Returning to

SQL DML SELECT STATEMENT

Join and Union Queries

CT230

Database Systems

RECALL EXAMPLE 18:

Version 1: List the details (name and birth date) of the children of the employee with SSN 333445555

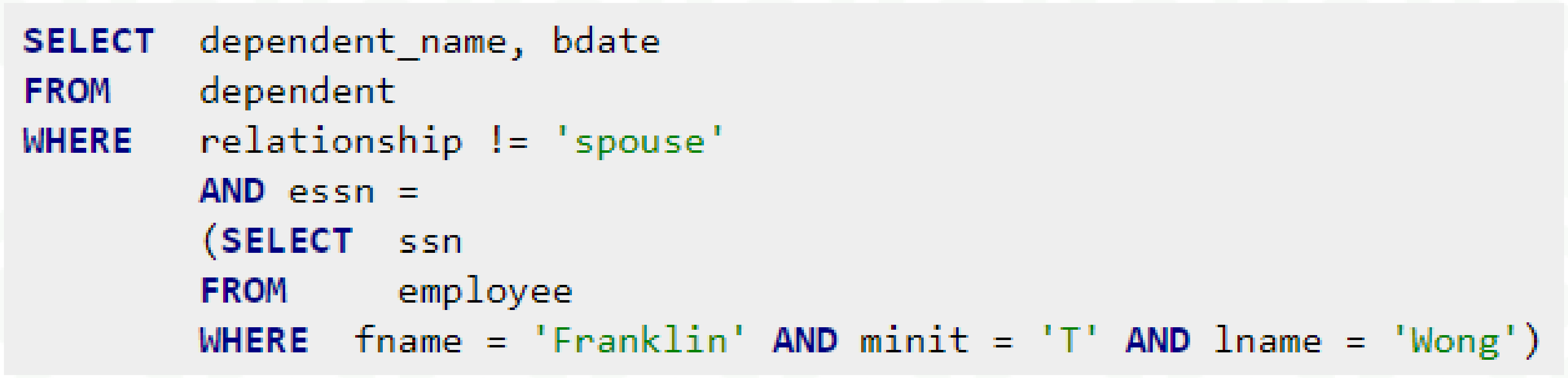
Version 2: List the details (name and birth date) of the children of Franklin T Wong?

Now consider a 3rd version:

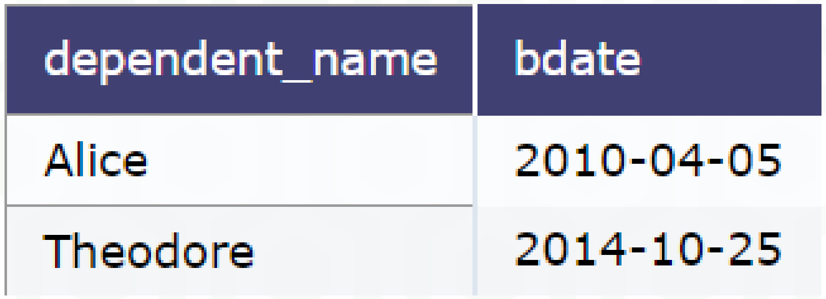
Version 3: List the details (name, birth date and address) of the children of Franklin T Wong (assuming the dependent's address is Franklin Wong's address)

RECALL sub-query solution to version 2:

List the details (name and birth date) of the children of Franklin T Wong?



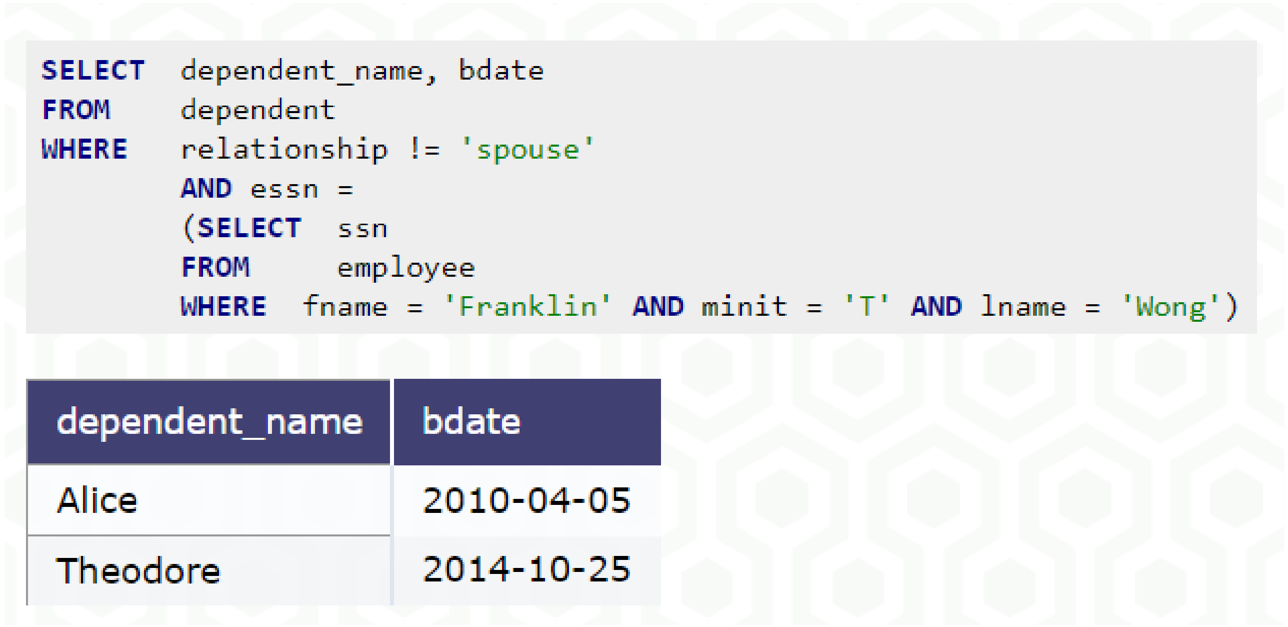
```
SELECT  dependent_name, bdate
FROM    dependent
WHERE   relationship != 'spouse'
        AND essn =
        (SELECT  ssn
        FROM     employee
        WHERE  fname = 'Franklin' AND minit = 'T' AND lname = 'Wong')
```



dependent_name	bdate
Alice	2010-04-05
Theodore	2014-10-25

CAN WE MODIFY THIS TO GET THE SOLUTION TO VERSION 3?

List the details (name, birth date and address) of the children of Franklin T Wong (assuming the dependent's address is Franklin Wong's address)



```
SELECT  dependent_name, bdate
FROM    dependent
WHERE   relationship != 'spouse'
        AND essn =
        (SELECT  ssn
        FROM     employee
        WHERE  fname = 'Franklin' AND minit = 'T' AND lname = 'Wong')
```

dependent_name	bdate
Alice	2010-04-05
Theodore	2014-10-25

No – because we need information from two tables –we need to use a *join* to join or *combine* the two tables so that the information from both is accessible and can be displayed as the output

JOINS

Joins combine multiple tables in to one table. This new (temporary) table is then queried to return results so we can return values from any of the tables which were joined.

Tables are joined by specifying links (or joins) across attributes in the tables.

Joins are carried out on 2 tables at a time but many tables can be joined, i.e., a third table can be joined to the table that results from joining two tables.

SPECIFYING JOINS

1. In SQL must specify all the tables which are part of join in the FROM clause
2. There are many different types of joins – all may not be supported in the DBMS you are using – we will mostly use an *inner join* which will always be supported.
3. Must then specify the join condition: for an inner join the condition is *foreign_key = primary_key/candidate_key*.
4. The join condition can be specified in the FROM or WHERE clause.

INNER JOINING TABLES:

The result of an inner join operation between two tables:

R(A_1, A_2, …, A_n) and

S (B_1, B_2, …, B_m)

is a table Q(A_1, A_2, …, A_n, B_1, B_2, …, B_m) where:

Q has one tuple for each combination of tuples (one from R and S) whenever the combination satisfies the join condition – the join will retrieve ALL attributes in each table

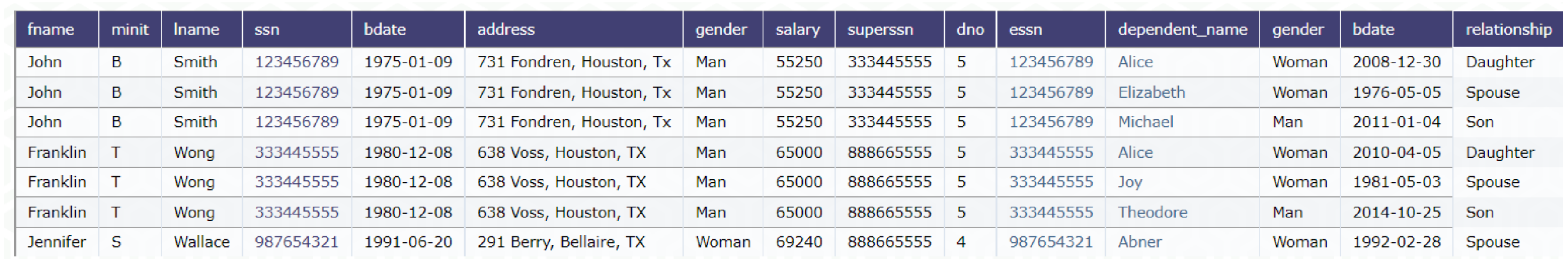
CONSIDER:
INNER JOIN CONDITION FOR employee AND dependent TABLES

Join condition: `ssn = essn`

Full query retrieving all employees and their dependents (when they have dependents):

```
SELECT *
FROM   employee INNER JOIN dependent
       ON ssn = essn;
```

Result from joining `employee` and `dependent`:

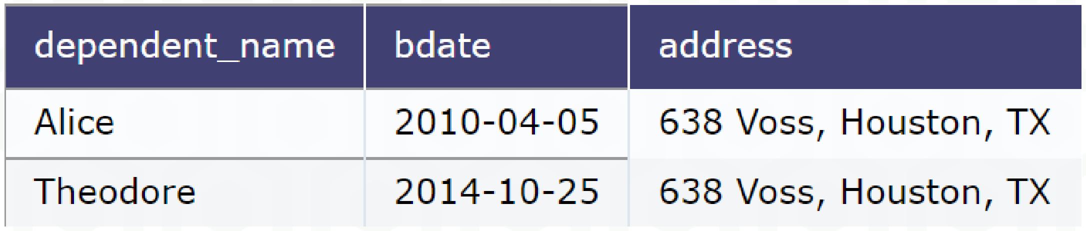


fname	minit	lname	ssn	bdate	address	gender	salary	superssn	dno	essn	dependent_name	gender	bdate	relationship
John	B	Smith	123456789	1975-01-09	731 Fondren, Houston, Tx	Man	55250	333445555	5	123456789	Alice	Woman	2008-12-30	Daughter
John	B	Smith	123456789	1975-01-09	731 Fondren, Houston, Tx	Man	55250	333445555	5	123456789	Elizabeth	Woman	1976-05-05	Spouse
John	B	Smith	123456789	1975-01-09	731 Fondren, Houston, Tx	Man	55250	333445555	5	123456789	Michael	Man	2011-01-04	Son
Franklin	T	Wong	333445555	1980-12-08	638 Voss, Houston, TX	Man	65000	888665555	5	333445555	Alice	Woman	2010-04-05	Daughter
Franklin	T	Wong	333445555	1980-12-08	638 Voss, Houston, TX	Man	65000	888665555	5	333445555	Joy	Woman	1981-05-03	Spouse
Franklin	T	Wong	333445555	1980-12-08	638 Voss, Houston, TX	Man	65000	888665555	5	333445555	Theodore	Man	2014-10-25	Son
Jennifer	S	Wallace	987654321	1991-06-20	291 Berry, Bellaire, TX	Woman	69240	888665555	4	987654321	Abner	Woman	1992-02-28	Spouse

EXAMPLE 18 VERSION 3 JOIN SOLUTION

List the details (name, birth date and address) of the children of Franklin T Wong

```
SELECT  dependent_name, dependent.bdate, address
FROM    employee INNER JOIN dependent ON
        ssn = essn
WHERE   relationship != ‘spouse’
        AND  fname = 'Franklin’
        AND  minit = 'T’
        AND  lname = 'Wong’;
```



dependent_name	bdate	address
Alice	2010-04-05	638 Voss, Houston, TX
Theodore	2014-10-25	638 Voss, Houston, TX

NOTE:

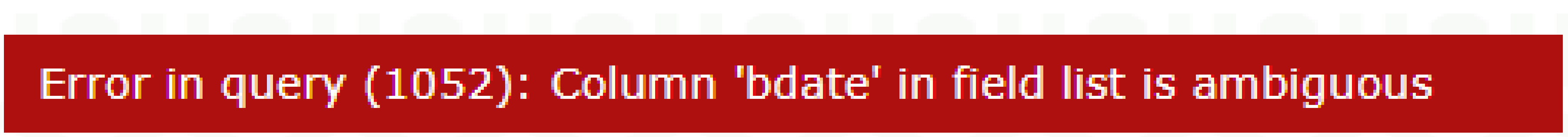
When attributes with the same name, but from different tables, are used in a join query, you need to specify the table name to avoid ambiguity with respect to the attribute names.

Example: bdate in employee and dependent relations.

Can refer to both of these unambiguously as:

```
employee.bdate

dependent.bdate
```

If you do not do this, the DBMS does not know which one you are referring to and gives an error:



```
Error in query (1052): Column 'bdate' in field list is ambiguous
```

EXAMPLE 39: Using an inner join, retrieve the names and addresses of all employees who work for the administration department

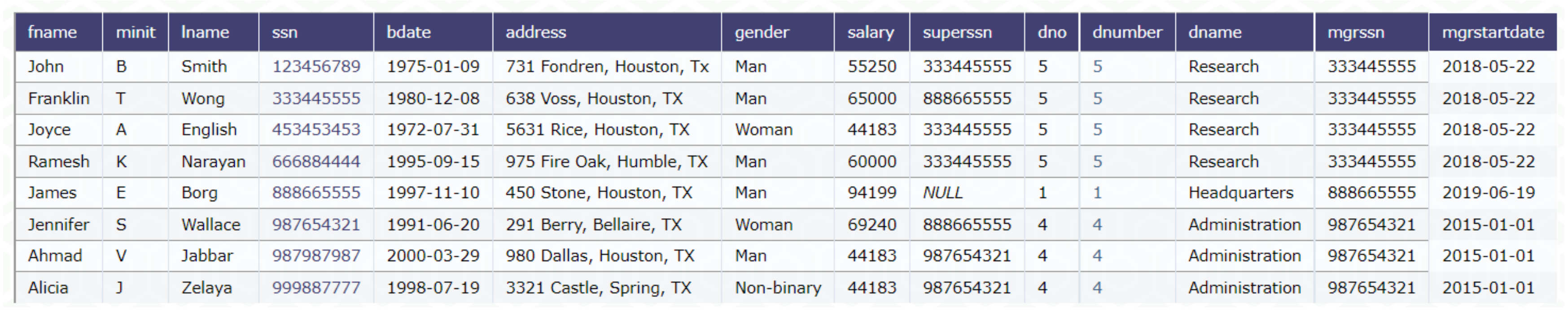
```
SELECT fname, lname, address

FROM   ???

WHERE  dname = 'administration’;
```

CONSIDER THE INNER JOIN CONDITION FOR `employee` AND `department` USING DEPARTMENT NUMBER

Join condition is: `dno = dnumber`

Full query retrieving all employees and their departments:

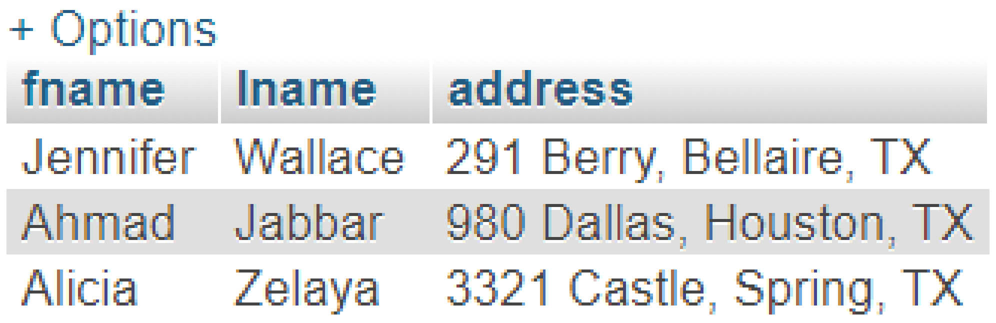
```
SELECT *
FROM   employee INNER JOIN department
       ON dno = dnumber;
```



fname	minit	lname	ssn	bdate	address	gender	salary	superssn	dno	dnumber	dname	mgrssn	mgrstartdate
John	B	Smith	123456789	1975-01-09	731 Fondren, Houston, Tx	Man	55250	333445555	5	5	Research	333445555	2018-05-22
Franklin	T	Wong	333445555	1980-12-08	638 Voss, Houston, TX	Man	65000	888665555	5	5	Research	333445555	2018-05-22
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	Woman	44183	333445555	5	5	Research	333445555	2018-05-22
Ramesh	K	Narayan	666884444	1995-09-15	975 Fire Oak, Humble, TX	Man	60000	333445555	5	5	Research	333445555	2018-05-22
James	E	Borg	888665555	1997-11-10	450 Stone, Houston, TX	Man	94199	*NULL*	1	1	Headquarters	888665555	2019-06-19
Jennifer	S	Wallace	987654321	1991-06-20	291 Berry, Bellaire, TX	Woman	69240	888665555	4	4	Administration	987654321	2015-01-01
Ahmad	V	Jabbar	987987987	2000-03-29	980 Dallas, Houston, TX	Man	44183	987654321	4	4	Administration	987654321	2015-01-01
Alicia	J	Zelaya	999887777	1998-07-19	3321 Castle, Spring, TX	Non-binary	44183	987654321	4	4	Administration	987654321	2015-01-01

EXAMPLE 39: Using a join, retrieve the names and addresses of all employees who work for the administration department

```
SELECT fname, lname, address
FROM   employee INNER JOIN department
       ON  employee.dno = department.dnumber
WHERE  dname = 'administration';
```


+ Options

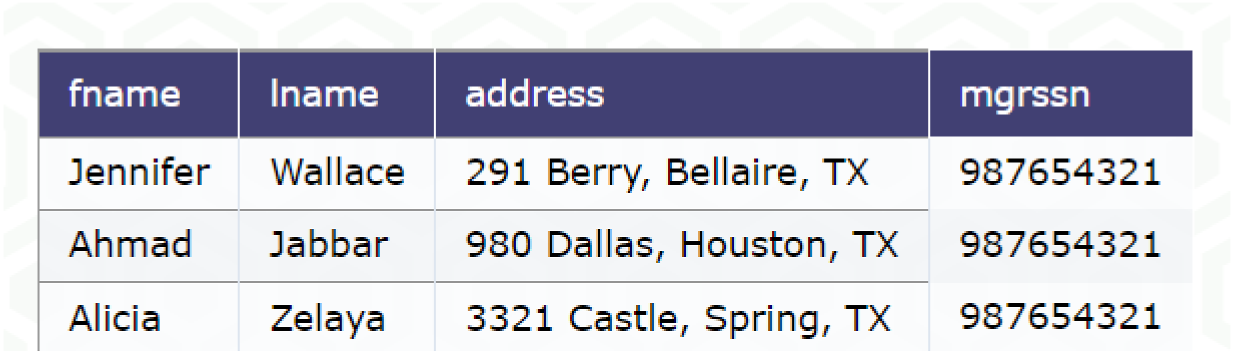
fname	lname	address
Jennifer	Wallace	291 Berry, Bellaire, TX
Ahmad	Jabbar	980 Dallas, Houston, TX
Alicia	Zelaya	3321 Castle, Spring, TX

Class Question: Can this be done with a sub-query?

Class Question: Can this be done with a sub-query? (EXAMPLE 39: Retrieve the names and addresses of all employees who work for the administration department)

EXAMPLE 40: Retrieve the names and addresses of all employees who work for the administration department and the ssn of the manager of the administration department

```
SELECT   fname, lname, address, mgrssn
FROM     employee INNER JOIN department
         ON employee.dno = department.dnumber
WHERE    dname = 'administration';
```



fname	lname	address	mgrssn
Jennifer	Wallace	291 Berry, Bellaire, TX	987654321
Ahmad	Jabbar	980 Dallas, Houston, TX	987654321
Alicia	Zelaya	3321 Castle, Spring, TX	987654321

IMPLICIT AND EXPLICIT JOINS

The join condition can be specified implicitly or explicitly as follows:

- An explicit join is specified in the `FROM` clause where the tables to be joined are listed. The keyword INNER JOIN is used for inner joins and the **join condition** is listed also using keyword ON
- An implicit join is specified in the `WHERE` clause without using the keyword ON. It is referred to as a **join condition.** The tables must be listed in the `FROM` clause, separated by commas. Other conditions can also be specified in the `WHERE` clause as well as the join condition.

IMPLICIT JOIN CONDITION IN WHERE CLAUSE:

- No additional syntax to learn.
- All tables involved *MUST* be listed in FROM clause.
- Condition to join tables is contained in the WHERE clause. If there are other conditions, the join condition is appended on with AND
- This is an **INNER JOIN**: all rows from both tables will be returned whenever there is a match between the attributes in the join condition

EXPLICIT JOIN CONDITION IN FROM CLAUSE

Syntax for joining 2 tables:

```
SELECT [DISTINCT] <attribute list>
FROM   <table>
        [INNER/LEFT/RIGHT] JOIN <table>
        ON <join condition>
WHERE  <condition>
```

* Will mostly use INNER JOIN

EXAMPLE 18 AGAIN … USING AN IMPLICT JOIN

List the details (name, birth date and address) of the children of Franklin T Wong

EXAMPLE 39 again: Retrieve the names and addresses of all employees who work for the administration department (using an implicit join)

```
SELECT fname, lname, address

FROM   ??

WHERE  dname = 'administration';
```

Syntax of **explicit join** with 3 tables

```
SELECT [DISTINCT] <attribute list>
FROM   (<table>
        [INNER/LEFT/RIGHT] JOIN <table>
        ON <join condition>)
        [INNER/LEFT/RIGHT] JOIN <table>
        ON <join condition>
WHERE <condition>
```

Syntax of **implicit join** with 3 tables

```
SELECT [DISTINCT] <attribute list>
FROM  <table>,<table>,<table>
WHERE <join condition> AND
      <join condition> AND
      <condition>
```

Syntax of **explicit join** with 4 tables

```
SELECT [DISTINCT] <attribute list>
FROM   ((<table>
        [INNER/LEFT/RIGHT] JOIN <table>
        ON <join condition>)
        [INNER/LEFT/RIGHT] JOIN <table>
        ON <join condition>)
        [INNER/LEFT/RIGHT] JOIN <table>
        ON <join condition>
WHERE <condition>
```

Syntax of **implicit join** with 4 tables

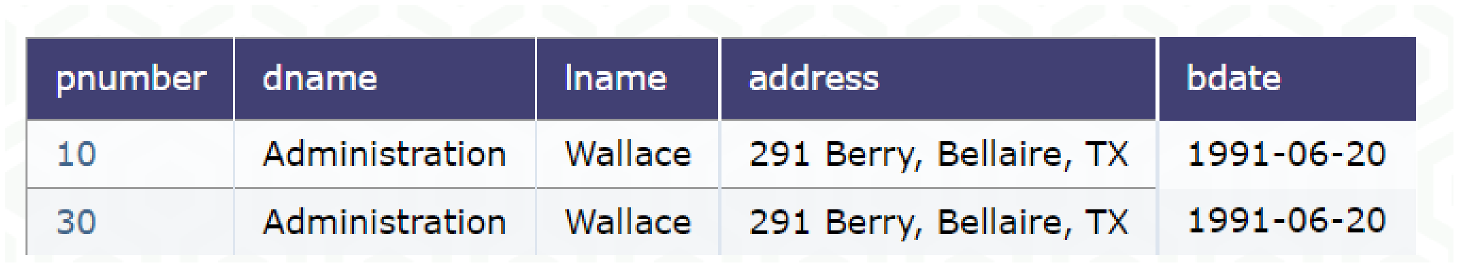
```
SELECT [DISTINCT] <attribute list>
FROM   <table>,<table>,<table>,<table>
WHERE <join condition> AND
      <join condition> AND
      <join condition> AND
      <condition>
```

EXAMPLE 41

For every project located in Stafford, list the project number, the controlling department name, and the department manager's surname, address and birth date.

EXAMPLE 41

```
SELECT pnumber, dname, lname, address, bdate
FROM   project INNER JOIN department
       ON project.dnum = department.dnumber
       INNER JOIN employee
       ON department.mgrssn = employee.ssn
WHERE  plocation = 'stafford';
```



pnumber	dname	lname	address	bdate
10	Administration	Wallace	291 Berry, Bellaire, TX	1991-06-20
30	Administration	Wallace	291 Berry, Bellaire, TX	1991-06-20

CLASS QUESTION:

> Re-write solution to example 41 using implicit joins?
> Can we re-write this using sub-queries?

DIFFERENT TYPES OF JOINS:

- Inner Join is the default when using Implicit Join
- An `INNER JOIN` includes the tuples from the first (left) of the two tables **only** when they satisfy the join condition and tuples from the second (right) table are **only** included when they also satisfy the join condition
- For explicit joins, should explicitly state the join used:

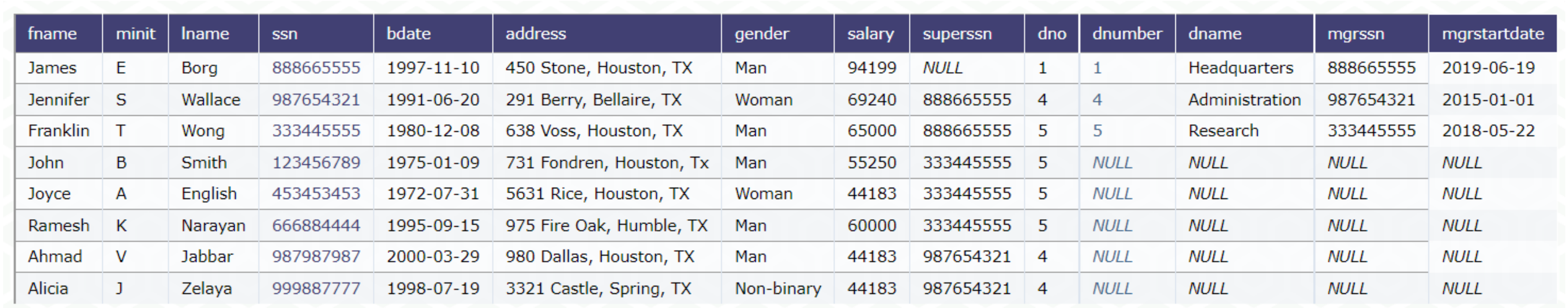
For example joining employee and department on ssn and mgrssn:

```
SELECT *
FROM     employee INNER JOIN department ON
         employee.ssn = department.mgrssn;
```

LEFT JOINS

Left (outer) joins include all of the tuples from the first (left) of two tables – when they satisfy the join condition and <u>even when they don't.</u> Tuples from the second (right) table are only included when they satisfy the join condition. Example:

```
SELECT *
FROM     employee LEFT JOIN department ON
         employee.ssn = department.mgrssn;
```

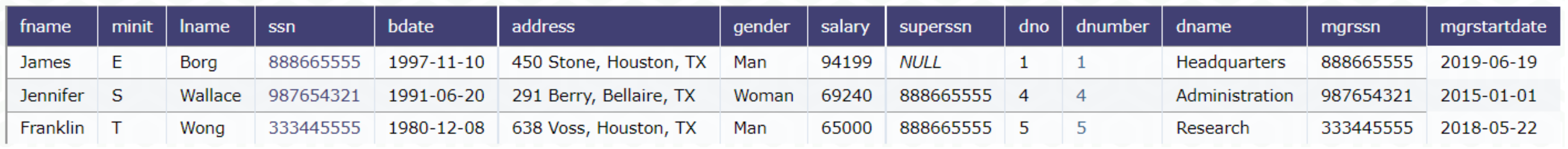


fname	minit	lname	ssn	bdate	address	gender	salary	superssn	dno	dnumber	dname	mgrssn	mgrstartdate
James	E	Borg	888665555	1997-11-10	450 Stone, Houston, TX	Man	94199	*NULL*	1	1	Headquarters	888665555	2019-06-19
Jennifer	S	Wallace	987654321	1991-06-20	291 Berry, Bellaire, TX	Woman	69240	888665555	4	4	Administration	987654321	2015-01-01
Franklin	T	Wong	333445555	1980-12-08	638 Voss, Houston, TX	Man	65000	888665555	5	5	Research	333445555	2018-05-22
John	B	Smith	123456789	1975-01-09	731 Fondren, Houston, Tx	Man	55250	333445555	5	*NULL*	*NULL*	*NULL*	*NULL*
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	Woman	44183	333445555	5	*NULL*	*NULL*	*NULL*	*NULL*
Ramesh	K	Narayan	666884444	1995-09-15	975 Fire Oak, Humble, TX	Man	60000	333445555	5	*NULL*	*NULL*	*NULL*	*NULL*
Ahmad	V	Jabbar	987987987	2000-03-29	980 Dallas, Houston, TX	Man	44183	987654321	4	*NULL*	*NULL*	*NULL*	*NULL*
Alicia	J	Zelaya	999887777	1998-07-19	3321 Castle, Spring, TX	Non-binary	44183	987654321	4	*NULL*	*NULL*	*NULL*	*NULL*

RIGHT JOINS

Right outer joins include **all** of the tuples from the second (right) of two tables, even if there are no matching values for records in the first (left) table. Tuples from the first (left) table are included **only** if they satisfy the join condition. Example:

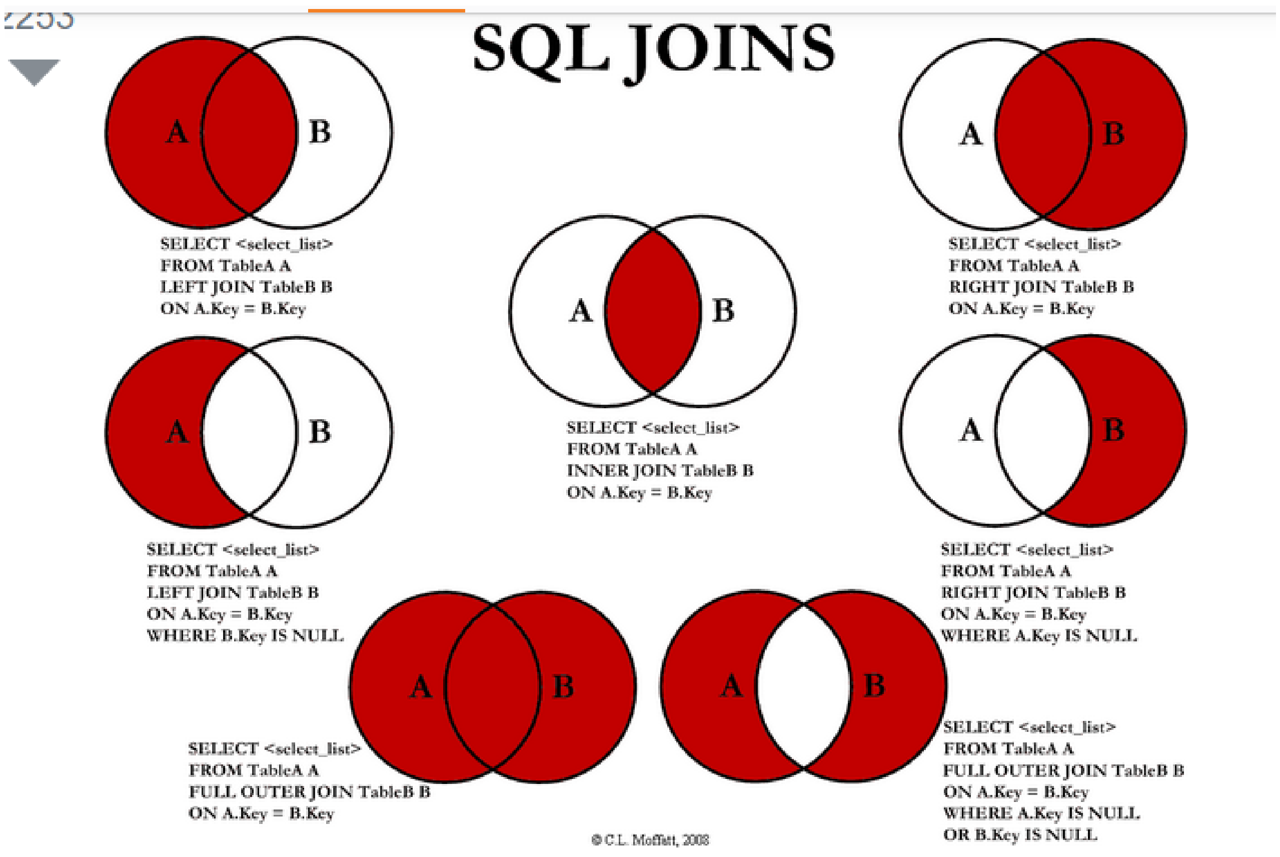
```
SELECT    *
FROM      employee RIGHT JOIN department ON
          employee.ssn = department.mgrssn;
```



fname	minit	lname	ssn	bdate	address	gender	salary	superssn	dno	dnumber	dname	mgrssn	mgrstartdate
James	E	Borg	888665555	1997-11-10	450 Stone, Houston, TX	Man	94199	*NULL*	1	1	Headquarters	888665555	2019-06-19
Jennifer	S	Wallace	987654321	1991-06-20	291 Berry, Bellaire, TX	Woman	69240	888665555	4	4	Administration	987654321	2015-01-01
Franklin	T	Wong	333445555	1980-12-08	638 Voss, Houston, TX	Man	65000	888665555	5	5	Research	333445555	2018-05-22

Graphical representation of different types of joins (C.L. Moffat, 2008)

In MySQL only INNER, LEFT and RIGHT joins are supported



```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

EXAMPLE 42: What is the difference in the output produced using INNER, LEFT and RIGHT joins in the following?

```
SELECT  *
FROM    employee [INNER/LEFT/RIGHT] JOIN dependent
          ON employee.ssn = dependent.essn;
```

SELF-JOINS AND ALIASES

A self-join is a normal SQL join that joins a table to itself.

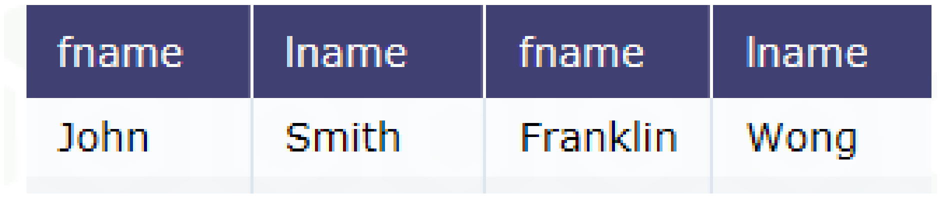
This is accomplished by using aliases to give each “instance” of the table a separate name – the keyword AS is used.

EXAMPLE 43: For each employee, retrieve the employee’s name and the name of the employee’s supervisor

Consider:

1. How to write the query if asked for the employee’s name and supervisor’s SSN?

2. What should output look like? e.g., for John Smith:



fname	lname	fname	lname
John	Smith	Franklin	Wong

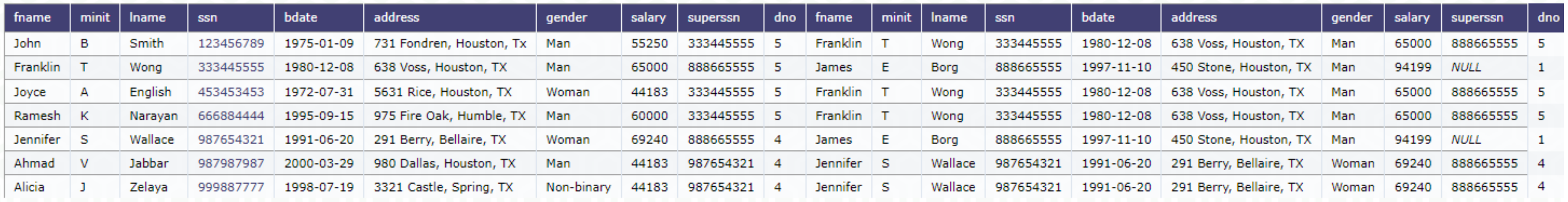
First consider joining employee to itself …

Need two "copies" or instances of table employee…

Call them E (for employee) and S (for supervisor)

```
SELECT *
FROM   employee AS e, employee AS s
WHERE  e.superssn = s.ssn;
```

```
SELECT *
FROM   employee AS e INNER JOIN employee AS s
       ON  e.superssn = s.ssn;
```

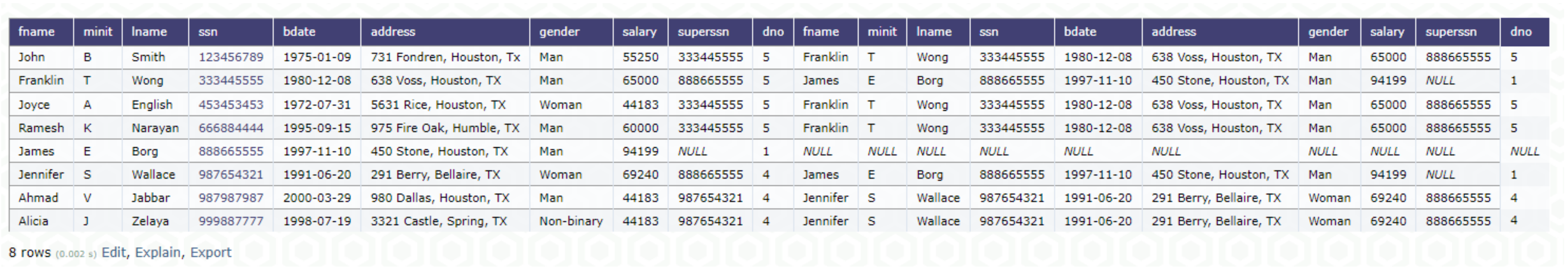


fname	minit	lname	ssn	bdate	address	gender	salary	superssn	dno	fname	minit	lname	ssn	bdate	address	gender	salary	superssn	dno
John	B	Smith	123456789	1975-01-09	731 Fondren, Houston, Tx	Man	55250	333445555	5	Franklin	T	Wong	333445555	1980-12-08	638 Voss, Houston, TX	Man	65000	888665555	5
Franklin	T	Wong	333445555	1980-12-08	638 Voss, Houston, TX	Man	65000	888665555	5	James	E	Borg	888665555	1997-11-10	450 Stone, Houston, TX	Man	94199	NULL	1
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	Woman	44183	333445555	5	Franklin	T	Wong	333445555	1980-12-08	638 Voss, Houston, TX	Man	65000	888665555	5
Ramesh	K	Narayan	666884444	1995-09-15	975 Fire Oak, Humble, TX	Man	60000	333445555	5	Franklin	T	Wong	333445555	1980-12-08	638 Voss, Houston, TX	Man	65000	888665555	5
Jennifer	S	Wallace	987654321	1991-06-20	291 Berry, Bellaire, TX	Woman	69240	888665555	4	James	E	Borg	888665555	1997-11-10	450 Stone, Houston, TX	Man	94199	NULL	1
Ahmad	V	Jabbar	987987987	2000-03-29	980 Dallas, Houston, TX	Man	44183	987654321	4	Jennifer	S	Wallace	987654321	1991-06-20	291 Berry, Bellaire, TX	Woman	69240	888665555	4
Alicia	J	Zelaya	999887777	1998-07-19	3321 Castle, Spring, TX	Non-binary	44183	987654321	4	Jennifer	S	Wallace	987654321	1991-06-20	291 Berry, Bellaire, TX	Woman	69240	888665555	4

Why is this version better?

"For each employee, retrieve the employee's name and the name of the employee's supervisor"

```
SELECT *
FROM   employee AS e LEFT JOIN employee AS s
       ON  e.superssn = s.ssn;
```

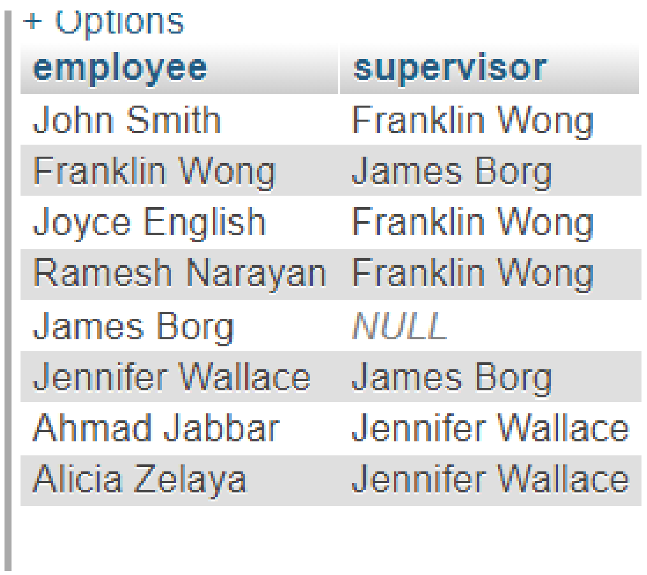


fname	minit	lname	ssn	bdate	address	gender	salary	superssn	dno	fname	minit	lname	ssn	bdate	address	gender	salary	superssn	dno
John	B	Smith	123456789	1975-01-09	731 Fondren, Houston, Tx	Man	55250	333445555	5	Franklin	T	Wong	333445555	1980-12-08	638 Voss, Houston, TX	Man	65000	888665555	5
Franklin	T	Wong	333445555	1980-12-08	638 Voss, Houston, TX	Man	65000	888665555	5	James	E	Borg	888665555	1997-11-10	450 Stone, Houston, TX	Man	94199	NULL	1
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	Woman	44183	333445555	5	Franklin	T	Wong	333445555	1980-12-08	638 Voss, Houston, TX	Man	65000	888665555	5
Ramesh	K	Narayan	666884444	1995-09-15	975 Fire Oak, Humble, TX	Man	60000	333445555	5	Franklin	T	Wong	333445555	1980-12-08	638 Voss, Houston, TX	Man	65000	888665555	5
James	E	Borg	888665555	1997-11-10	450 Stone, Houston, TX	Man	94199	NULL	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
Jennifer	S	Wallace	987654321	1991-06-20	291 Berry, Bellaire, TX	Woman	69240	888665555	4	James	E	Borg	888665555	1997-11-10	450 Stone, Houston, TX	Man	94199	NULL	1
Ahmad	V	Jabbar	987987987	2000-03-29	980 Dallas, Houston, TX	Man	44183	987654321	4	Jennifer	S	Wallace	987654321	1991-06-20	291 Berry, Bellaire, TX	Woman	69240	888665555	4
Alicia	J	Zelaya	999887777	1998-07-19	3321 Castle, Spring, TX	Non-binary	44183	987654321	4	Jennifer	S	Wallace	987654321	1991-06-20	291 Berry, Bellaire, TX	Woman	69240	888665555	4

8 rows (0.002 s) Edit, Explain, Export

EXAMPLE 43: For each employee, retrieve the employee's name and the name of the employee's supervisor

```
SELECT  CONCAT(e.fname, ' ' , e.lname) AS employee,
        CONCAT(s.fname, ' ' , s.lname) AS supervisor
FROM    employee AS e LEFT JOIN employee AS s
        ON  e.superssn = s.ssn;
```


+ Options

employee	supervisor
John Smith	Franklin Wong
Franklin Wong	James Borg
Joyce English	Franklin Wong
Ramesh Narayan	Franklin Wong
James Borg	*NULL*
Jennifer Wallace	James Borg
Ahmad Jabbar	Jennifer Wallace
Alicia Zelaya	Jennifer Wallace

EXAMPLE 44: For each department, list the department name, and the names, addresses and the start date of all managers, ordered by department name

```
SELECT

FROM

WHERE

ORDER BY          ;
```

CAN SUB-QUERIES AND JOINS BE USED INTERCHANGEABLY?

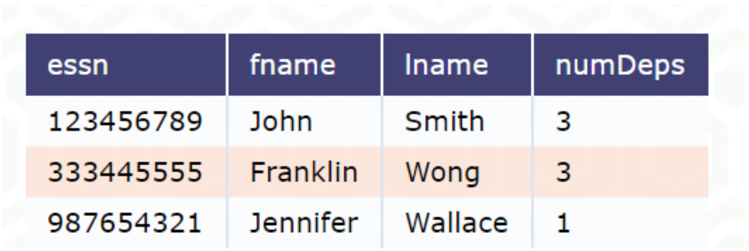
In some cases, yes, can replace a join of tables (where appropriate) with a sub-query

But recall …

- Joins are needed when values across multiple tables must be displayed.
- Sub-queries are needed when an existing value from a table needs to be retrieved and used as part of the query solution.
- Sub-queries are needed when an aggregate function needs to be performed and used as part of a query solution.

EXAMPLE 45: JOINS AND GROUP BY

List the employee name, and number of dependents of each employee who has dependents

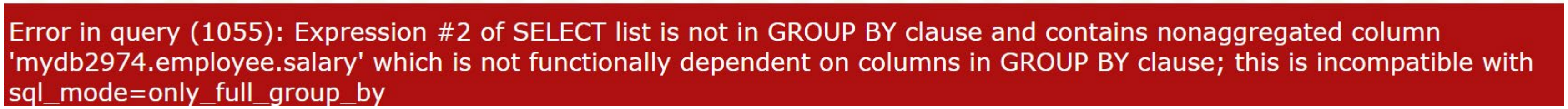


essn	fname	lname	numDeps
123456789	John	Smith	3
333445555	Franklin	Wong	3
987654321	Jennifer	Wallace	1

```
SELECT      essn, fname, lname,
            COUNT(*) AS numDeps
FROM        employee INNER JOIN dependent
            ON ssn = essn
GROUP BY    essn, fname, lname;
```

Why won’t this work?

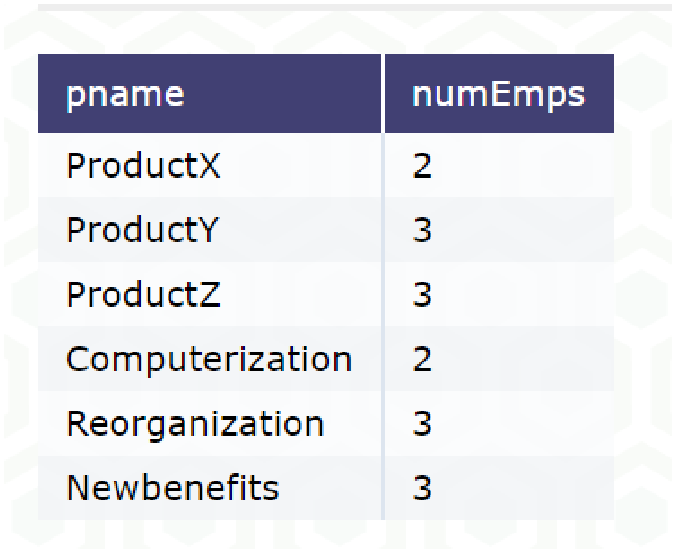
```
SELECT      essn, fname, lname,  COUNT(*) AS numDeps
FROM        employee INNER JOIN dependent
            ON ssn = essn
GROUP BY    essn;
```



```
Error in query (1055): Expression #2 of SELECT list is not in GROUP BY clause and contains nonaggregated column
'mydb2974.employee.salary' which is not functionally dependent on columns in GROUP BY clause; this is incompatible with
sql_mode=only_full_group_by
```

EXAMPLE 46: List the project name and the number of employees who work on the project for projects that have 2 or more employees

```
SELECT      pname,
            COUNT(*) AS numEmps
FROM
GROUP BY
HAVING
```



pname	numEmps
ProductX	2
ProductY	3
ProductZ	3
Computerization	2
Reorganization	3
Newbenefits	3

UNION QUERIES

The keyword UNION is used to combine the results of two or more queries or tables

MySQL does not support minus or intersection (intersect) operators but the same functionality can be built using joins

For union queries, tables must be **union compatible**

UNION COMPATIBLE

Two relations are **union compatible** if the schemas of the two relations match, i.e.,

same number of attributes in each relation and each pair of corresponding attributes have the same domain

Example 47: Using both subqueries and union queries (no joins) list all project numbers for projects that involve a worker whose last name is 'Wallace' or a manager, of the department that controls the project, with last name 'Wallace'

Steps:

First, consider two queries on their own and these can be combined with a Union query:

Query 1. Finding the employees 'Wallace' working on projects …

Query 2. Finding the manger 'Wallace' of a department that controls project

Example 47: Using both subqueries and union queries (no joins) list all project numbers for projects that involve a worker whose last name is 'Wallace' or a manager, of the department that controls the project, with last name 'Wallace'

```
-- employee
SELECT pno
FROM  works_on
WHERE essn IN
 (SELECT ssn
  FROM employee
  WHERE lname =
 'Wallace');
```

```
-- manager
SELECT pnumber
FROM project
WHERE dnum IN
 (SELECT dnumber
  FROM department
  WHERE mgrssn IN
      (SELECT ssn
       FROM employee
       WHERE lname =
 'Wallace'));
```

EXAMPLE 47 Full solution

```
(SELECT pno
 FROM   works_on
 WHERE  essn IN
  (SELECT ssn  FROM employee
   WHERE lname = 'Wallace'))
UNION
(SELECT pnumber
 FROM   project
 WHERE  dnum IN (SELECT dnumber FROM department
 WHERE  mgrssn IN (SELECT ssn   FROM employee
                  WHERE  lname = 'Wallace')));
```

MORE EXAMPLES

Example 48

Using a join, list all the locations of the research department

Example 49

For all projects located in 'Houston' list the name of the project and the department which controls the project

Example 50

List the names of employees, and the number of hours they work, for employees who work greater than the average number of hours

SUMMARY: JOINS AND UNION QUERIES

Important to know:

- How joins work in general
- How implicit and explicit inner joins work
- How left and right joins work
- When to use sub-queries and joins
- How Union queries work