**CHAPTER 3**

**SQL STATEMENT**

# Select statement

* Objectives: to extract data from the database
* Syntax:

**SELECT \*|{[DISTINCT] *column*|*expression* [*alias*],...}**

**FROM *table;***

In the syntax:

SELECT is a list of one or more columns

\* selects all columns

DISTINCT suppresses duplicates //loại các bản sao chỉ lấy bản ghi

*column|expression* selects the named column or the expression

*alias* gives selected columns different headings

FROM *table*

Example:

**select** \* **from** employees

|  |
| --- |
| By *lines* I assume you mean rows in the table person. What you're looking for is:  select p.name  from person p  where p.name LIKE '%A%'; --contains the character 'A'  The above is case sensitive. For a case insensitive search, you can do:  select p.name  from person p  where UPPER(p.name) LIKE '%A%'; --contains the character 'A' or 'a'  For the special character, you can do:  select p.name  from person p  where p.name LIKE '%'||chr(8211)||'%'; --contains the character chr(8211) |

## Distinct:

* Sau khi lấy dữ liệu ra thì distinct sẽ loại bỏ các bản ghi trùng nhau (toàn bộ cả hàng)

# Where

* Objectives:
  + Restrict the rows that are returned by using the WHERE clause
  + The WHERE clause follows the FROM clause
* Syntax:

**SELECT \*|{[DISTINCT] *column|expression* [*alias*],...}**

**FROM *table***

**[WHERE *condition(s)*];**

In the syntax:

WHERE restricts the query to rows that meet a condition

*condition* is composed of column names, expressions, constants, and a comparison operator

Example:

**SELECT** employee\_id, last\_name, job\_id, department\_id

**FROM** employees

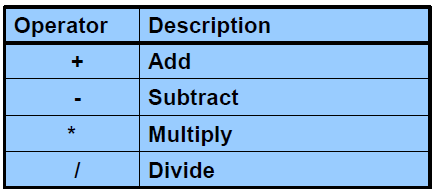
**WHERE** department\_id = 90 ;

**SELECT** \*

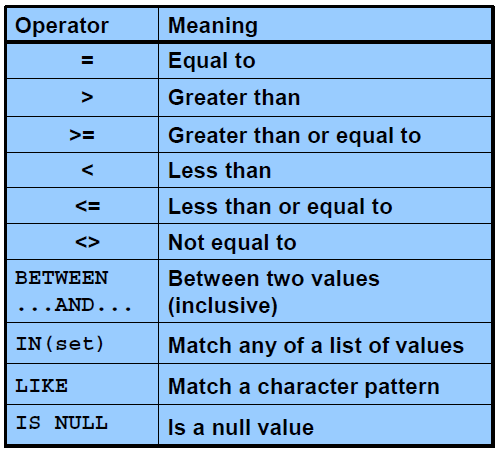
**FROM** employees

**WHERE** hire\_date>='17-jun-1989';

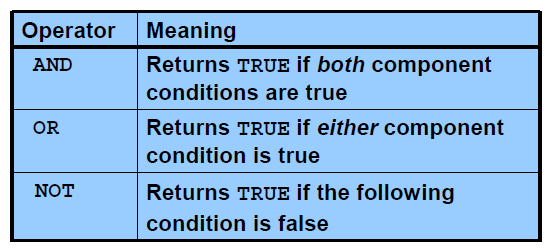
* Arithmetic operators



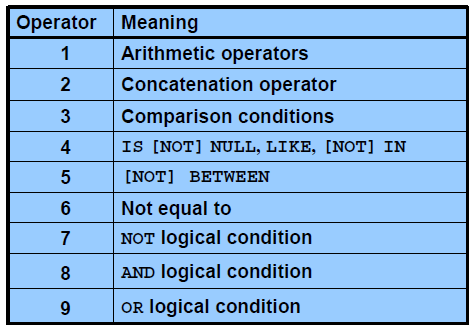
* Comparison Conditions



* Logical Conditions



* Rules of Precedence



Example:

**SELECT** last\_name, job\_id, salary

**FROM** employees

**WHERE** job\_id = 'SA\_REP'

**OR** job\_id = 'AD\_PRES'

**AND** salary > 15000;

* The first condition is that the job ID is AD\_PRES *and* the salary is greater than $15,000.
* The second condition is that the job ID is SA\_REP.
* Sorting

Sort retrieved rows with the ORDER BY clause:

* + ASC: ascending order, default
  + DESC: descending order

The ORDER BY clause comes last in the SELECT statement

**Syntax**

SELECT *expr*

FROM *table*

[WHERE *condition(s)*]

[ORDER BY {*column*, *expr, numeric\_position*} [ASC|DESC]];

In the syntax:

ORDER BY specifies the order in which the retrieved rows are displayed

ASC orders the rows in ascending order (this is the default order)

DESC orders the rows in descending order

**The default sort order is ascending:**

* Numeric values are displayed with the lowest values first (for example, 1 to 999).
* Date values are displayed with the earliest value first (for example, 01-JAN-92 before
* 01-JAN-95).
* Character values are displayed in alphabetical order (for example, A first and Z last).
* Null values are displayed last for ascending sequences and first for descending sequences.
* You can sort by a column that is not in the SELECT list.

# Insert statement

* Add new rows to a table by using the INSERT statement
* With this syntax, only one row is inserted at a time.
* Syntax:

**INSERT INTO *table* [(*column* [*, column...*])]**

**VALUES *(value* [*, value...*]);**

In the syntax:

*table* is the name of the table

*column* is the name of the column in the table to populate

*value* is the corresponding value for the column

* Example:

**INSERT** **INTO** job\_grades **VALUES** ('A', 1000, 2999);

**Insert** **into** job\_grades

**Select** \* **from** job\_grades\_bk

# Update statement

* Modify existing rows with the UPDATE statement
* Update more than one row at a time (if required).
* Syntax:

**UPDATE *table***

**SET *column* = *value* [, *column* = *value, ...*]**

**[WHERE *condition*];**

In the syntax:

*table* is the name of the table

*column* is the name of the column in the table to populate

*value* is the corresponding value or subquery for the column

*condition* identifies the rows to be updated and is composed of column names, expressions, constants, subqueries, and comparison operators

* Example:

**UPDATE** employees

SET department\_id = 70

**WHERE** employee\_id = 113;

**UPDATE** copy\_emp

SET department\_id = 110;

**UPDATE** employees –- update with subquery

SET

job\_id = (**SELECT** job\_id **FROM** employees **WHERE** employee\_id = 205),

salary = (**SELECT** salary **FROM** employees **WHERE** employee\_id = 205)

**WHERE** employee\_id = 114;

# Delete statement

You can remove existing rows from a table by using the DELETE statement

* Syntax:

**DELETE [FROM] *table***

**[WHERE *condition*];**

In the syntax:

*table* is the table name

*condition* identifies the rows to be deleted and is composed of column names,

expressions, constants, subqueries, and comparison operators

* Example:

**DELETE** **FROM** departments

**WHERE** department\_name = 'Finance';

**DELETE** **FROM** copy\_emp;

# Joining table

* Equijoins: Equijoins are also called *simple joins* or *inner joins*

Syntax:

**SELECT *table1.column, table2.column***

**FROM *table1, table2***

**WHERE *table1.column1* = *table2.column2;***

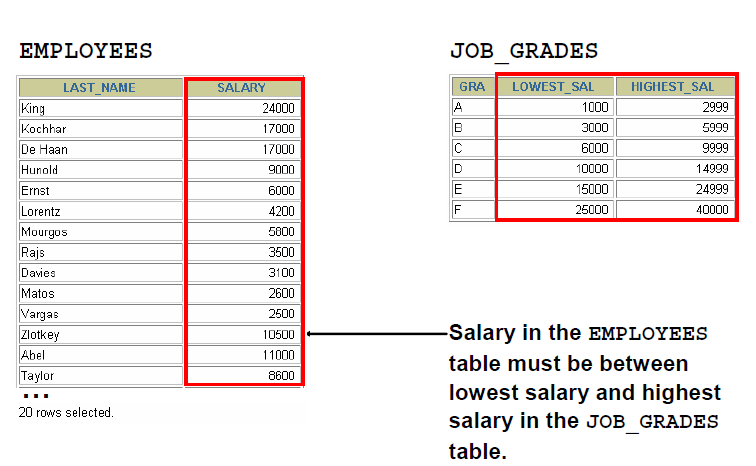
In the syntax:

*table1.column* denotes the table and column from which data is retrieved

*table1.column1* = is the condition that joins (or relates) the tables together

*table2.column2*

* Non-Equijoins



Example:

**SELECT** e.last\_name, e.salary, j.grade\_level

**FROM** employees e, job\_grades j

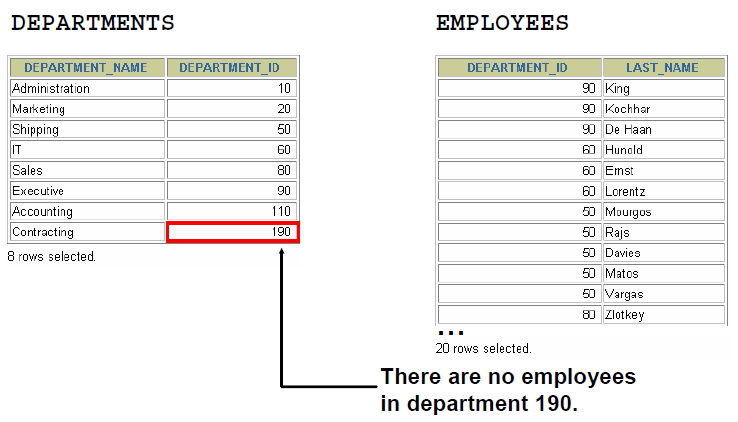
**WHERE** e.salary

**BETWEEN** j.lowest\_sal **AND** j.highest\_sal;

* Outer Joins

You use an outer join to see rows that do not meet the join condition.

The outer join operator is the plus sign (+).



Syntax:

**SELECT *table1.column, table2.column***

**FROM *table1, table2***

**WHERE *table1.column(+)* = *table2.column;***

**SELECT *table1.column, table2.column***

**FROM *table1, table2***

**WHERE *table1.column* = *table2.column(+);***

In the syntax:

*table1.column =* is the condition that joins (or relates) the tables together

*table2.column* (+) is the outer join symbol, which can be placed on either side of the WHERE clause condition, but not on both sides. (Place the outer join symbol following the name of the column in the table without the matching rows.)

Example:

**SELECT** e.last\_name, e.department\_id, d.department\_name

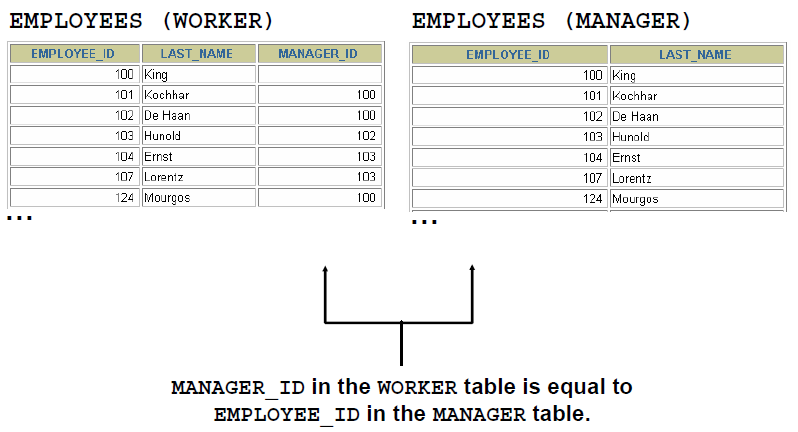
**FROM** employees e, departments d

**WHERE** e.department\_id(+) = d.department\_id ;

* Self-Joins: Sometimes you need to join a table to itself.

Example:

To find the name of each employee’s manager, you need to join the EMPLOYEES table to itself; this type of join is called a *self-join.*



**SELECT** worker.last\_name || ' works for ' || manager.last\_name

**FROM** employees worker, employees manager

**WHERE** worker.manager\_id = manager.employee\_id ;

# Practices

* 1. There are four coding errors in the following statement. Can you identify them?

SELECT employee\_id, last\_name

sal x 12 ANNUAL SALARY

FROM employees;

* 1. The following select statement executes successfully: True/False

**SELECT** last\_name, job\_id, salary **AS** Sal

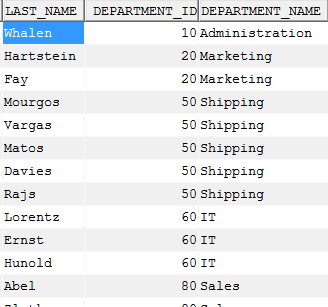
**FROM** employees;

* 1. Result of following statement

**select** **null** \* 2 **from** dual

**select** **null** || 'name' **from** dual

* 1. The HR department needs a report of all employees. Write a query to display the last name, department number, and department name for all employees.

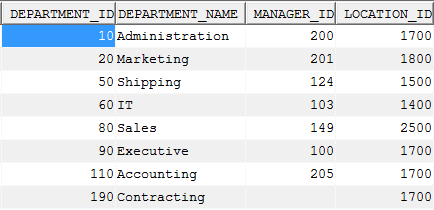


## Select để hiển thị thành 1 câu

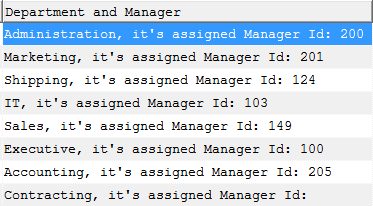
The HR department needs a report of all department with the following format: **Department\_Name + “, it's assigned Manager Id:” + Manager\_ID**

Example: “Administration, it's assigned Manager Id: 200”



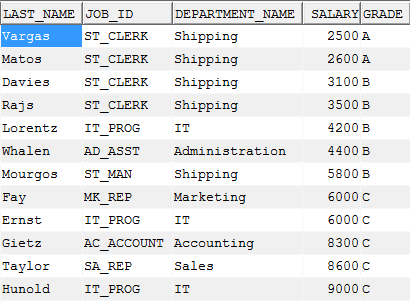


=> Result:

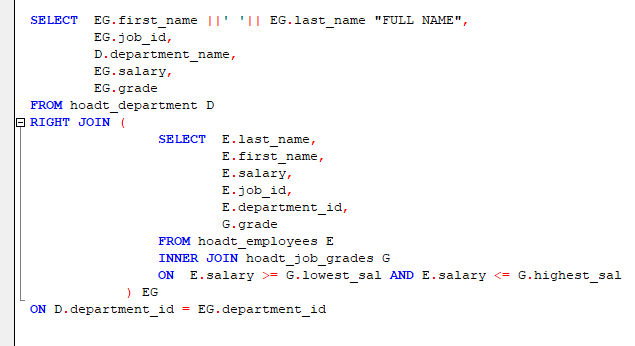


## Câu Select lồng

The HR department needs a report on job grades and salaries. To familiarize yourself with the JOB\_GRADES table, first show the structure of the JOB\_GRADES table. Then create a query that displays the name, job, department name, salary, and grade for all employees

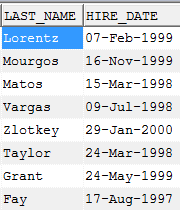


**BÀI LÀM:**



## Định dạng ngày/tháng/năm khi select

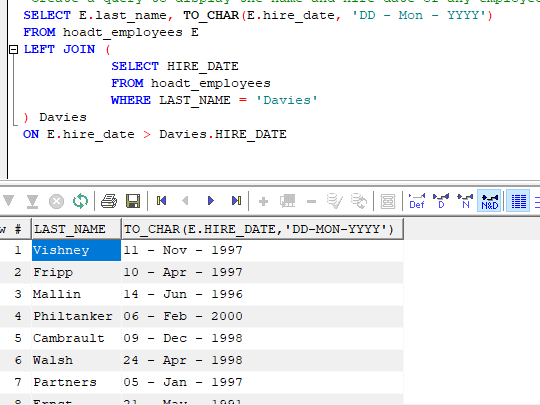
The HR department wants to determine the names of all employees who were hired after Davies. Create a query to display the name and hire date of any employee hired after employee Davies.



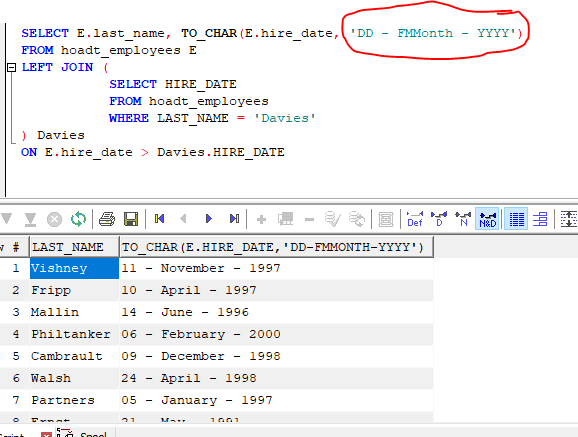
**BÀI LÀM:**

'DD - Mon - YYYY': 10 – Jan – 1997

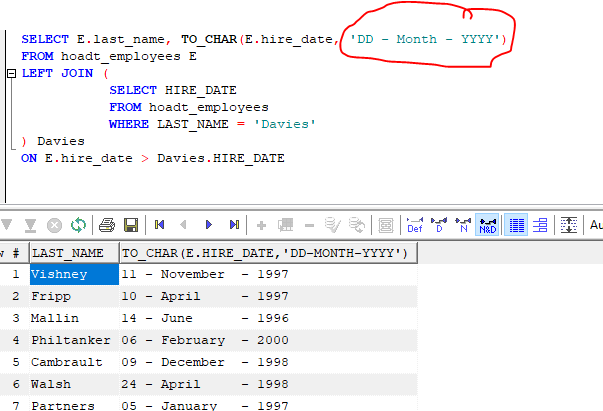
'DD - Month - YYYY': 10 - January – 1997



==========================================================

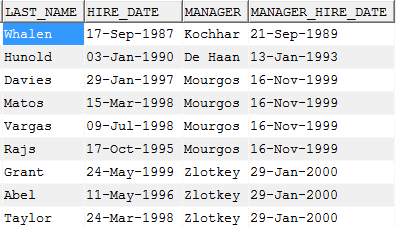


======================================================

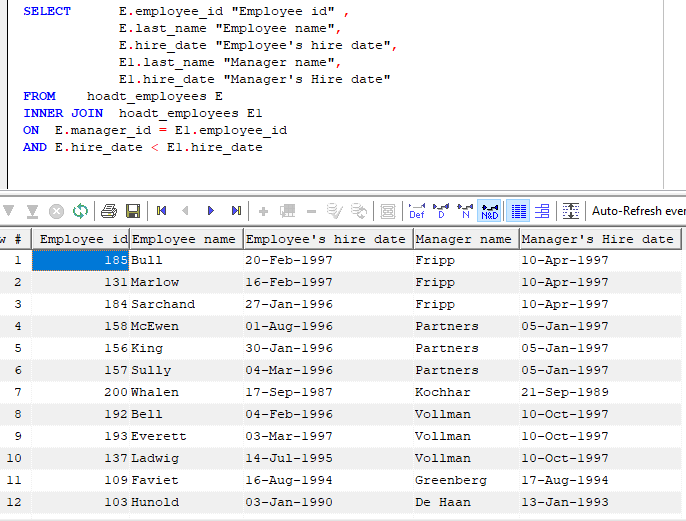


## So sánh ngày trong select

The HR department needs to find the names and hire dates for all employees who were hired before their managers, along with their managers’ names and hire dates

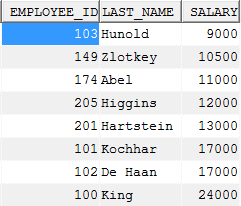


**BÀI LÀM:**

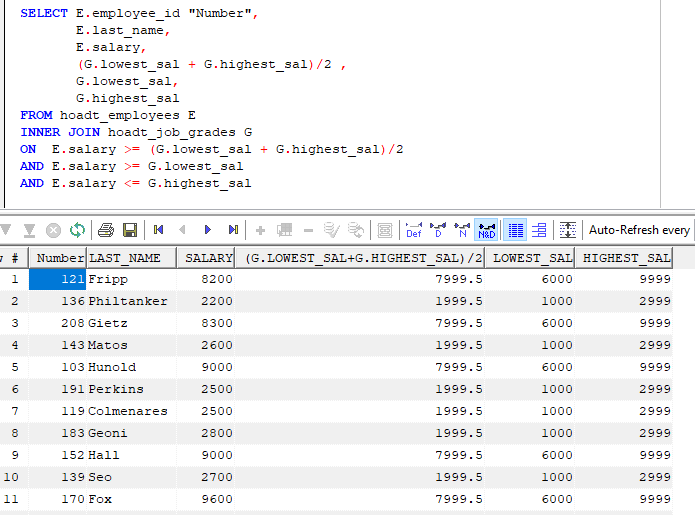


## Câu lệnh sort (asc, desc) – order by

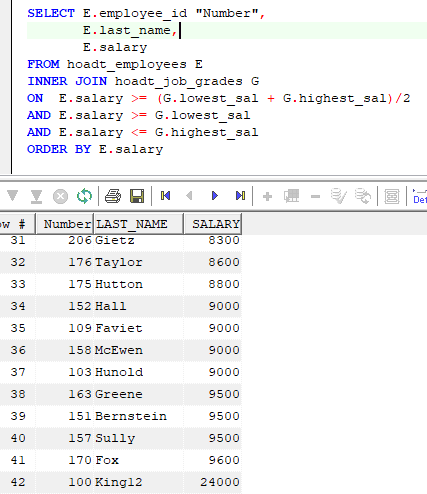
Create a report that displays the employee number, last name, and salary of all employees who earn more than the average salary. Sort the results in order of ascending salary.



**BÀI LÀM:**

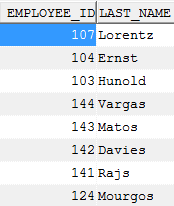


Viết gọn lại thành:

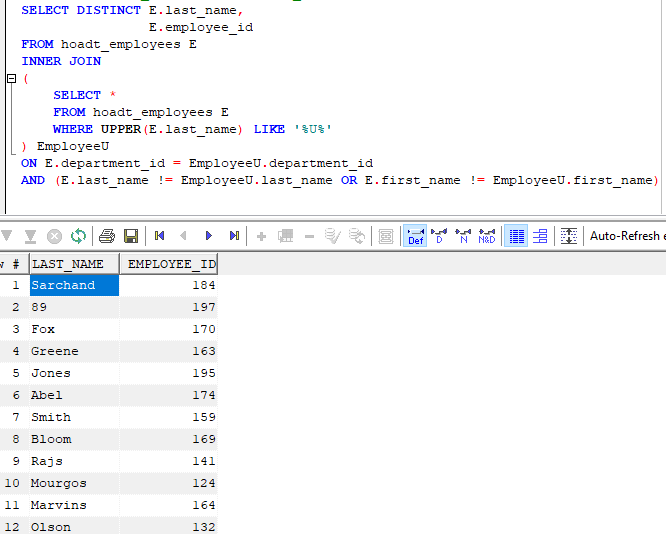


## Format string: contain

Write a query that displays the employee number and last name of all employees who work in a department with any employee whose last name contains a “u”.

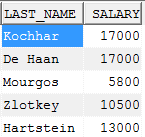


**BÀI LÀM:**

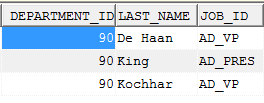


## Like as

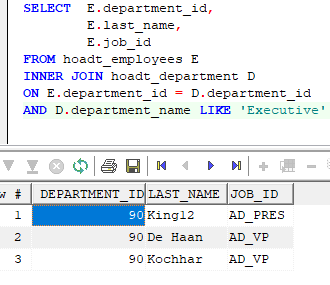
* 1. Create a report for HR that displays the last name and salary of every employee who reports to King.



* 1. Create a report for HR that displays the department number, last name, and job ID for every employee in the Executive department.



**BÀI LÀM:**

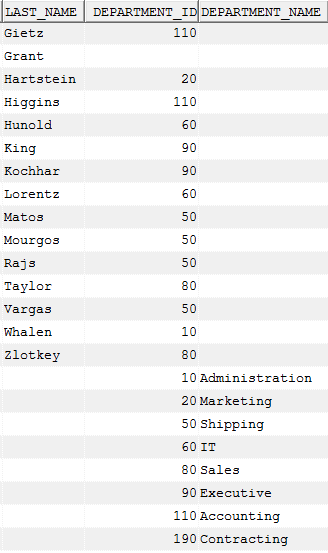


## Full join

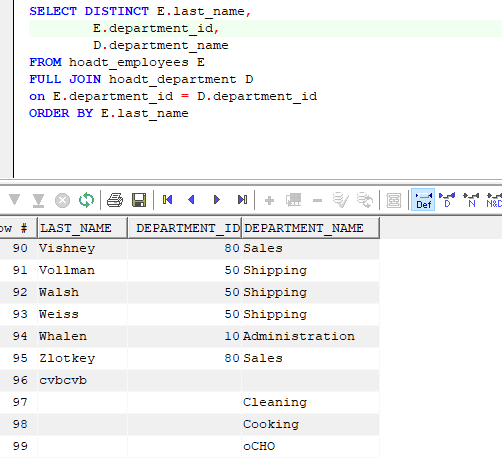
The HR department needs a report with the following specifications:

* + Last name and department ID of all the employees from the EMPLOYEES table, regardless of whether or not they belong to a department
  + Department ID and department name of all the departments from the DEPARTMENTS table, regardless of whether or not they have employees working in them

Write a compound query to accomplish this.



**BÀI LÀM:**



* 1. Create table My\_Employee from Employee table.
  + Create Insert statement to insert the following row:



* + Change the last name of employee 207 to FPTS
  + Confirm your changes to the table.
  + Delete TEST from My\_Employee table
  + Commit all pending changes