30	CLERK	950
30	MANAGER	2850
30	SALESMAN	5600

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Using the GROUP BY Clause on Multiple Columns

```
SELECT    department_no, job_id, sum(salary)
FROM      EMPLOYEES
GROUP BY department_no, job_id;
```

```
   DEPTNO JOB         SUM(SAL)
--------- --------- ----------
        10 CLERK           1300
        10 MANAGER         2450
        10 PRESIDENT       5000
        20 ANALYST         6000
        20 CLERK           1900
...
9 rows selected.
```

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

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_no, COUNT(first_name)
FROM EMPLOYEES;
```

Column missing in the GROUP BY clause

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

Illegal Queries Using Group Functions

- You can not use the WHERE clause to restrict groups.
- You use the HAVING clause to restrict groups.

```
SELECT  department_no, AVG(salary)
FROM  EMPLOYEES
WHERE  AVG(salary) > 2000
GROUP BY department_no;
```

```
WHERE AVG(salary) > 2000
      *
ERROR at line 3:
ORA-00934: group function is not allowed here
```

Cannot use the WHERE clause to restrict groups

Excluding Group Results

EMPLOYEES

DEPTNO	SAL
10	2450
10	5000
10	1300
20	800
20	1100
20	3000
20	3000
20	2975
30	1600
30	2850
30	1250
30	950
30	1500
30	1250

5000

3000

2850

"maximum salary per department greater than $2900"

DEPTNO	MAX(SAL)
10	5000
20	3000

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Excluding Group Results: HAVING Clause

- Use the HAVING clause to restrict groups
  - 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];
```

Using the HAVING Clause

```
SELECT    department_no, max(salary)
FROM      EMPLOYEES
GROUP BY  department_no
HAVING    max(salary)>2900;
```

```
   DEPTNO  MAX(SAL)
--------- ----------
        10       5000
        20       3000
```

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Using the HAVING Clause

```
SELECT job_id, SUM(salary) PAYROLL
FROM   EMPLOYEES
WHERE job_id NOT LIKE 'SALES%'
GROUP BY  job_id
HAVING    SUM(salary)>5000
ORDER BY  SUM(salary);
```

```
JOB_ID          PAYROLL
---------     ---------
ANALYST            6000
MANAGER            8275
```

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Nesting Group Functions

- Display the maximum average salary.

```
SELECT   max(avg(salary))
FROM     EMPLOYEES
GROUP BY department_no;
```

```
MAX(AVG(SALARY))
----------------
     2916.6667
```

Order of precedence

- Order of evaluation of the clauses:
  - WHERE clause
  - GROUP BY clause
  - HAVING clause

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

Lab Activities

- Complete this week's lab exercise

AUT
UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU