SQL 4: DML 3

SCSD2523 Database

Semester 1 2024/2025

Learning Objective

- At the end of the topic, students will be able to construct SQL statements for data manipulations with multiple tables
  - Subqueries
  - Simple Join
  - Equijoin:
    - NATURAL JOIN;
    - JOIN … ON;
    - INNER JOIN ON
  - Outer join:
    - LEFT OUTER JOIN;
    - RIGHT OUTER JOIN;
    - FULL OUTER JOIN

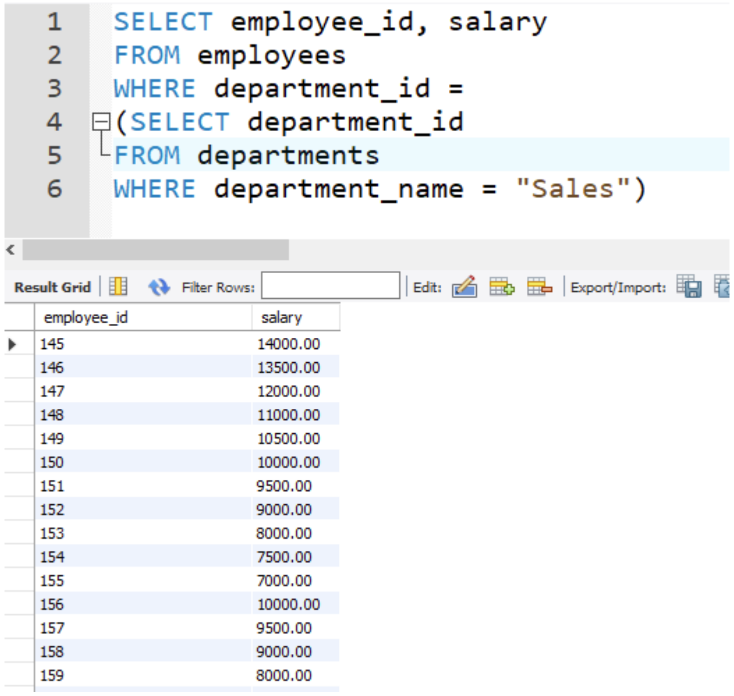
Subqueries

- **Nested query**
  - Contains outer & inner **SELECT** statements.
  - The result of inner **SELECT** (subquery) is used by the outer query to determine the final output.
  - Subqueries written at:
    - **WHERE** or **HAVING** clause
    - **SELECT** or **FROM** clause
  - When subquery written at **SELECT** clause, the subquery output is a single value.

Subqueries

- Example 1:
  - List employee id and salary of all employees who are working in the sales department.

```
SELECT employee_id, salary
FROM employees
WHERE department_id =
( SELECT department_id
  FROM departments
  WHERE department_name = "Sales" )
```



```
SELECT employee_id, salary
FROM employees
WHERE department_id =
(SELECT department_id
FROM departments
WHERE department_name = "Sales")
```

employee_id	salary
145	14000.00
146	13500.00
147	12000.00
148	11000.00
149	10500.00
150	10000.00
151	9500.00
152	9000.00
153	8000.00
154	7500.00
155	7000.00
156	10000.00
157	9500.00
158	9000.00
159	8000.00

Subqueries

- What if we want to also display the department name in the result? How should the statement be written?

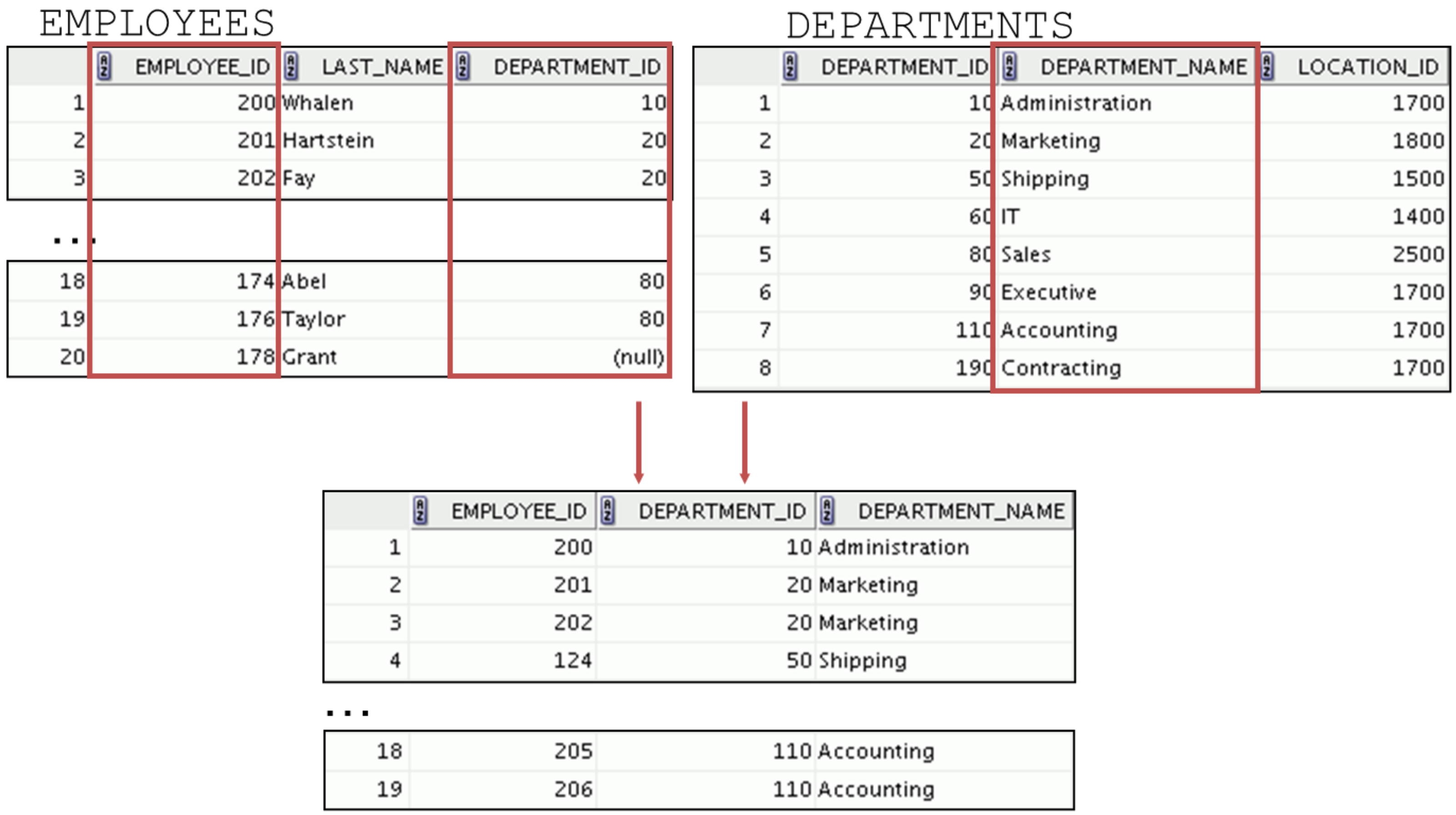
```
SELECT employee_id, salary, department_name
FROM employees
WHERE department_id =
( SELECT department_id
  FROM departments
  WHERE department_name = “Sales” )
```

- Will this query execute correctly???

Obtaining Data From Multiple Tables

- Sometimes you need to use and display data from more than one table.
  - Employee IDs exist in the EMPLOYEES table.
  - Department IDs exist in both the EMPLOYEES and DEPARTMENTS tables.
  - Department names exist in the DEPARTMENTS table.
- To produce the report, you need to **join** the EMPLOYEES and DEPARTMENTS tables, and access data from **both of them**.

Obtaining Data From Multiple Tables



EMPLOYEES

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	200	Whalen	10
2	201	Hartstein	20
3	202	Fay	20

. . .

18	174	Abel	80
19	176	Taylor	80
20	178	Grant	(null)

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700

	EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	200	10	Administration
2	201	20	Marketing
3	202	20	Marketing
4	124	50	Shipping

. . .

18	205	110	Accounting
19	206	110	Accounting

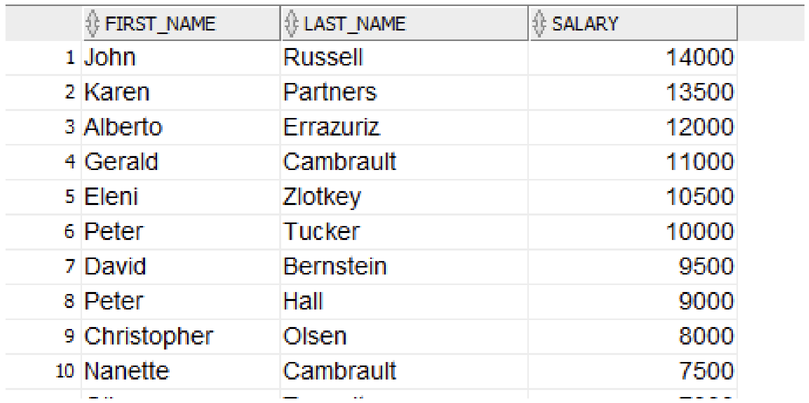
Simple Join / Traditional Method

Query 1:

- List the employee first name, last name and employee salary for the SALES department.

```
SELECT first_name, last_name, salary
FROM employees, departments
WHERE employees.department_id = departments.department_id
AND departments.department_name = ‘Sales’;
```

This is a simple join *without the use of any **special syntax** in SQL



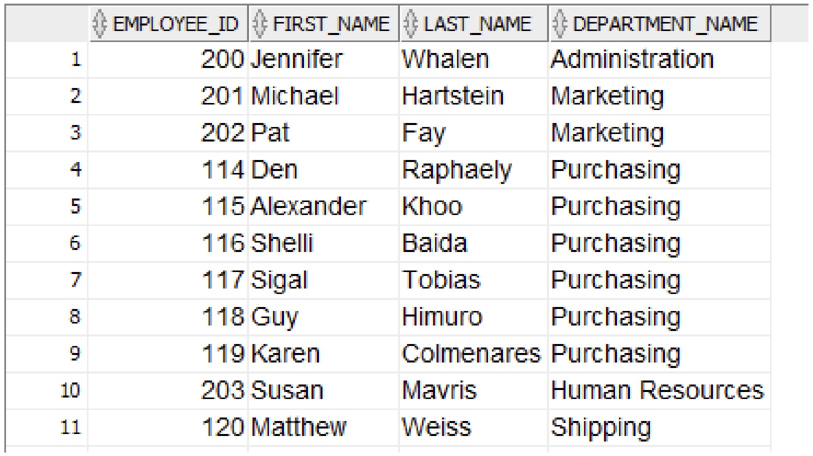
	FIRST_NAME	LAST_NAME	SALARY
1	John	Russell	14000
2	Karen	Partners	13500
3	Alberto	Errazuriz	12000
4	Gerald	Cambrault	11000
5	Eleni	Zlotkey	10500
6	Peter	Tucker	10000
7	David	Bernstein	9500
8	Peter	Hall	9000
9	Christopher	Olsen	8000
10	Nanette	Cambrault	7500

Simple Join / Traditional Method

Query 2:

- List all employees id, first name, last name and department name.

```
SELECT employee_id, first_name, last_name, department_name
FROM employees, departments
WHERE employees.department_id = departments.department_id;
```



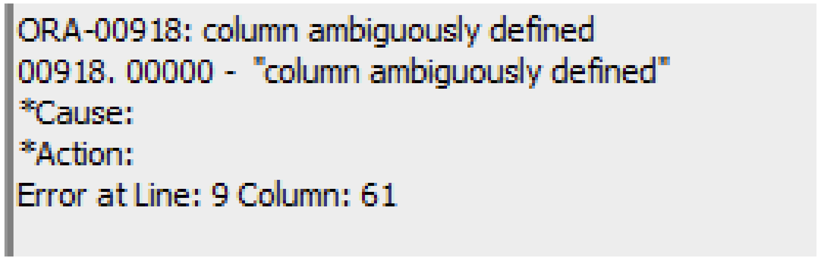
	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_NAME
1	200	Jennifer	Whalen	Administration
2	201	Michael	Hartstein	Marketing
3	202	Pat	Fay	Marketing
4	114	Den	Raphaely	Purchasing
5	115	Alexander	Khoo	Purchasing
6	116	Shelli	Baida	Purchasing
7	117	Sigal	Tobias	Purchasing
8	118	Guy	Himuro	Purchasing
9	119	Karen	Colmenares	Purchasing
10	203	Susan	Mavris	Human Resources
11	120	Matthew	Weiss	Shipping

Simple Join / Traditional Method

Query 3:

- List all employee id, first name, last name, department id and department name.

```
SELECT employee_id, first_name, last_name, department_name, department_id
FROM employees, departments
WHERE employees.department_id = departments.department_id;
```



```
ORA-00918: column ambiguously defined
00918. 00000 -  "column ambiguously defined"
*Cause:
*Action:
Error at Line: 9 Column: 61
```

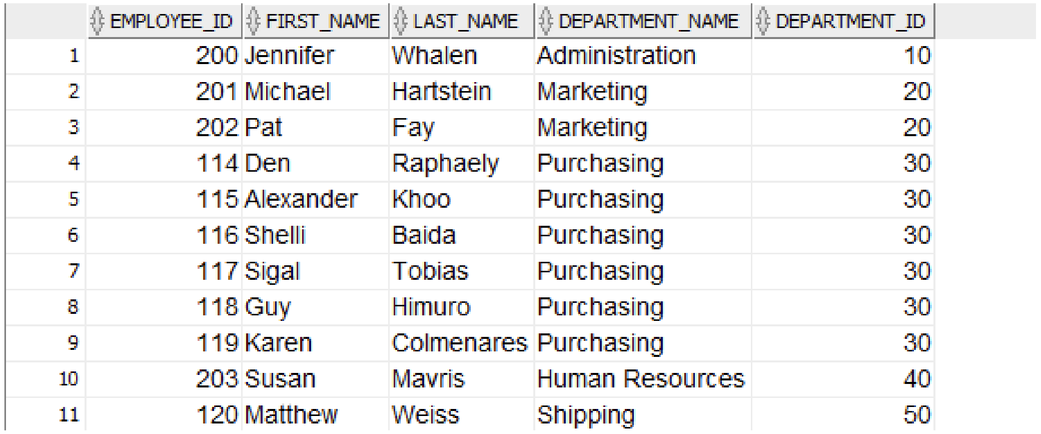
Why error occur?
How to correct the error?

Simple Join / Traditional Method

Query 3:

- List all employee id, first name, last name, department id and department name.

```
SELECT employee_id, first_name, last_name, department_name,
employee.department_id
FROM employees, departments
WHERE employees.department_id = departments.department_id;
```



	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_NAME	DEPARTMENT_ID
1	200	Jennifer	Whalen	Administration	10
2	201	Michael	Hartstein	Marketing	20
3	202	Pat	Fay	Marketing	20
4	114	Den	Raphaely	Purchasing	30
5	115	Alexander	Khoo	Purchasing	30
6	116	Shelli	Baida	Purchasing	30
7	117	Sigal	Tobias	Purchasing	30
8	118	Guy	Himuro	Purchasing	30
9	119	Karen	Colmenares	Purchasing	30
10	203	Susan	Mavris	Human Resources	40
11	120	Matthew	Weiss	Shipping	50

To solve the ambiguous column issue, use **Table prefix** to correct the error

Joining Tables Using SQL:1999 syntax

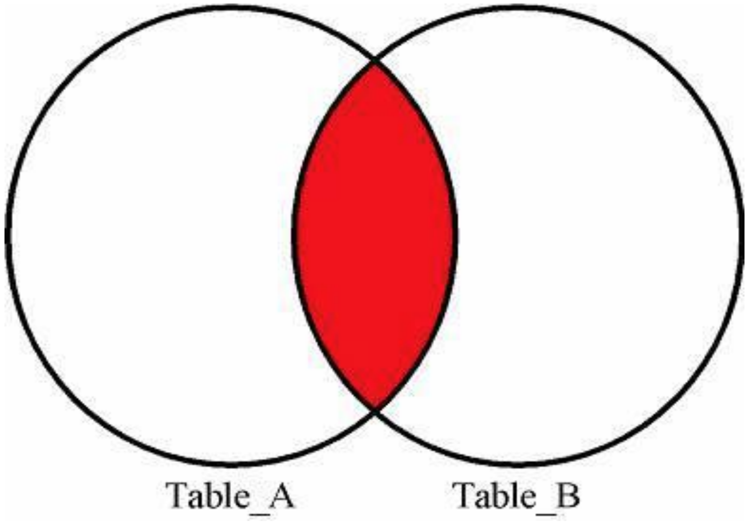
- Use a join to query data from more than one table:

```
SELECT table1.column, table2.column
FROM table1
[NATURAL JOIN table2]
[JOIN table2 USING (column_name)]
[JOIN table2 ON (table1.column_name = table2.column_name)]
[LEFT OUTER JOIN table2 ON (table1.column_name = table2.column_name)]
[RIGHT OUTER JOIN table2 ON (table1.column_name = table2.column_name)]
[FULL OUTER JOIN table2 ON (table1.column_name = table2.column_name)]
[CROSS JOIN table2];
```

[ ] = optional

Equijoins

- That is where a column (or multiple columns) in two or more tables match
- Can write using:
  - Simple join:
    - **SELECT…FROM…WHERE**
  - ANSI syntax:
    - **NATURAL JOIN**
    - **JOIN…USING**
    - **JOIN…ON**
    - **INNER JOIN…ON**



Using NATURAL JOIN

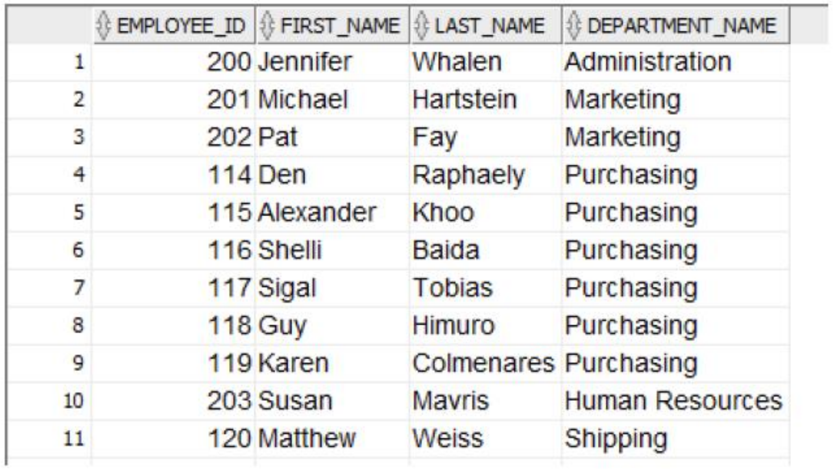
- The **NATURAL JOIN** clause is based on all columns in the two tables that have the **same name**.
- It selects rows from the two tables that have **equal values** in all matched columns.
- If the columns having the same names have different data types, an error is returned.

Using NATURAL JOIN

Query 2:

- List all employees id, first name, last name and department name.

```
SELECT employee_id, first_name, last_name, department_name
FROM employees
NATURAL JOIN departments;
```



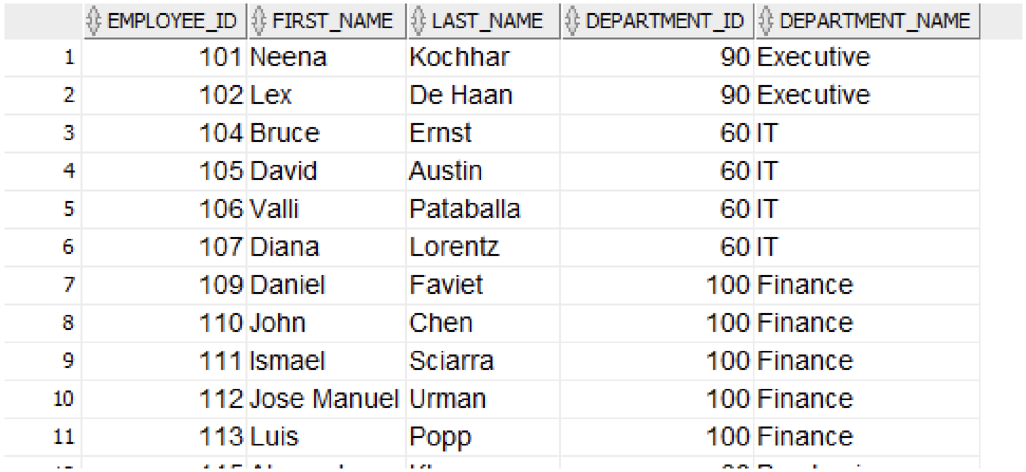
	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_NAME
1	200	Jennifer	Whalen	Administration
2	201	Michael	Hartstein	Marketing
3	202	Pat	Fay	Marketing
4	114	Den	Raphaely	Purchasing
5	115	Alexander	Khoo	Purchasing
6	116	Shelli	Baida	Purchasing
7	117	Sigal	Tobias	Purchasing
8	118	Guy	Himuro	Purchasing
9	119	Karen	Colmenares	Purchasing
10	203	Susan	Mavris	Human Resources
11	120	Matthew	Weiss	Shipping

Using NATURAL JOIN

Query 3:

- List all employee id, first name, last name, department id and department name.

```
SELECT employee_id, first_name, last_name, department_name, department_id
FROM employees
NATURAL JOIN departments;
```



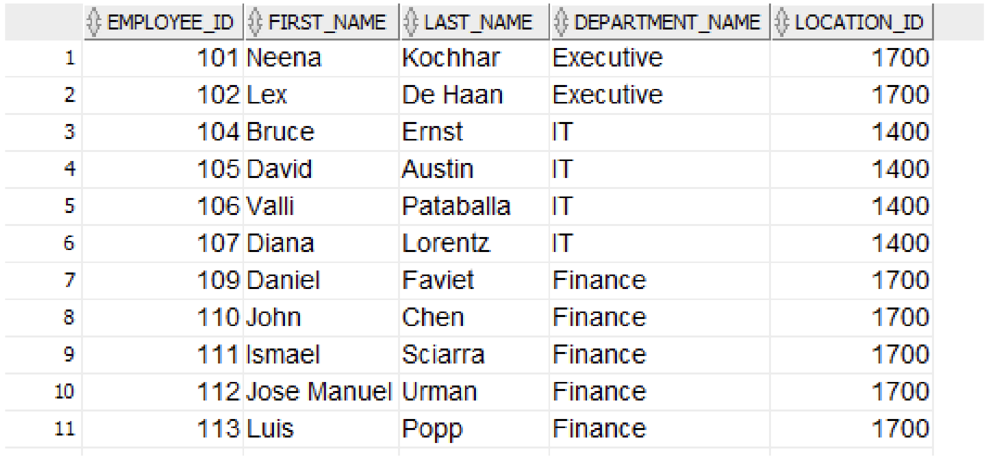
	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	101	Neena	Kochhar	90	Executive
2	102	Lex	De Haan	90	Executive
3	104	Bruce	Ernst	60	IT
4	105	David	Austin	60	IT
5	106	Valli	Pataballa	60	IT
6	107	Diana	Lorentz	60	IT
7	109	Daniel	Faviet	100	Finance
8	110	John	Chen	100	Finance
9	111	Ismael	Sciarra	100	Finance
10	112	Jose Manuel	Urman	100	Finance
11	113	Luis	Popp	100	Finance

Using NATURAL JOIN

Query 4:

- List all employee id, first name, last name, department name and department location id.

```
SELECT employee_id, first_name, last_name, department_name, location_id
FROM employees
NATURAL JOIN departments;
```



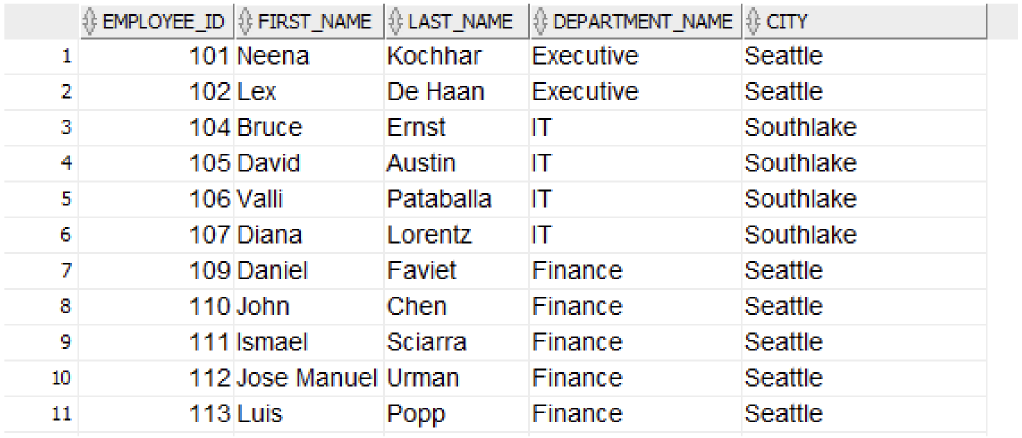
	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_NAME	LOCATION_ID
1	101	Neena	Kochhar	Executive	1700
2	102	Lex	De Haan	Executive	1700
3	104	Bruce	Ernst	IT	1400
4	105	David	Austin	IT	1400
5	106	Valli	Pataballa	IT	1400
6	107	Diana	Lorentz	IT	1400
7	109	Daniel	Faviet	Finance	1700
8	110	John	Chen	Finance	1700
9	111	Ismael	Sciarra	Finance	1700
10	112	Jose Manuel	Urman	Finance	1700
11	113	Luis	Popp	Finance	1700

Using NATURAL JOIN

Query 4b:

- List all employee id, first name, last name, department name and city.

```
SELECT employee_id, first_name, last_name, department_name, city
FROM employees
NATURAL JOIN departments
NATURAL JOIN locations;
```



	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_NAME	CITY
1	101	Neena	Kochhar	Executive	Seattle
2	102	Lex	De Haan	Executive	Seattle
3	104	Bruce	Ernst	IT	Southlake
4	105	David	Austin	IT	Southlake
5	106	Valli	Pataballa	IT	Southlake
6	107	Diana	Lorentz	IT	Southlake
7	109	Daniel	Faviet	Finance	Seattle
8	110	John	Chen	Finance	Seattle
9	111	Ismael	Sciarra	Finance	Seattle
10	112	Jose Manuel	Urman	Finance	Seattle
11	113	Luis	Popp	Finance	Seattle

Natural join with multiple tables

Using JOIN…USING clause

- If several columns have the same names but the data types do not match, use the **USING** clause to specify the columns for the equijoin.
- Use the **USING** clause to match only one column when more than one column matches.

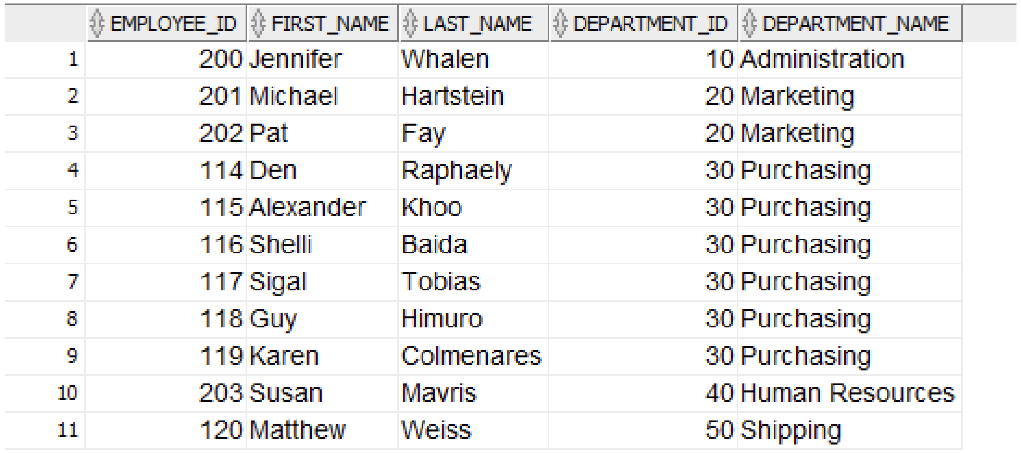
Using JOIN…USING clause

Query 3:

- List all employee id, first name, last name, department id and department name.

```
SELECT employee_id, first_name, last_name, department_id, department_name
FROM employees
JOIN departments
USING (department_id);
```

Join based on the **department_id** attribute



	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	200	Jennifer	Whalen	10	Administration
2	201	Michael	Hartstein	20	Marketing
3	202	Pat	Fay	20	Marketing
4	114	Den	Raphaely	30	Purchasing
5	115	Alexander	Khoo	30	Purchasing
6	116	Shelli	Baida	30	Purchasing
7	117	Sigal	Tobias	30	Purchasing
8	118	Guy	Himuro	30	Purchasing
9	119	Karen	Colmenares	30	Purchasing
10	203	Susan	Mavris	40	Human Resources
11	120	Matthew	Weiss	50	Shipping

Using JOIN…ON clause

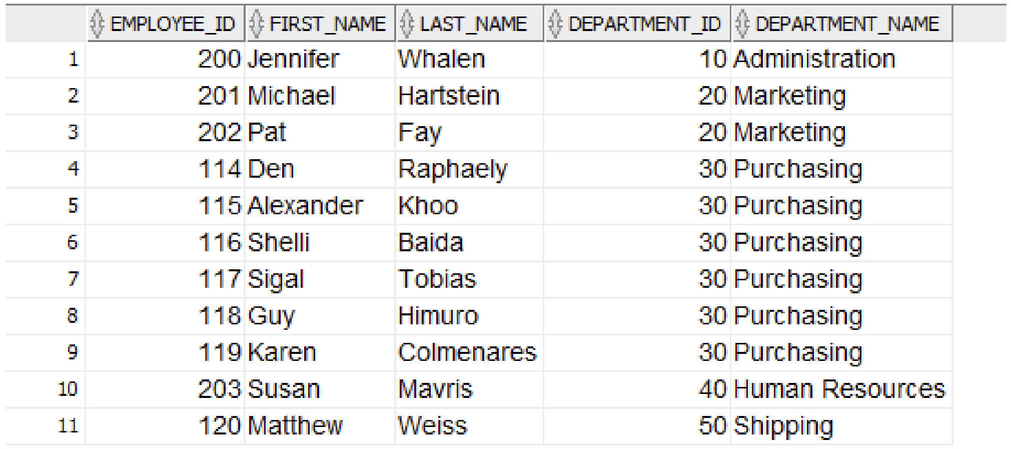
- The **JOIN** condition for the **NATURAL JOIN** is basically an equijoin of all columns with the same name.
- Use the **ON** clause to specify arbitrary conditions or specify columns to join.
- The join condition is separated from other search conditions.
- The **ON** clause makes code easy to understand.

Using JOIN…ON clause

Query 3:

- List all employee id, first name, last name, department id and department name.

```
SELECT employee_id, first_name, last_name, department_id, department_name
FROM employees
JOIN departments
ON (employees.department_id = departments.department_id);
```



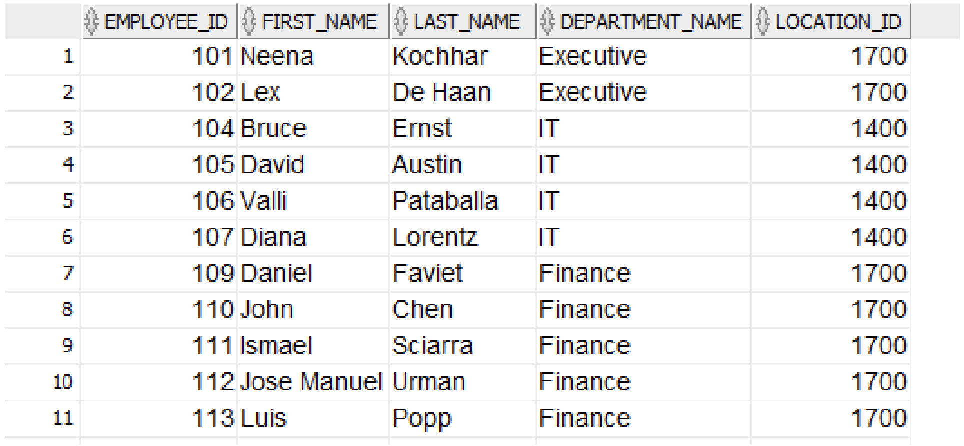
	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	200	Jennifer	Whalen	10	Administration
2	201	Michael	Hartstein	20	Marketing
3	202	Pat	Fay	20	Marketing
4	114	Den	Raphaely	30	Purchasing
5	115	Alexander	Khoo	30	Purchasing
6	116	Shelli	Baida	30	Purchasing
7	117	Sigal	Tobias	30	Purchasing
8	118	Guy	Himuro	30	Purchasing
9	119	Karen	Colmenares	30	Purchasing
10	203	Susan	Mavris	40	Human Resources
11	120	Matthew	Weiss	50	Shipping

Using JOIN…ON clause

Query 4:

- List all employee id, first name, last name, department name and department location id.

```
SELECT employee_id, first_name, last_name, department_name, location_id
FROM employees
JOIN departments
ON (employees.department_id = departments.department_id);
```



	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_NAME	LOCATION_ID
1	101	Neena	Kochhar	Executive	1700
2	102	Lex	De Haan	Executive	1700
3	104	Bruce	Ernst	IT	1400
4	105	David	Austin	IT	1400
5	106	Valli	Pataballa	IT	1400
6	107	Diana	Lorentz	IT	1400
7	109	Daniel	Faviet	Finance	1700
8	110	John	Chen	Finance	1700
9	111	Ismael	Sciarra	Finance	1700
10	112	Jose Manuel	Urman	Finance	1700
11	113	Luis	Popp	Finance	1700

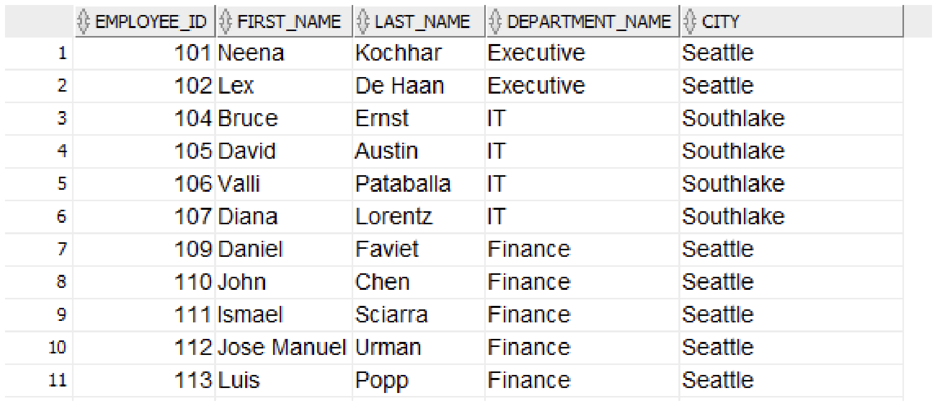
Using JOIN…ON clause

Query 4b:

- List all employee id, first name, last name, department name and city.

```
SELECT employee_id, first_name, last_name, department_name, city
FROM employees e
JOIN departments d
ON e.department_id = d.department_id
JOIN locations l
ON d.location_id = l.location_id;
```

Using **JOIN…ON** with multiple tables



	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_NAME	CITY
1	101	Neena	Kochhar	Executive	Seattle
2	102	Lex	De Haan	Executive	Seattle
3	104	Bruce	Ernst	IT	Southlake
4	105	David	Austin	IT	Southlake
5	106	Valli	Pataballa	IT	Southlake
6	107	Diana	Lorentz	IT	Southlake
7	109	Daniel	Faviet	Finance	Seattle
8	110	John	Chen	Finance	Seattle
9	111	Ismael	Sciarra	Finance	Seattle
10	112	Jose Manuel	Urman	Finance	Seattle
11	113	Luis	Popp	Finance	Seattle

Applying Additional Conditions to a Join

- Use the **AND** clause or the **WHERE** clause to apply additional conditions:

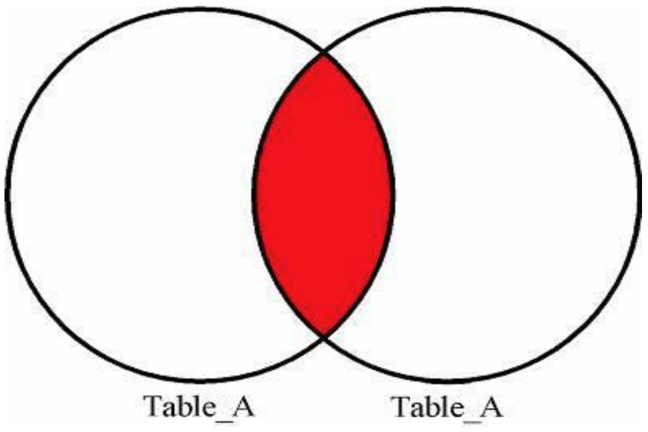
```
SELECT    e.employee_id, e.last_name, e.department_id,
          d.department_id, d.location_id
FROM      employees e JOIN departments d
ON        (e.department_id = d.department_id)
AND       e.manager_id = 149 ;
```

Or

```
SELECT    e.employee_id, e.last_name, e.department_id,
          d.department_id, d.location_id
FROM      employees e JOIN departments d
ON        (e.department_id = d.department_id)
WHERE     e.manager_id = 149 ;
```

Self Join

- A self join is a special form of equijoin or **INNER JOIN** where a table is **joined against itself**.
- This means that the table **must exists two times** in the **FROM** clause of the SQL query.
- Note that when joining a table to itself **an alias must be used** for each of the tables in the **FROM** clause and then also used in the select list and **WHERE** clause

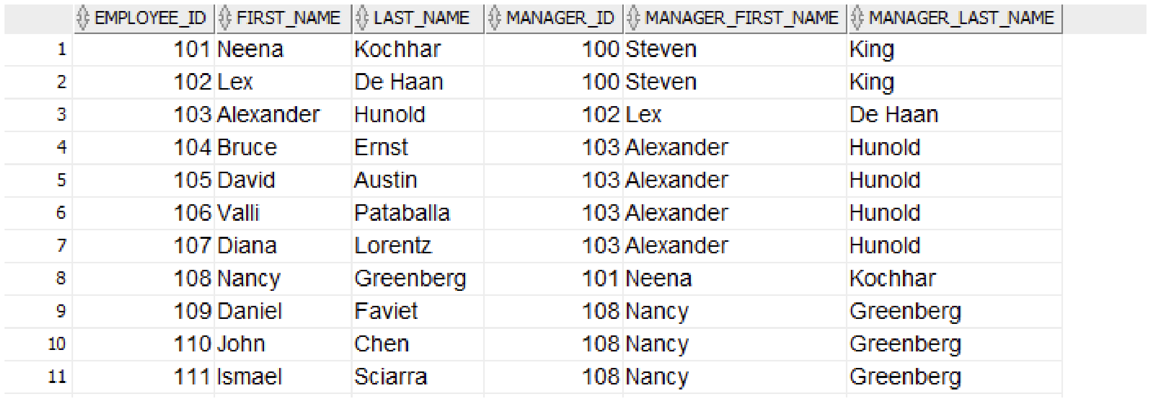


Self join using INNER JOIN…ON

Query 5:

- List all employee id, first name, last name and their manager's id and first name, last name.

```
SELECT e.employee_id, e.first_name, e.last_name, m.employee_id
       MANAGER_ID, m.first_name MANAGER_FNAME, m.last_name MANAGER_LNAME
FROM employees e
INNER JOIN employees m
ON e.manager_id = m.employee_id;
```



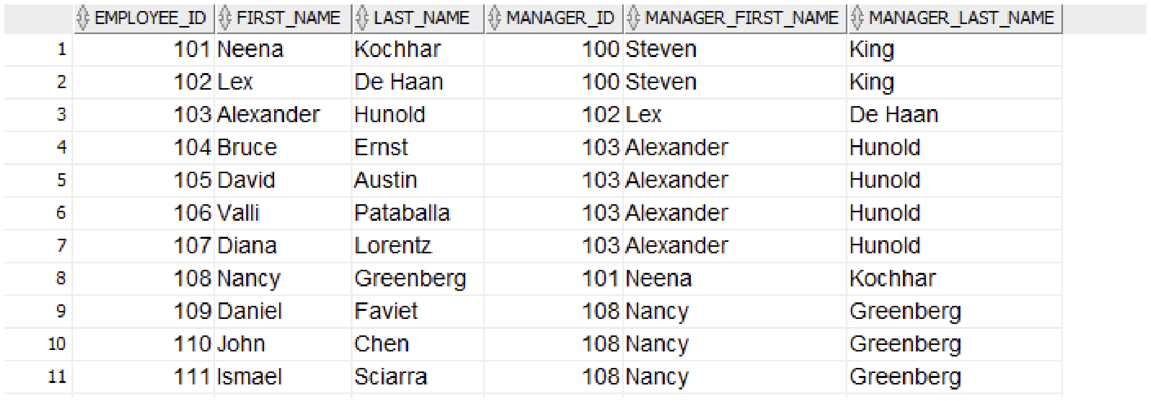
	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	MANAGER_ID	MANAGER_FIRST_NAME	MANAGER_LAST_NAME
1	101	Neena	Kochhar	100	Steven	King
2	102	Lex	De Haan	100	Steven	King
3	103	Alexander	Hunold	102	Lex	De Haan
4	104	Bruce	Ernst	103	Alexander	Hunold
5	105	David	Austin	103	Alexander	Hunold
6	106	Valli	Pataballa	103	Alexander	Hunold
7	107	Diana	Lorentz	103	Alexander	Hunold
8	108	Nancy	Greenberg	101	Neena	Kochhar
9	109	Daniel	Faviet	108	Nancy	Greenberg
10	110	John	Chen	108	Nancy	Greenberg
11	111	Ismael	Sciarra	108	Nancy	Greenberg

Self join using INNER JOIN…ON

Query 5:

- List all employee id, first name, last name and their manager's id and first name, last name.

```
SELECT e.employee_id, e.first_name, e.last_name, m.employee_id
       MANAGER_ID, m.first_name MANAGER_FNAME, m.last_name MANAGER_LNAME
FROM employees e
JOIN employees m
ON e.manager_id = m.employee_id;
```



	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	MANAGER_ID	MANAGER_FIRST_NAME	MANAGER_LAST_NAME
1	101	Neena	Kochhar	100	Steven	King
2	102	Lex	De Haan	100	Steven	King
3	103	Alexander	Hunold	102	Lex	De Haan
4	104	Bruce	Ernst	103	Alexander	Hunold
5	105	David	Austin	103	Alexander	Hunold
6	106	Valli	Pataballa	103	Alexander	Hunold
7	107	Diana	Lorentz	103	Alexander	Hunold
8	108	Nancy	Greenberg	101	Neena	Kochhar
9	109	Daniel	Faviet	108	Nancy	Greenberg
10	110	John	Chen	108	Nancy	Greenberg
11	111	Ismael	Sciarra	108	Nancy	Greenberg

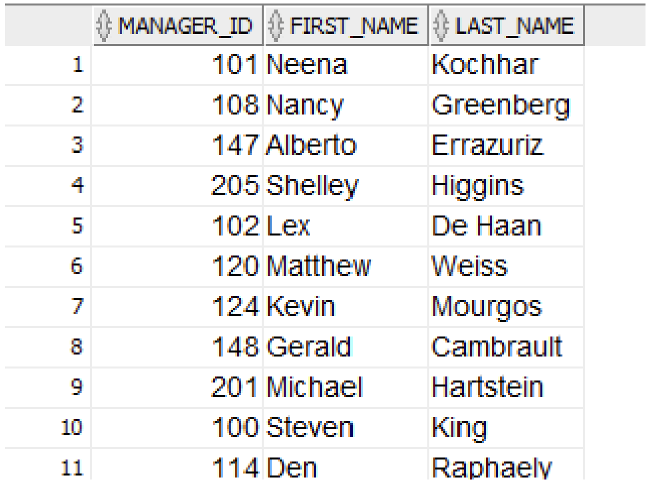
JOIN and **INNER JOIN** are functionally equivalent

Self join using INNER JOIN…ON

Query 6:

- List all employee who are the manager.

```
SELECT DISTINCT e.manager_id, m.first_name, m.last_name
FROM employees e
INNER JOIN employees m
ON e.manager_id = m.employee_id;
```



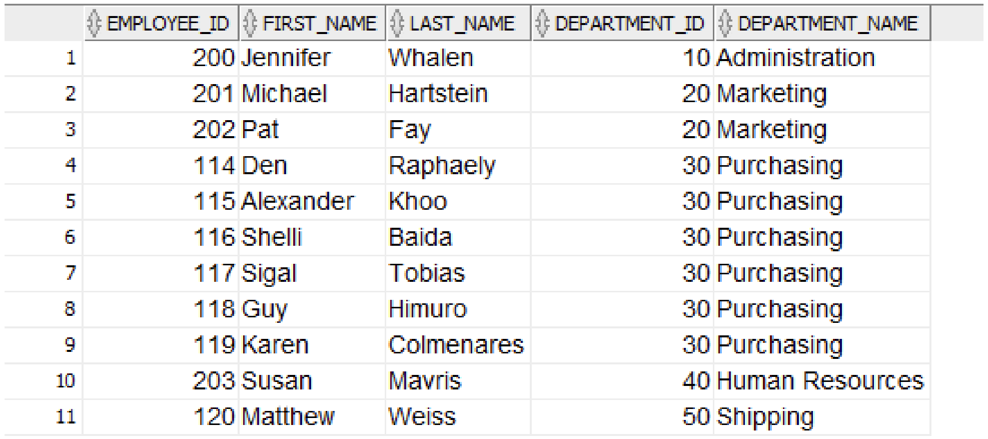
	MANAGER_ID	FIRST_NAME	LAST_NAME
1	101	Neena	Kochhar
2	108	Nancy	Greenberg
3	147	Alberto	Errazuriz
4	205	Shelley	Higgins
5	102	Lex	De Haan
6	120	Matthew	Weiss
7	124	Kevin	Mourgos
8	148	Gerald	Cambrault
9	201	Michael	Hartstein
10	100	Steven	King
11	114	Den	Raphaely

Using INNER JOIN…ON

Query 3:

- List all employee id, first name, last name, department id and department name.

```
SELECT employee_id, first_name, last_name, department_id, department_name
FROM employees
INNER JOIN departments
ON (employees.department_id = departments.department_id);
```



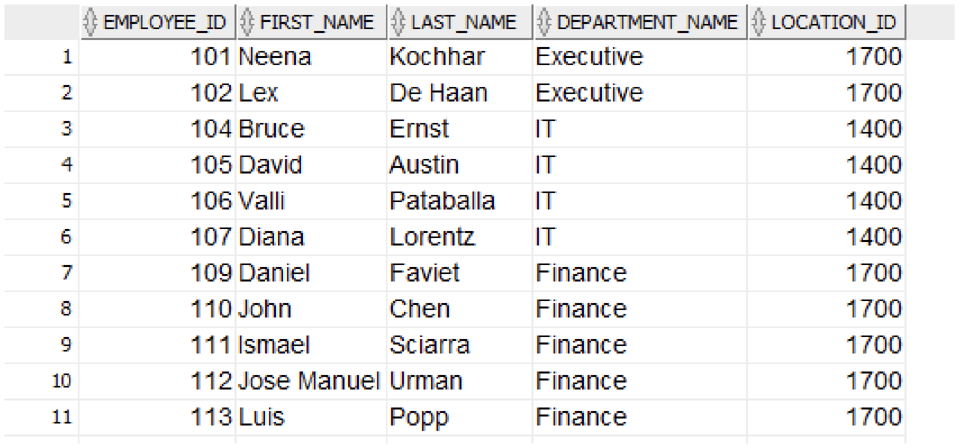
	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	200	Jennifer	Whalen	10	Administration
2	201	Michael	Hartstein	20	Marketing
3	202	Pat	Fay	20	Marketing
4	114	Den	Raphaely	30	Purchasing
5	115	Alexander	Khoo	30	Purchasing
6	116	Shelli	Baida	30	Purchasing
7	117	Sigal	Tobias	30	Purchasing
8	118	Guy	Himuro	30	Purchasing
9	119	Karen	Colmenares	30	Purchasing
10	203	Susan	Mavris	40	Human Resources
11	120	Matthew	Weiss	50	Shipping

Using INNER JOIN...ON

Query 4:

- List all employee id, first name, last name, department name and department location id.

```
SELECT employee_id, first_name, last_name, department_name, location_id
FROM employees
INNER JOIN departments
ON (employees.department_id = departments.department_id);
```



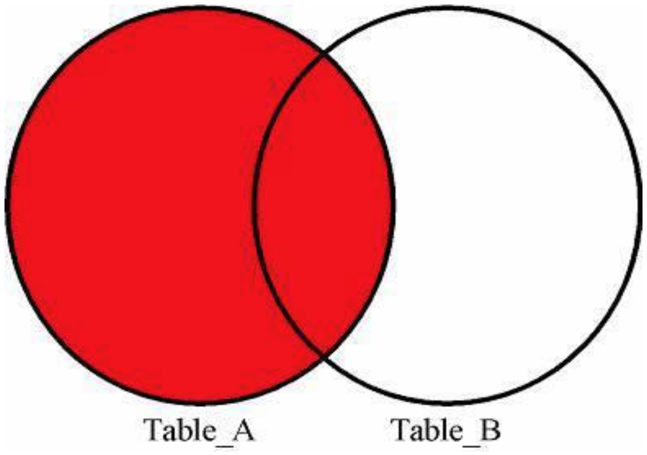
	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_NAME	LOCATION_ID
1	101	Neena	Kochhar	Executive	1700
2	102	Lex	De Haan	Executive	1700
3	104	Bruce	Ernst	IT	1400
4	105	David	Austin	IT	1400
5	106	Valli	Pataballa	IT	1400
6	107	Diana	Lorentz	IT	1400
7	109	Daniel	Faviet	Finance	1700
8	110	John	Chen	Finance	1700
9	111	Ismael	Sciarra	Finance	1700
10	112	Jose Manuel	Urman	Finance	1700
11	113	Luis	Popp	Finance	1700

OUTER JOIN

- Often times we need to return rows from one table even if there are no matching rows that are produced through a join condition. For this situation, we use outer joins.
- Outer joins:
  - **LEFT OUTER JOIN**
  - **RIGHT OUTER JOIN**
  - **FULL OUTER JOIN**

LEFT OUTER JOIN

- A left outer join is where the table, on the left of a **FROM** clause is required to **return all of its rows** regardless of having matching rows from the table it is being joined on.

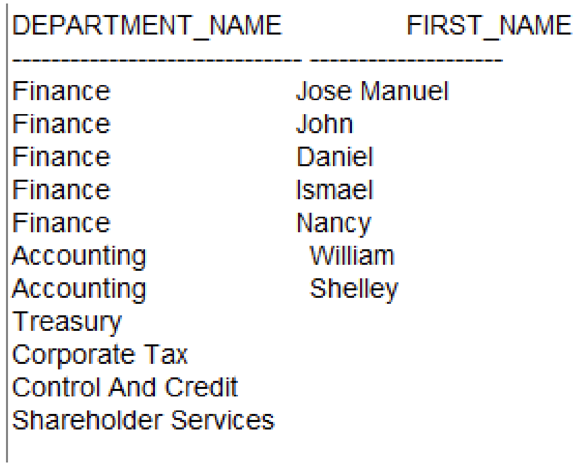


Using LEFT OUTER JOIN

Query 7:

- List all department with all its employees, including all department without employee.

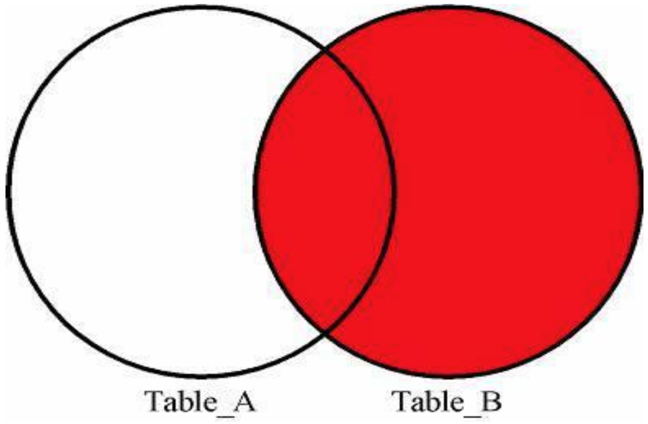
```
SELECT department_name, first_name
FROM departments
LEFT OUTER JOIN employees
ON (departments.department_id = employees.department_id);
```



```
DEPARTMENT_NAME            FIRST_NAME
------------------------------ --------------------
Finance                    Jose Manuel
Finance                    John
Finance                    Daniel
Finance                    Ismael
Finance                    Nancy
Accounting                  William
Accounting                  Shelley
Treasury
Corporate Tax
Control And Credit
Shareholder Services
```

RIGHT OUTER JOIN

- A right outer join is just the opposite of a left outer join. It states that you would like all rows from the right table in the **FROM** clause to be returned regardless of having a true match defined in the WHERE clause against the left side table in the FROM clause.

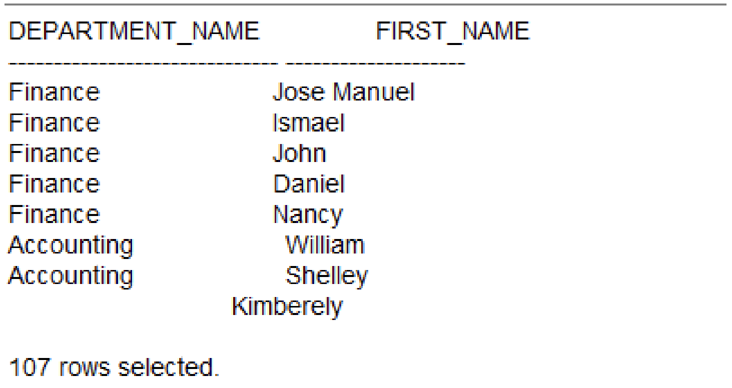


Using RIGHT OUTER JOIN

Query 8:

- List all employees with their departments, including all employees without department.

```
SELECT department_name, first_name
FROM departments
RIGHT OUTER JOIN employees
ON (departments.department_id = employees.department_id);
```

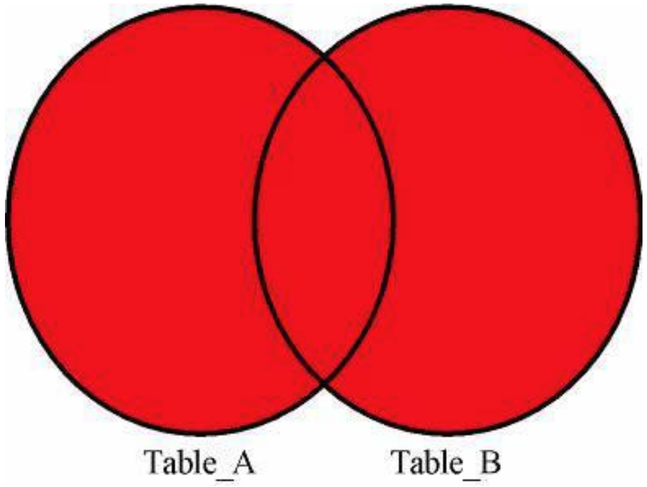


```
DEPARTMENT_NAME              FIRST_NAME
------------------------------ --------------------
Finance                      Jose Manuel
Finance                      Ismael
Finance                      John
Finance                      Daniel
Finance                      Nancy
Accounting                     William
Accounting                     Shelley
                         Kimberely

107 rows selected.
```

FULL OUTER JOIN

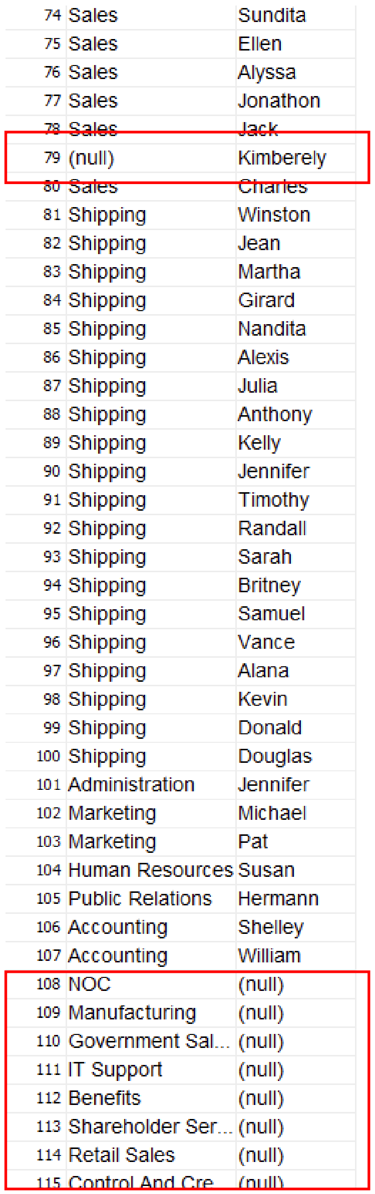
- Return both left and right sides of a query regardless of having a match.



Using FULL OUTER JOIN

- How can we get the records for all employees AND all departments whether they are missing data or not?

```
SELECT department_name, first_name
FROM departments
FULL OUTER JOIN employees
ON (departments.department_id = employees.department_id);
```



74	Sales	Sundita
75	Sales	Ellen
76	Sales	Alyssa
77	Sales	Jonathon
78	Sales	Jack
79	(null)	Kimberely
80	Sales	Charles
81	Shipping	Winston
82	Shipping	Jean
83	Shipping	Martha
84	Shipping	Girard
85	Shipping	Nandita
86	Shipping	Alexis
87	Shipping	Julia
88	Shipping	Anthony
89	Shipping	Kelly
90	Shipping	Jennifer
91	Shipping	Timothy
92	Shipping	Randall
93	Shipping	Sarah
94	Shipping	Britney
95	Shipping	Samuel
96	Shipping	Vance
97	Shipping	Alana
98	Shipping	Kevin
99	Shipping	Donald
100	Shipping	Douglas
101	Administration	Jennifer
102	Marketing	Michael
103	Marketing	Pat
104	Human Resources	Susan
105	Public Relations	Hermann
106	Accounting	Shelley
107	Accounting	William
108	NOC	(null)
109	Manufacturing	(null)
110	Government Sal...	(null)
111	IT Support	(null)
112	Benefits	(null)
113	Shareholder Ser...	(null)
114	Retail Sales	(null)
115	Control And Cre...	(null)