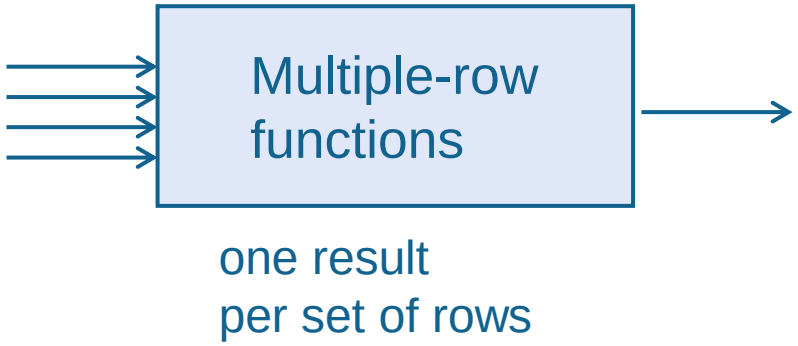
Module 3
Group Functions

Table of Content

Group Functions

- Group functions
  - Operate on sets of rows
  - Give one result per group



- E.g,
  - *The average salary for all the employees*
  - *The number of employees in each department*

Group Functions

```
SELECT      { group_function (column), ... }
FROM        table
[ WHERE     condition(s) ] ;
```

- Group functions
  - AVG
  - COUNT
  - MAX
  - MIN
  - SUM
  - STDEV
  - VAR

Group Functions

- Without any group by clause (see further), all rows are considered as a group

  ⇨ Only one row displayed as result of the select

AVG and SUM Functions

- For numeric data
- E.g,

```
select   avg (salary), sum(salary)
from     employee ;
```

↳ only **one row** as result :

i.e, *the average salary and the sum of salaries of all employees*

MIN and MAX Functions

- Can be used on
  - Numeric
  - Character
  - Date

- E.g,

```
select   max(salary), min(hire_date), max(last_name)
from     employee ;
```

COUNT Function

- **COUNT(*)** returns the number of rows in a table
  - E.g,

```
select    count(*)
from      employee ;
```

- **COUNT(expression)** returns the number of rows with *non-null* values for expression
  - E.g,

```
select    count(commission_pct)
from      employee ;
```

↳*the number of employees who have a not null commission_pct*

COUNT Function

- N.B. COUNT(*mandatory column*) ≡ COUNT(*)
  - ⇨ COUNT(*identifier column*) ≡ COUNT(*)
  - E.g,

    count(employee_id) ≡ count(*)

    count(salary) ≡ count(*)

- COUNT(*optional column*) ≠ COUNT(*)
  - E.g,

    count(commision_pct) ≠ count(*)

COUNT Function with DISTINCT Keyword

- **COUNT(DISTINCT expression)**
  returns the number of ***distinct non-null*** values of *expression*
  - E.g,

```
select    count(distinct department_id)
from      employee ;
```

↳ *the number of distinct departments*

- N.B. **COUNT(DISTINCT** *mandatory column***)**
  ≠ **COUNT**(*mandatory column*)
  - E.g,
    count(salary) ≡ count(*)
    count(distinct salary) ≠ count(salary)

Group Functions and Null Values

- Group functions ignore null values in the column
  - E.g,

```
select    avg(commission_pct)
from      employee ;
```

↳ *average of the not null values for commission_pct*

- Use **ISNULL** function to force group functions to include null values
  - E.g,

```
select    avg ( isnull (commission_pct, 0) )
from      employee ;
```

↳ *average of the commission_pct of all employees*
(a null value is considered here as 0)

GROUP BY Clause

- To divide rows of a table into groups of rows based on a column
- To apply group function on each group
- By using the GROUP BY clause

```
SELECT      { column | group_function (column), …}
FROM          table
[ WHERE       condition(s) ]
GROUP BY    { column, … }
[ ORDER BY  { column | group_function (column) | expr | alias [ ASC|DESC ], … } ] ;
```

GROUP BY Clause

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

```
select      department_id, avg(salary)
from        employee
group by    department_id ;
```

- **Illegal queries !**
  - E.g,

```
select      department_id, job_id, count(*)
from        employee
group by    department_id ;
```

GROUP BY Clause

- If grouping on an optional column, a group is created with null values
  - E.g, *group by (department_id)*

    ⇨ *a group is created with employees with no department*

GROUP BY Clause on Multiple Columns

- To divide rows of a table into groups and sub-groups
  - E.g,

    group employees by department

    + among a same department, by job

```
select      department_id, job_id, sum(salary)
from        employee
group by    department_id,  job_id
```

HAVING Clause

- To restrict groups
  - Groups that do not satisfy group conditions of the having clause are discarded

```
SELECT        { column | group_function (column), …}
FROM           table
[ WHERE       condition(s) ]
GROUP BY     { column, … }
HAVING        group_condition(s)
[ ORDER BY  { column | group_function (column) | alias [ ASC | DESC ], … } ] ;
```

↳Where ***group_conditions*** are expressed using group functions

HAVING Clause

- The sequence in execution
  1. Rows that do not satisfy the conditions of the **WHERE** clause are discarded
  2. Rows are grouped based on the **GROUP BY** criteria
  3. Group functions are applied
  4. Groups matching the **HAVING** clause are displayed

- E.g.

```
select      department_id, max(salary)
from        employee
where       commission_pct is not null
group by    department_id
having      max(salary) >10000 ;
```

Nesting Group Functions

- Group functions can be nested
  - E.g, *the maximum average salary*

```
select      max ( avg ( salary ) )
from        employee
group by    department_id ;
```

↳ only one row as result

Summary

```
SELECT        { column | group_function (column), …}
FROM           table
[ WHERE       condition(s) ]
GROUP BY     { column, … }
HAVING         group_condition(s)
[ ORDER BY  { column | group_function (column) | alias | expr [ ASC|DESC ], … } ] ;
```