Module 1
Basic Select Statements

Table of Content

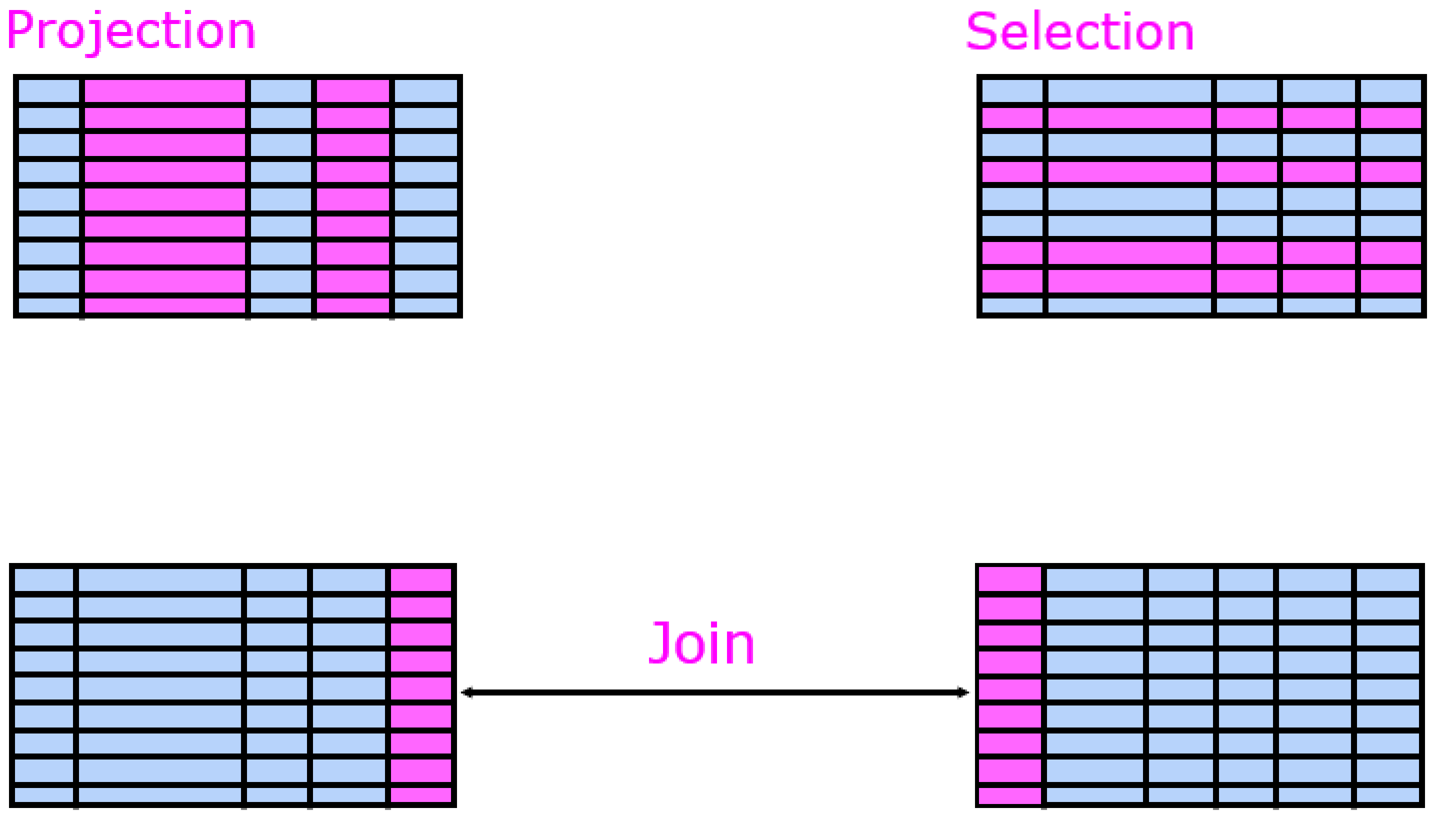
Table of Content

SQL Statements Rules

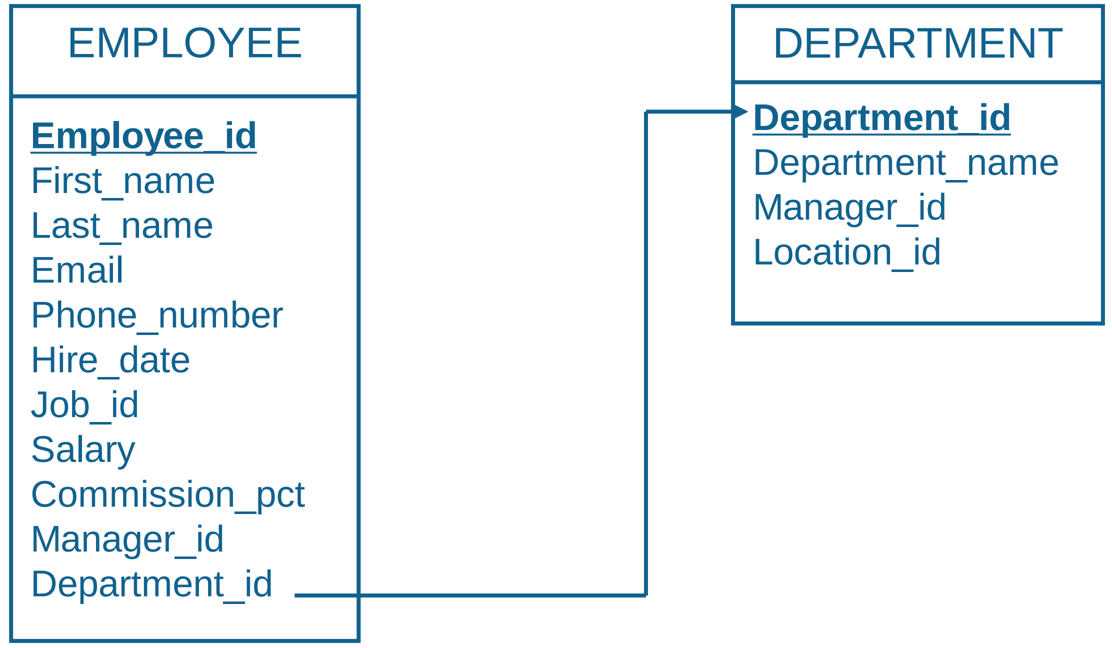
- SQL statements are not case-sensitive
  - By convention, in this course,

    all SQL reserved words will be written in uppercase !
- SQL statements can be entered on one or more lines
- Keywords cannot be abbreviated or split across lines
- Clauses can be placed on separate lines
- Indents can be used to enhance readability
- SQL statements must be terminated by a semicolon (;)
  - In script with multiple SQL statements

Types of Select



Tables Used in Examples



Oracle (or Microsoft) Definition Syntax

- Not case sensitive
- SQL reserved words in uppercase
- **[ ]** ⇨ optional
- **{ X, … }** ⇨ a list of at least one value
  - Multiple values are separated by commas
- **… | …** ⇨ choise between two options (OR)

SELECT Statement Syntax

```
SELECT  * | [ DISTINCT ] { column | expression [ alias ] , ... }
FROM   table ;
```

- The **SELECT** clause identifies the columns to be displayed
- The **FROM** clause identifies the table containing those columns

Selecting All Columns

- By using * in the SELECT clause
- *E.g.*

```
select  *
from    employee ;
```

Selecting Specific Columns

- By specifying the names of the columns in the SELECT clause
- *E.g.*

```
select   department_id,  location_id
from     department ;
```

Arithmetic Expressions

- Arithmetic operators can be used on number and date columns
  - Add : **+**
  - Subtract : **-**
  - Multiply : *****
  - Divide : **/**

- *E.g.*

```
select    last_name,  12 * salary + 1000
from      employee ;
```

Null Value

- If a value is unavailable, unassigned, unknown or inapplicable
  ↳ Use the NULL value
- Null is not the same as zero or a blank space
- *E.g, the commision_pct column is optional ⇨ contains null values*

```
select   last_name,  salary,  commission_pct
from     employee ;
```

Null Value

- Arithmetic expressions containing a null value evaluate to null

- *E.g.*

```
select  last_name,  12 * salary * commission_pct
from    employee ;
```

Column Alias

- Renames a column heading
- Useful with calculations
- Immediately follows the column name
- Optional AS keyword between the column name and alias
- Double quotation marks if it contains spaces or special characters

Column Alias

E.g.

```
select    last_name  "Name", salary * 12   "Annual Salary"
from      employee ;
```

Name	Annual Salary
King	288000
Kochar	204000
De Haan	204000
…	…

Concatenation Operator

- Links columns or character strings to other columns
- By using **+** operator
- Date and character literal values must be enclosed within single quotation marks (**'**)
- *E.g*

```
select    last_name + ' is a ' + job_id   as "Employee Details"
from      employee ;
```

Employee Details
Abel is a SA_REP
Davies is a ST_CLERK
De Haan is a AD_VP
Ernst is a IT_PROG
…

The DISTINCT Keyword

- By default duplicate rows are displayed
  - *E.g.*

```
select  department_id
from    employee ;
```

The same department will be displayed several times if it contains several employees

- To avoid duplicate rows ⇨ use **DISTINCT**
  - *E.g.*

```
select  distinct  department_id
from    employee ;
```

A same department will be displayed only once even if it contains several employees

Restricting Selected Rows

- By using the **WHERE** clause

```
SELECT  * | [ DISTINCT ] { column | expression [ alias ] , ... }
FROM   table
[ WHERE  condition(s) ] ;
```

- *E.g.*

```
select    *
from      employee
where     department_id = 20 ;
```

Only rows where department_id is equal to 20 are displayed

Restricting Selected Rows

- Character strings and date values are enclosed with single quotation marks (')
- Pay attention to the format of dates ! (see module 2)
- *E.g.*

```
select    *
from      employee
where     last_name = 'Smith' ;
```

```
select    last_name
from      employee
where     hire_date  > '20-DEC-2010' ;
```

Comparison Operators

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
BETWEEN ... AND	Between two values (inclusive)
IN (…)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

BETWEEN … AND … Operator

- To express conditions based on a range of values
- On columns of number, string or date type

- *E.g.*

```
select    *
from      employee
where     salary  between   2000  and  10000 ;
```

Lower limit   Upper limit

- N.B. The limits are included into the range of values

IN (…) Operator

- To express membership condition
- To test if values are included into a list
- On columns of number, string or date type
- *E.g.*

```
select   *
from     employee
where    department_id  in (10, 20, 30) ;
```

LIKE Operator

- Wildcard searches of valid values for literal characters or numbers
  - Zero or many characters : `%`
    - NB: Or ***** in some Database Systems
  - One character : _
    - NB: Or **?** in some Database System
- On columns of number, string or date type
- *E.g.*

```
select    *
from      employee
where     last_name  like  'D%n_' ;
```

IS NULL Operator

- To search for unknown value (i.e. NULL value)

- *E.g.*

```
select    last_name
from      employee
where     department_id  is null ;
```

AND and OR Operators

- *E.g.*

```
select   *
from     employee
where    salary  between  5000  and  8000
and      job_id  like  '%M%' ;
```

```
select   last_name, salary
from     employee
where    department_id  in (10,40)
or       last_name  like 'S%' ;
```

NOT Operator

- **NOT IN (…)**
  - *E.g.*

```
select    *
from      employee
where     last_name  not in ('Janis', 'Smith') ;
```

- **NOT BETWEEN … AND …**
- **NOT LIKE …**
- **IS NOT NULL**
- **NOT (… AND …)**
- **NOT (… OR …)**

Rules of Precedence

Operator	Meaning
1	Arithmetic operators
2	Concatenation operator
3	Comparison conditions
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Not equal to
7	NOT logical condition
8	AND logical condition
9	OR logical condition

Use parenthesis to override rules of precedence

ORDER BY Clause

- To sort retrieved rows
  - Ascending order : ASC (by default)
  - Descending order : **DESC**
- Must be the last clause in the SELECT statement
- *E.g.*

```
select      last_name, hire_date
from        employee
order by    hire_date ;
```

ORDER BY Clause

```
select      *
from        employee
order by  last_name  desc ;
```

```
select      last_name, department_id, salary
from        employee
order by  2  ;
```

```
select      *
from        employee
order by  department_id  desc, salary ;
```

Summary

```
SELECT  * | [ DISTINCT ] { column | expression [ alias ], ... }
FROM    table
[ WHERE  condition(s) ]
[ ORDER BY { column | expression | alias [ ASC | DESC ] , ... } ] ;
```