CS4.301 Data & Applications

Ponnurangam Kumaraguru ("PK")
#ProfGiri @ IIIT Hyderabad

pk.profgiri /in/ponguru @ponguru Ponnurangam.kumaraguru

Aliasing,Renaming and Tuple Variables (contd.)

The attribute names can also be renamed

```
EMPLOYEE AS E(Fn, Mi, Ln, Ssn, Bd, Addr, Sex,
Sal, Sssn, Dno)
```

Note that the relation EMPLOYEE now has a variable name E which corresponds to a tuple variable

The “AS” may be dropped in most SQL implementations

Nested Queries, Tuples, and Set/Multiset Comparisons

Nested queries

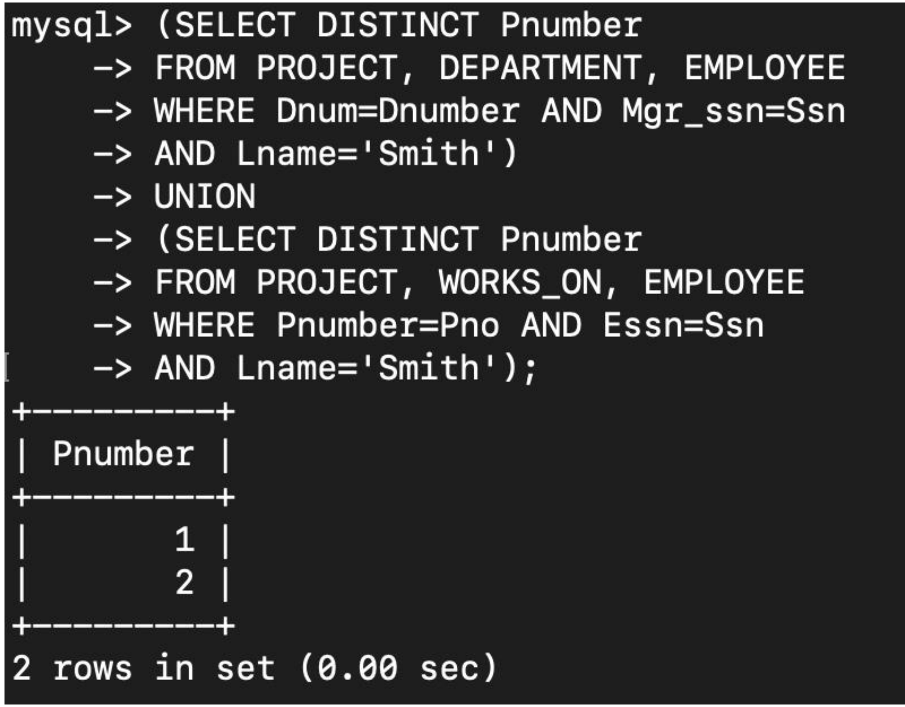
- Complete select-from-where blocks within WHERE clause of another query
- **Outer query and nested subqueries**

Comparison operator `IN`

- Compares value *v* with a set (or multiset) of values *V*
- Evaluates to `TRUE` if *v* is one of the elements in *V*

Nested Queries (cont'd.)

```
SELECT      DISTINCT    Pnumber
FROM        PROJECT
WHERE       Pnumber IN
            (SELECT     Pnumber
            FROM        PROJECT, DEPARTMENT, EMPLOYEE
            WHERE       Dnum = Dnumber AND
            Mgr_ssn = Ssn and Lname = 'Smith')
            OR
            Pnumber IN
            (SELECT     Pno
            FROM        WORKS_ON, EMPLOYEE
            WHERE       Essn = Ssn AND Lname = 'Smith');
```



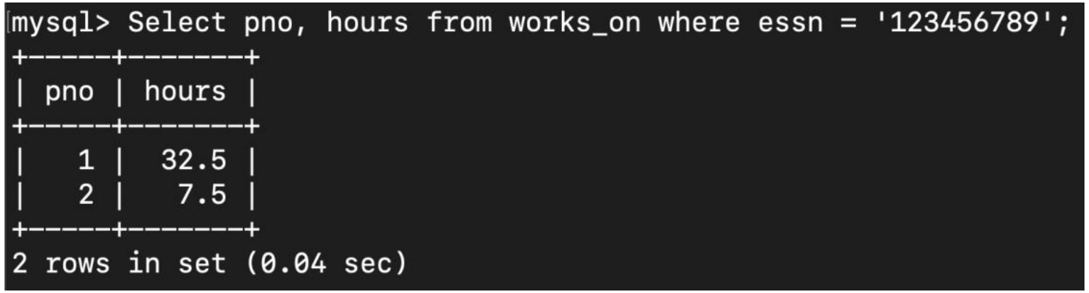
```
mysql> (SELECT DISTINCT Pnumber
    -> FROM PROJECT, DEPARTMENT, EMPLOYEE
    -> WHERE Dnum=Dnumber AND Mgr_ssn=Ssn
    -> AND Lname='Smith')
    -> UNION
    -> (SELECT DISTINCT Pnumber
    -> FROM PROJECT, WORKS_ON, EMPLOYEE
    -> WHERE Pnumber=Pno AND Essn=Ssn
    -> AND Lname='Smith');
+---------+
| Pnumber |
+---------+
|       1 |
|       2 |
+---------+
2 rows in set (0.00 sec)
```

Nested Queries (cont'd.)

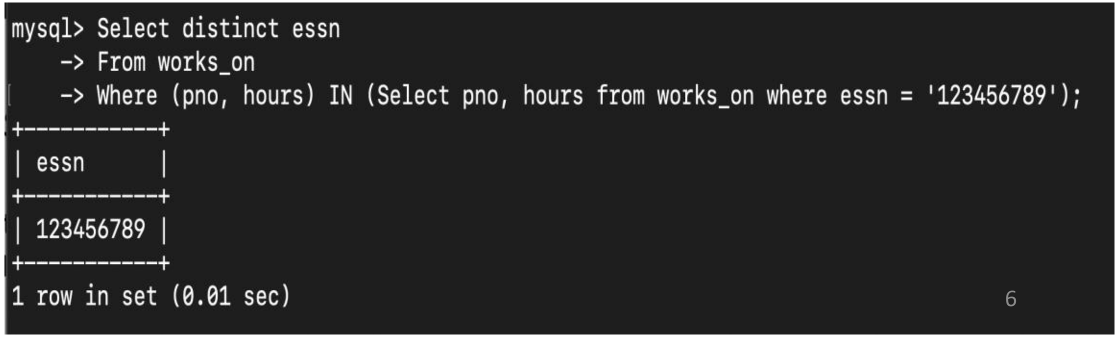
Use tuples of values in comparisons

Place them within parentheses

Select distinct essn
From works_on
Where (pno, hours) IN
(Select pno, hours from
works_on where essn =
'123456789');



```
mysql> Select pno, hours from works_on where essn = '123456789';
+-----+-------+
| pno | hours |
+-----+-------+
|   1 |  32.5 |
|   2 |   7.5 |
+-----+-------+
2 rows in set (0.04 sec)
```

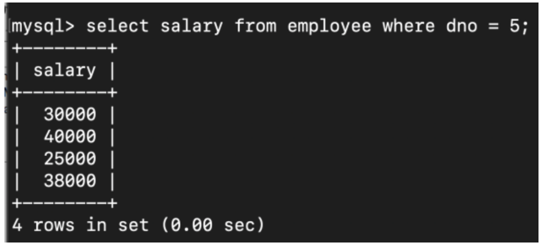


```
mysql> Select distinct essn
    -> From works_on
    -> Where (pno, hours) IN (Select pno, hours from works_on where essn = '123456789');
+-----------+
| essn      |
+-----------+
| 123456789 |
+-----------+
1 row in set (0.01 sec)
```

Nested Queries (cont'd.)

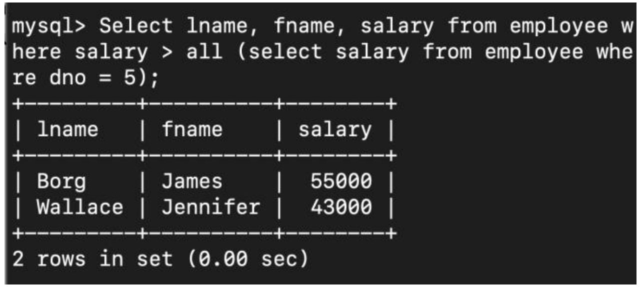
Use other comparison operators to compare a single value *v*

- `= ANY` (or `= SOME`) operator
  - Returns `TRUE` if the value *v* is equal to some value in the set *V* and is hence equivalent to `IN`
- Other operators that can be combined with `ANY` (or `SOME`): >, >=, <, <=, and <>
- `ALL`: value must exceed all values from nested query



```
mysql> select salary from employee where dno = 5;
+--------+
| salary |
+--------+
|  30000 |
|  40000 |
|  25000 |
|  38000 |
+--------+
4 rows in set (0.00 sec)
```

Select lname, fname, salary from employee where salary > all (select salary from employee where dno = 5);



```
mysql> Select lname, fname, salary from employee w
here salary > all (select salary from employee whe
re dno = 5);
+---------+----------+--------+
| lname   | fname    | salary |
+---------+----------+--------+
| Borg    | James    |  55000 |
| Wallace | Jennifer |  43000 |
+---------+----------+--------+
2 rows in set (0.00 sec)
```

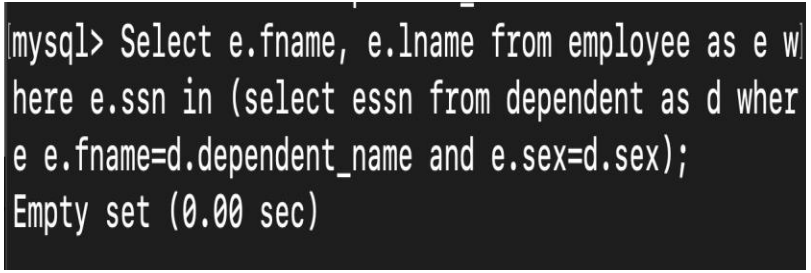
Nested Queries (cont'd.)

Avoid potential errors and ambiguities

Create tuple variables (aliases) for all tables referenced in SQL query

Query 16. Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee.

Select e.fname, e.lname from employee as e where e.ssn in (select essn from dependent as d where e.fname=d.dependent_name and e.sex=d.sex);



```
mysql> Select e.fname, e.lname from employee as e w
here e.ssn in (select essn from dependent as d wher
e e.fname=d.dependent_name and e.sex=d.sex);
Empty set (0.00 sec)
```

Correlated Nested Queries

Queries that are nested using the = or IN comparison operator can be collapsed into one single block, last query can be changed like

```
Select e.fname, e.lname from
employee as e where e.ssn IN
(select essn from dependent as
d where
e.fname=d.dependent_name
and e.sex=d.sex);
```

```
SELECT E.Fname, E.Lname
FROM EMPLOYEE AS E,
DEPENDENT AS D WHERE
E.Ssn=D.Essn AND E.Sex=D.Sex
AND
E.Fname=D.Dependent_name;
```

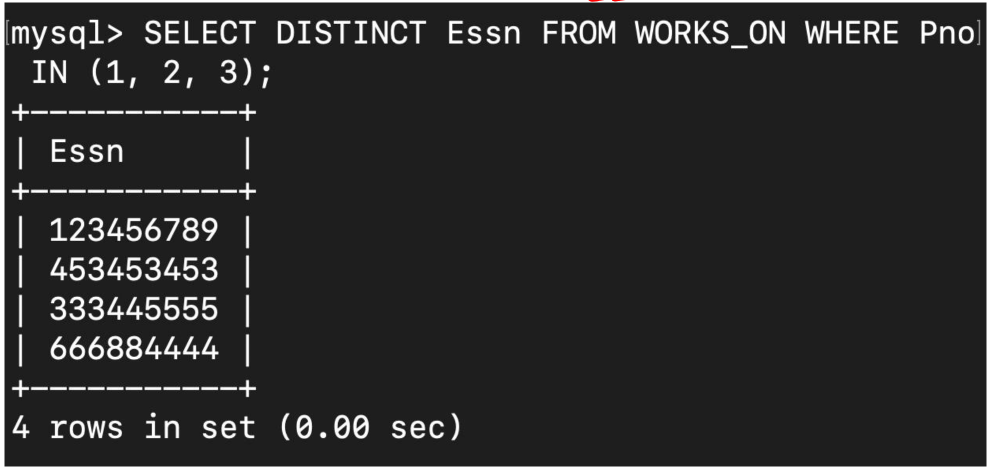
Correlated nested query

Evaluated once for each tuple in the outer query

Explicit Sets and Renaming of Attributes in SQL

Can use explicit set of values in WHERE clause

SELECT DISTINCT Essn FROM WORKS_ON WHERE   Pno IN $\boxed{(1, 2, 3);}$



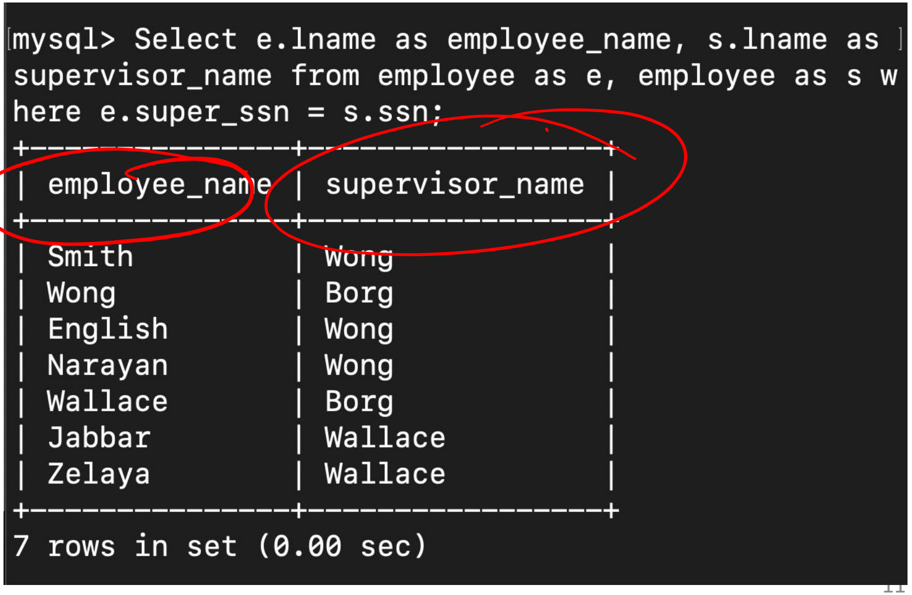
```
mysql> SELECT DISTINCT Essn FROM WORKS_ON WHERE Pno
 IN (1, 2, 3);
+-----------+
| Essn      |
+-----------+
| 123456789 |
| 453453453 |
| 333445555 |
| 666884444 |
+-----------+
4 rows in set (0.00 sec)
```

Explicit Sets and Renaming of Attributes in SQL

Use qualifier AS followed by desired new name

Rename any attribute that appears in the result of a query

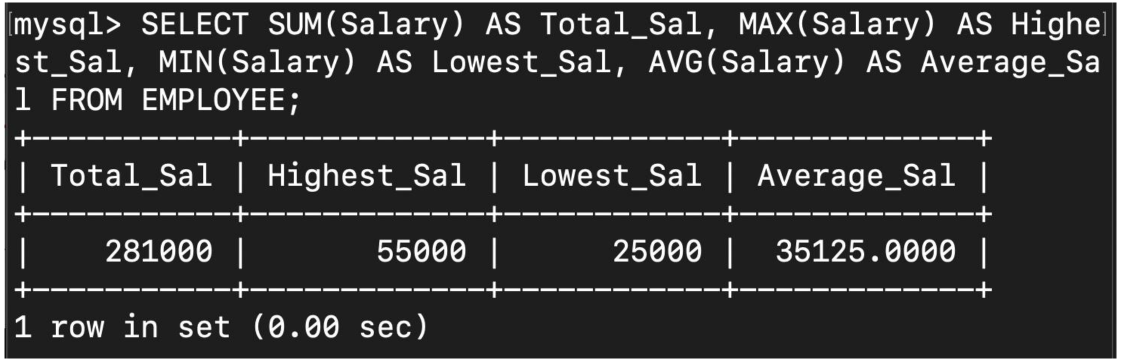
Select e.lname as employee_name, s.lname as supervisor_name from employee as e, employee as s where e.super_ssn = s.ssn;



```
mysql> Select e.lname as employee_name, s.lname as
supervisor_name from employee as e, employee as s w
here e.super_ssn = s.ssn;
+---------------+-----------------+
| employee_name | supervisor_name |
+---------------+-----------------+
| Smith         | Wong            |
| Wong          | Borg            |
| English       | Wong            |
| Narayan       | Wong            |
| Wallace       | Borg            |
| Jabbar        | Wallace         |
| Zelaya        | Wallace         |
+---------------+-----------------+
7 rows in set (0.00 sec)
```

Renaming Results of Aggregation

```
SELECT SUM(Salary) AS
Total_Sal, MAX(Salary)
AS Highest_Sal,
MIN(Salary) AS
Lowest_Sal, AVG(Salary)
AS Average_Sal FROM
EMPLOYEE;
```

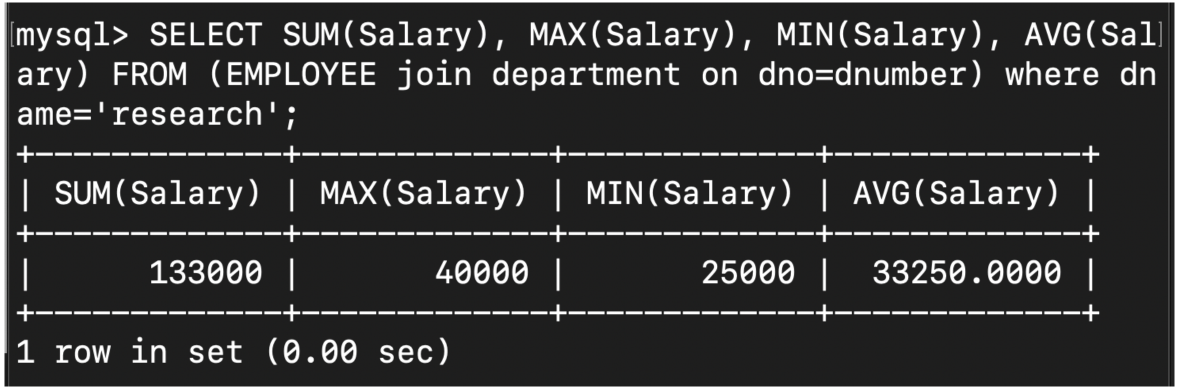


```
mysql> SELECT SUM(Salary) AS Total_Sal, MAX(Salary) AS Highe
st_Sal, MIN(Salary) AS Lowest_Sal, AVG(Salary) AS Average_Sa
l FROM EMPLOYEE;
+-----------+-------------+------------+-------------+
| Total_Sal | Highest_Sal | Lowest_Sal | Average_Sal |
+-----------+-------------+------------+-------------+
|    281000 |       55000 |      25000 |  35125.0000 |
+-----------+-------------+------------+-------------+
1 row in set (0.00 sec)
```

Aggregate Functions in SQL (cont'd.)

Query 20. Find the sum of the salaries of all employees of the 'Research' department, as well as the maximum salary, the minimum salary, and the average salary in this department.

```
SELECT SUM(Salary),
MAX(Salary),
MIN(Salary), AVG(Salary)
FROM (EMPLOYEE join
department on
dno=dnumber) where
dname='research';
```



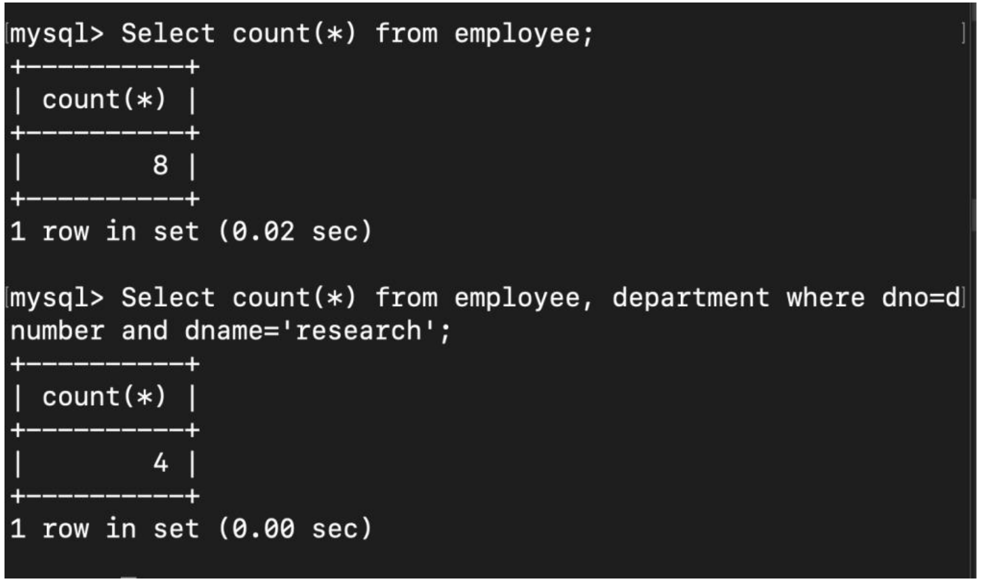
```
mysql> SELECT SUM(Salary), MAX(Salary), MIN(Salary), AVG(Sal
ary) FROM (EMPLOYEE join department on dno=dnumber) where dn
ame='research';
+-------------+-------------+-------------+-------------+
| SUM(Salary) | MAX(Salary) | MIN(Salary) | AVG(Salary) |
+-------------+-------------+-------------+-------------+
|      133000 |       40000 |       25000 |  33250.0000 |
+-------------+-------------+-------------+-------------+
1 row in set (0.00 sec)
```

Aggregate Functions in SQL (cont'd.)

Queries 21 and 22. Retrieve the total number of employees in the company (Q21) and the number of employees in the 'Research' department (Q22).

```
Select count(*) from
employee;

Select count(*) from
employee, department
where dno=dnumber
and dname='research';
```



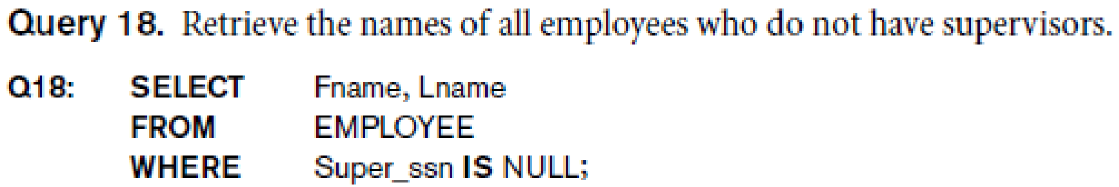
```
mysql> Select count(*) from employee;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.02 sec)

mysql> Select count(*) from employee, department where dno=d
number and dname='research';
+----------+
| count(*) |
+----------+
|        4 |
+----------+
1 row in set (0.00 sec)
```

Comparisons Involving NULL

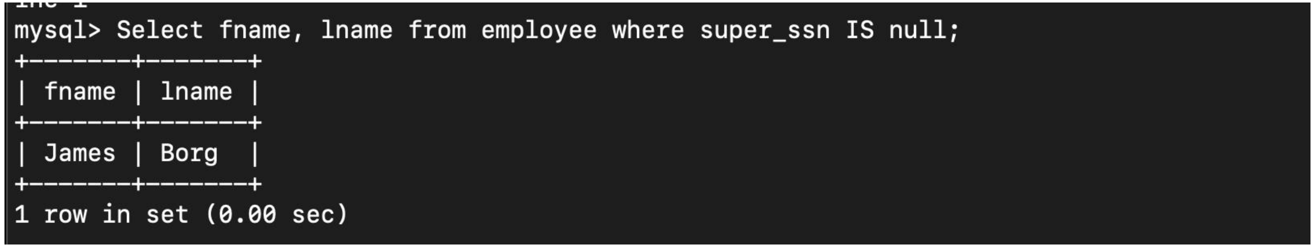
SQL allows queries that check whether an attribute value is `NULL`

`IS` or `IS NOT NULL`



Query 18. Retrieve the names of all employees who do not have supervisors.

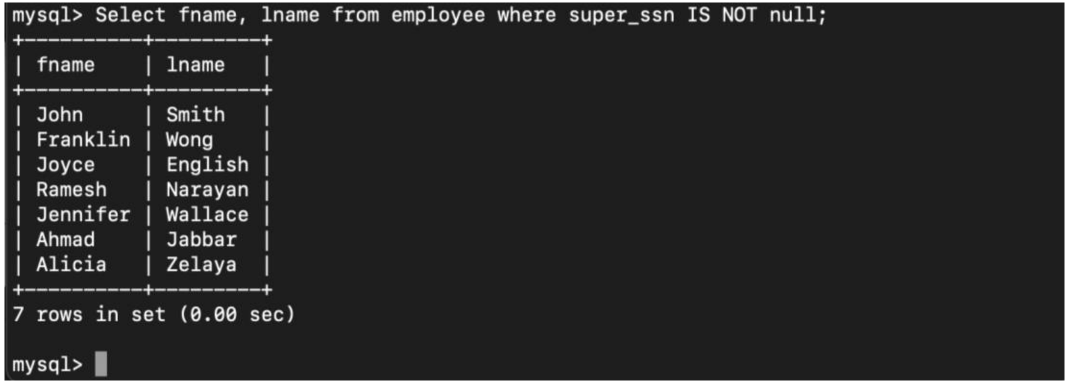
```
Q18:    SELECT    Fname, Lname
        FROM      EMPLOYEE
        WHERE     Super_ssn IS NULL;
```



```
mysql> Select fname, lname from employee where super_ssn IS null;
+-------+-------+
| fname | lname |
+-------+-------+
| James | Borg  |
+-------+-------+
1 row in set (0.00 sec)
```

IS & IS NOT

Select fname, lname from employee where super_ssn IS NOT null;



```
mysql> Select fname, lname from employee where super_ssn IS NOT null;
+----------+---------+
| fname    | lname   |
+----------+---------+
| John     | Smith   |
| Franklin | Wong    |
| Joyce    | English |
| Ramesh   | Narayan |
| Jennifer | Wallace |
| Ahmad    | Jabbar  |
| Alicia   | Zelaya  |
+----------+---------+
7 rows in set (0.00 sec)

mysql> 
```

Grouping: The GROUP BY Clause

Partition relation into subsets of tuples

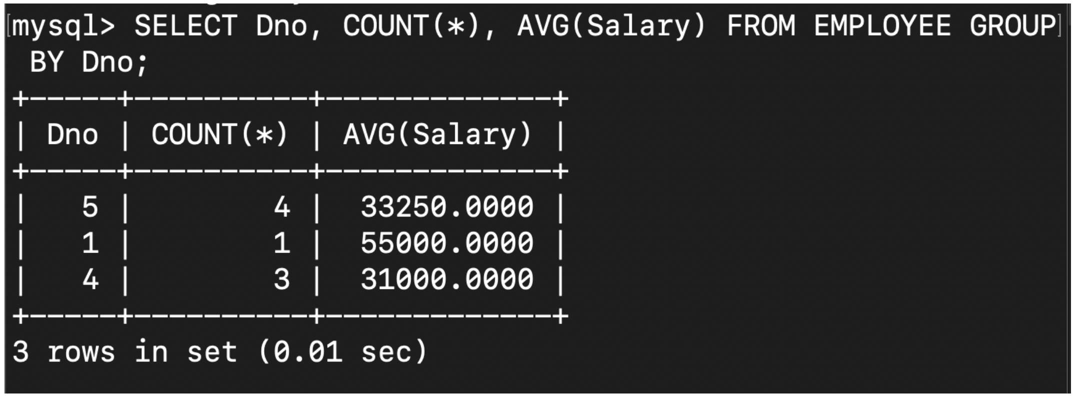
- Based on **grouping attribute(s)**
- Apply function to each such group independently

`GROUP BY` clause

- Specifies grouping attributes

Group BY example

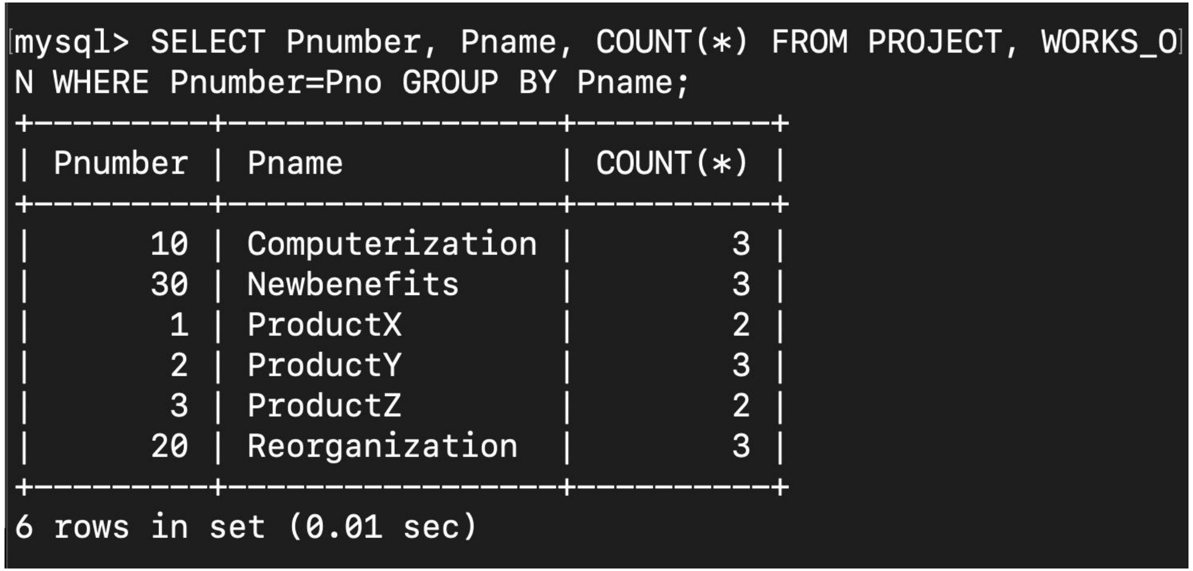
SELECT Dno, COUNT(*), AVG(Salary) FROM EMPLOYEE GROUP BY Dno;



```
mysql> SELECT Dno, COUNT(*), AVG(Salary) FROM EMPLOYEE GROUP
 BY Dno;
+-----+----------+-------------+
| Dno | COUNT(*) | AVG(Salary) |
+-----+----------+-------------+
|   5 |        4 |  33250.0000 |
|   1 |        1 |  55000.0000 |
|   4 |        3 |  31000.0000 |
+-----+----------+-------------+
3 rows in set (0.01 sec)
```

Group BY example

SELECT Pnumber, Pname, COUNT(*) FROM PROJECT, WORKS_ON WHERE Pnumber=Pno GROUP BY Pname;



```
mysql> SELECT Pnumber, Pname, COUNT(*) FROM PROJECT, WORKS_O
N WHERE Pnumber=Pno GROUP BY Pname;
+---------+-----------------+----------+
| Pnumber | Pname           | COUNT(*) |
+---------+-----------------+----------+
|      10 | Computerization |        3 |
|      30 | Newbenefits     |        3 |
|       1 | ProductX        |        2 |
|       2 | ProductY        |        3 |
|       3 | ProductZ        |        2 |
|      20 | Reorganization  |        3 |
+---------+-----------------+----------+
6 rows in set (0.01 sec)
```

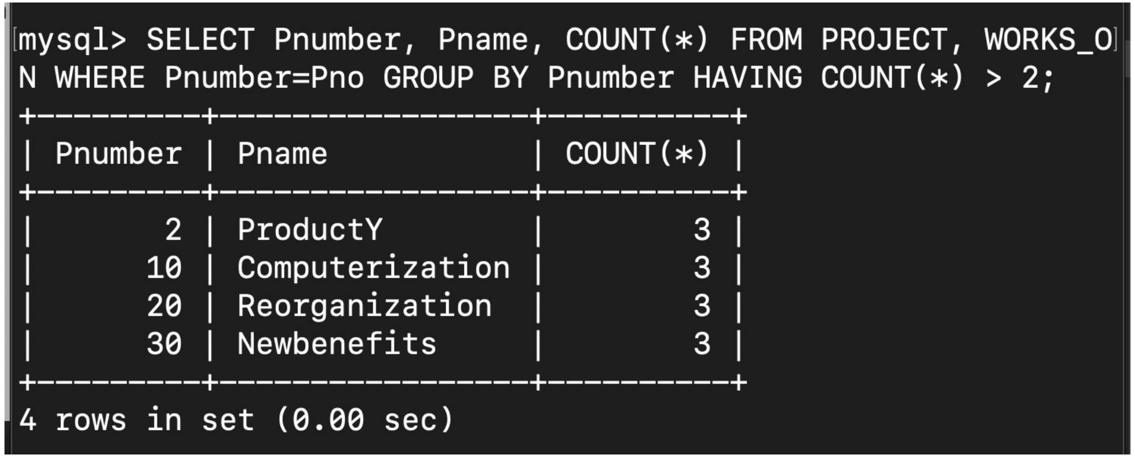
Grouping: The GROUP BY and HAVING Clauses (cont’d.)

HAVING clause

Provides a condition to select or reject an entire group:

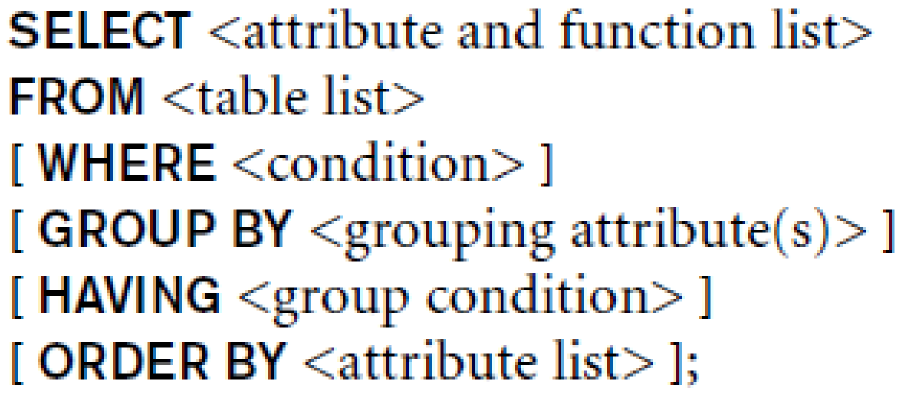
Query 26. For each project *on which more than two employees work,* retrieve the project number, the project name, and the number of employees who work on the project.

```
SELECT Pnumber,
Pname, COUNT(*) FROM
PROJECT, WORKS_ON
WHERE Pnumber=Pno
GROUP BY Pnumber
HAVING COUNT(*) > 2;
```



```
mysql> SELECT Pnumber, Pname, COUNT(*) FROM PROJECT, WORKS_O
N WHERE Pnumber=Pno GROUP BY Pnumber HAVING COUNT(*) > 2;
+---------+-----------------+----------+
| Pnumber | Pname           | COUNT(*) |
+---------+-----------------+----------+
|       2 | ProductY        |        3 |
|      10 | Computerization |        3 |
|      20 | Reorganization  |        3 |
|      30 | Newbenefits     |        3 |
+---------+-----------------+----------+
4 rows in set (0.00 sec)
```

EXPANDED Block Structure of SQL Queries



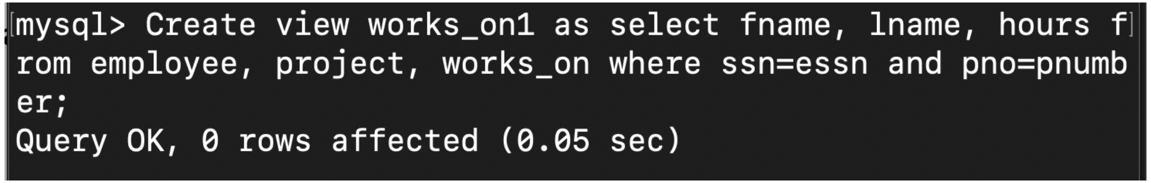
```
SELECT <attribute and function list>
FROM <table list>
[ WHERE <condition> ]
[ GROUP BY <grouping attribute(s)> ]
[ HAVING <group condition> ]
[ ORDER BY <attribute list> ];
```

Specification of Views in SQL

`CREATE VIEW` command

Give table name, list of attribute names, and a query to specify the contents of the view

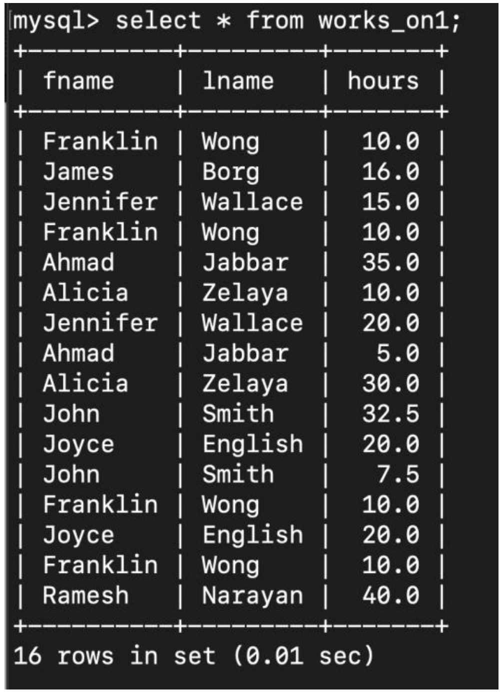
Create view works_on1 as select fname, lname, hours from employee, project, works_on where ssn=essn and pno=pnumber;



```
mysql> Create view works_on1 as select fname, lname, hours f
rom employee, project, works_on where ssn=essn and pno=pnumb
er;
Query OK, 0 rows affected (0.05 sec)
```

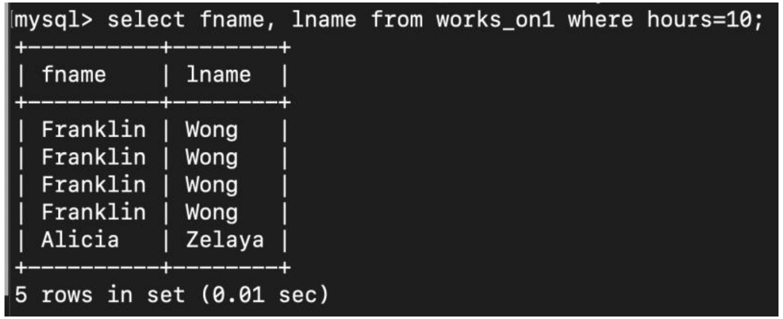
Data in view, query view

select * from works_on1;



```
mysql> select * from works_on1;
+----------+---------+-------+
| fname    | lname   | hours |
+----------+---------+-------+
| Franklin | Wong    |  10.0 |
| James    | Borg    |  16.0 |
| Jennifer | Wallace |  15.0 |
| Franklin | Wong    |  10.0 |
| Ahmad    | Jabbar  |  35.0 |
| Alicia   | Zelaya  |  10.0 |
| Jennifer | Wallace |  20.0 |
| Ahmad    | Jabbar  |   5.0 |
| Alicia   | Zelaya  |  30.0 |
| John     | Smith   |  32.5 |
| Joyce    | English |  20.0 |
| John     | Smith   |   7.5 |
| Franklin | Wong    |  10.0 |
| Joyce    | English |  20.0 |
| Franklin | Wong    |  10.0 |
| Ramesh   | Narayan |  40.0 |
+----------+---------+-------+
16 rows in set (0.01 sec)
```

select fname, lname from works_on1 where hours=10;



```
mysql> select fname, lname from works_on1 where hours=10;
+----------+--------+
| fname    | lname  |
+----------+--------+
| Franklin | Wong   |
| Franklin | Wong   |
| Franklin | Wong   |
| Franklin | Wong   |
| Alicia   | Zelaya |
+----------+--------+
5 rows in set (0.01 sec)
```

This Lecture

The ALTER table command

Alter table actions include:

- Adding or dropping a column (attribute)
- Changing a column definition
- Adding or dropping table constraints

Example:

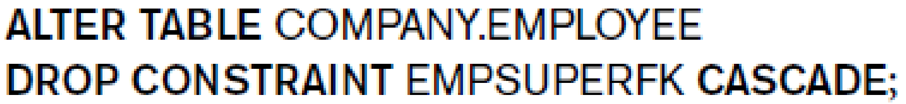
```
ALTER TABLE COMPANY.EMPLOYEE ADD COLUMN Job
VARCHAR(12);
```

- Keeping track of jobs of employees

Adding and Dropping Constraints

Change constraints specified on a table

- Add or drop a named constraint



```
ALTER TABLE COMPANY.EMPLOYEE
DROP CONSTRAINT EMPSUPERFK CASCADE;
```

To be dropped, a constraint must have been given a name when it is specified

Dropping Columns, Default Values

To drop a column

- Choose either `CASCADE` or `RESTRICT`
- `CASCADE` would drop the column from views etc. `RESTRICT` is possible if no views refer to it.

  ALTER TABLE COMPANY.EMPLOYEE **DROP COLUMN** Address **CASCADE**;

  removes the attribute Address from the employee base table

Default values can be dropped and altered :

ALTER TABLE COMPANY.DEPARTMENT **ALTER COLUMN** Mgr_ssn **DROP DEFAULT**;

ALTER TABLE COMPANY.DEPARTMENT **ALTER COLUMN** Mgr_ssn **SET DEFAULT** ‘333445555’;

The EXISTS Functions in SQL for correlating queries

`EXISTS` function

- Check whether the result of a correlated nested query is empty or not. They are Boolean functions that return a TRUE or FALSE result.

`EXISTS` and `NOT EXISTS`

- Typically used in conjunction with a correlated nested query

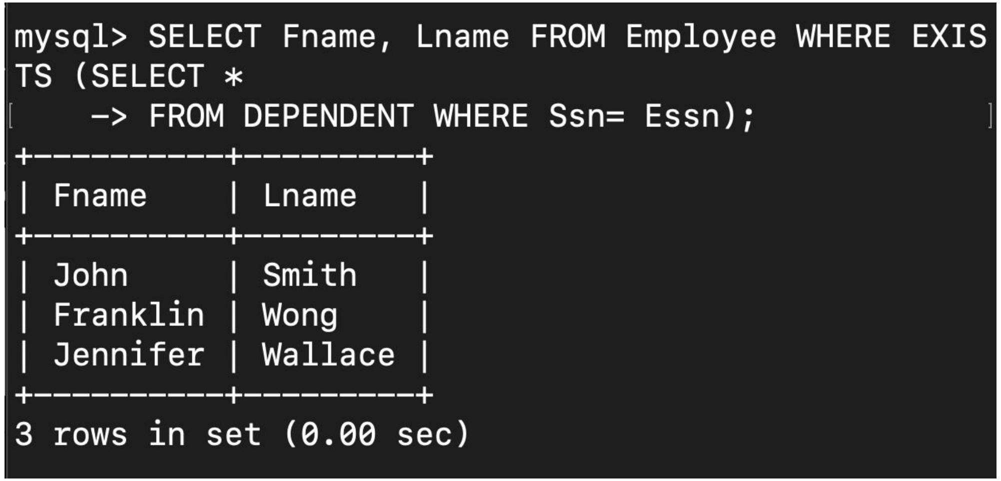
USE of EXISTS

```
SELECT Fname, Lname FROM
Employee WHERE EXISTS
(SELECT * FROM DEPENDENT
WHERE Ssn= Essn);
```

Try it yourself!

USE of EXISTS

```
SELECT Fname, Lname FROM
Employee WHERE EXISTS
(SELECT * FROM DEPENDENT
WHERE Ssn= Essn);
```



```
mysql> SELECT Fname, Lname FROM Employee WHERE EXIS
TS (SELECT *
    -> FROM DEPENDENT WHERE Ssn= Essn);
+----------+---------+
| Fname    | Lname   |
+----------+---------+
| John     | Smith   |
| Franklin | Wong    |
| Jennifer | Wallace |
+----------+---------+
3 rows in set (0.00 sec)
```

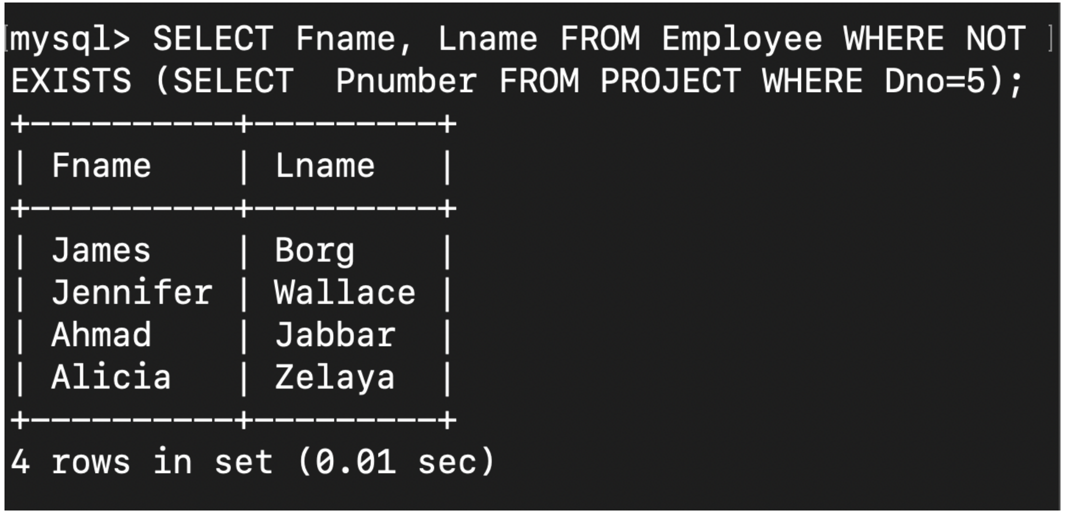
USE OF NOT EXISTS

```
SELECT Fname, Lname FROM
Employee WHERE NOT EXISTS
(SELECT  Pnumber FROM PROJECT
WHERE Dno=5);
```

Try it yourself!

USE OF NOT EXISTS

SELECT Fname, Lname FROM Employee WHERE NOT EXISTS (SELECT Pnumber FROM PROJECT WHERE Dno=5);



```
mysql> SELECT Fname, Lname FROM Employee WHERE NOT
EXISTS (SELECT  Pnumber FROM PROJECT WHERE Dno=5);
+----------+---------+
| Fname    | Lname   |
+----------+---------+
| James    | Borg    |
| Jennifer | Wallace |
| Ahmad    | Jabbar  |
| Alicia   | Zelaya  |
+----------+---------+
4 rows in set (0.01 sec)
```

Specifying Joined Tables in the FROM Clause of SQL

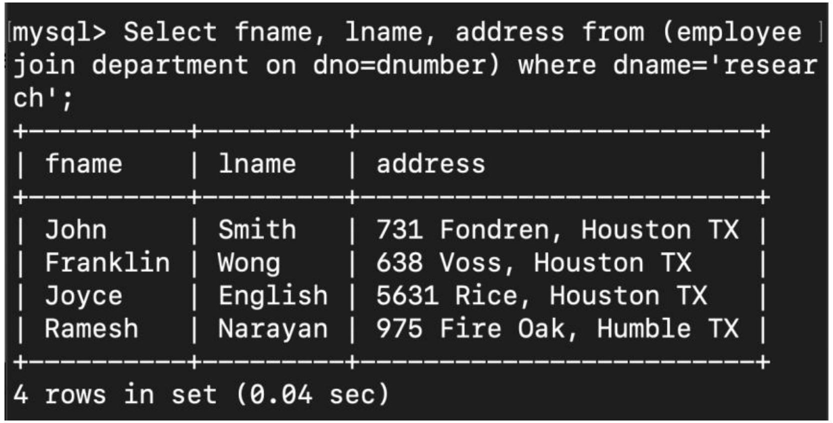
Joined table

Permits users to specify a table resulting from a join operation in the FROM clause of a query

The FROM clause in Q1A

Contains a single joined table. JOIN may also be called INNER JOIN

```
Select fname, lname, address
from (employee join department
on dno=dnumber) where
dname='research';
```



```
mysql> Select fname, lname, address from (employee 
join department on dno=dnumber) where dname='resear
ch';
+----------+---------+------------------------+
| fname    | lname   | address                |
+----------+---------+------------------------+
| John     | Smith   | 731 Fondren, Houston TX |
| Franklin | Wong    | 638 Voss, Houston TX   |
| Joyce    | English | 5631 Rice, Houston TX  |
| Ramesh   | Narayan | 975 Fire Oak, Humble TX |
+----------+---------+------------------------+
4 rows in set (0.04 sec)
```

Specifying Joined Tables in the FROM Clause of SQL

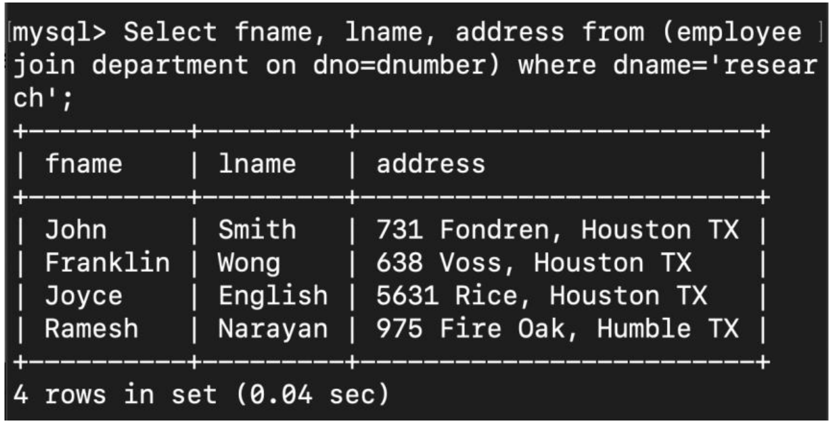
Joined table

Permits users to specify a table resulting from a join operation in the FROM clause of a query

The FROM clause in Q1A

Contains a single joined table. JOIN may also be called INNER JOIN

```
Select fname, lname, address
from (employee join department
on dno=dnumber) where
dname='research';
```

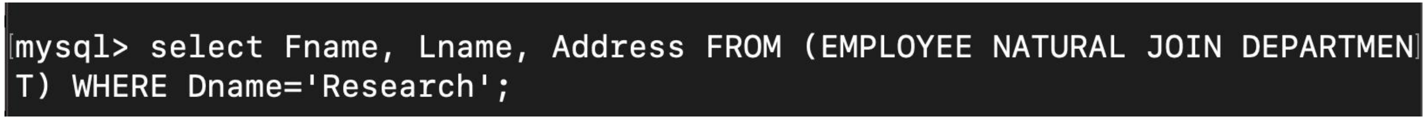


```
mysql> Select fname, lname, address from (employee
join department on dno=dnumber) where dname='resear
ch';
+----------+---------+------------------------+
| fname    | lname   | address                |
+----------+---------+------------------------+
| John     | Smith   | 731 Fondren, Houston TX |
| Franklin | Wong    | 638 Voss, Houston TX   |
| Joyce    | English | 5631 Rice, Houston TX  |
| Ramesh   | Narayan | 975 Fire Oak, Humble TX |
+----------+---------+------------------------+
4 rows in set (0.04 sec)
```

Different Types of JOINed Tables in SQL

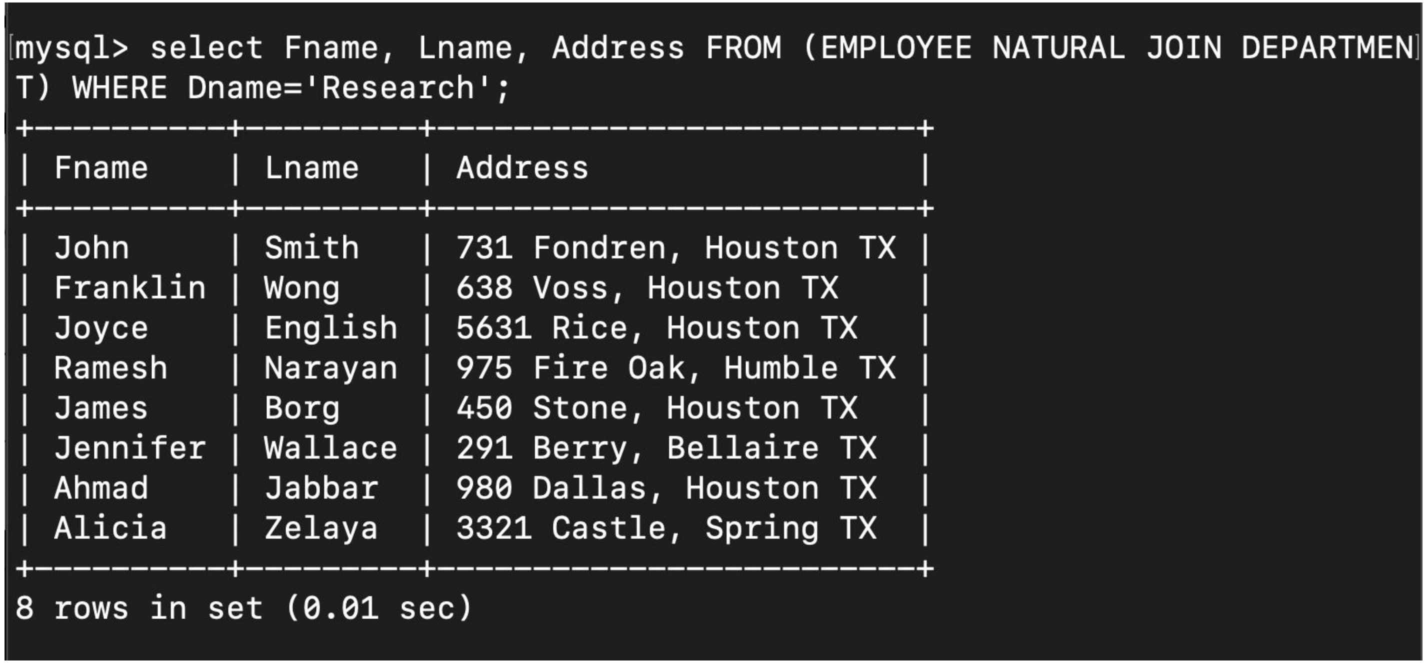
- Specify different types of join
  - NATURAL JOIN
  - Various types of OUTER JOIN (LEFT, RIGHT, FULL)
- NATURAL JOIN on two relations R and S
  - No join condition specified
  - Is equivalent to an implicit EQUIJOIN condition for each pair of attributes with same name from R and S
  - The associated tables have one or more pairs of identically named columns
  - The columns must be the same data type
  - No need for ON

NATURAL JOIN



```
mysql> select Fname, Lname, Address FROM (EMPLOYEE NATURAL JOIN DEPARTMEN
T) WHERE Dname='Research';
```

NATURAL JOIN



```
mysql> select Fname, Lname, Address FROM (EMPLOYEE NATURAL JOIN DEPARTMEN
T) WHERE Dname='Research';
+----------+---------+-------------------------+
| Fname    | Lname   | Address                 |
+----------+---------+-------------------------+
| John     | Smith   | 731 Fondren, Houston TX |
| Franklin | Wong    | 638 Voss, Houston TX    |
| Joyce    | English | 5631 Rice, Houston TX   |
| Ramesh   | Narayan | 975 Fire Oak, Humble TX |
| James    | Borg    | 450 Stone, Houston TX   |
| Jennifer | Wallace | 291 Berry, Bellaire TX  |
| Ahmad    | Jabbar  | 980 Dallas, Houston TX  |
| Alicia   | Zelaya  | 3321 Castle, Spring TX  |
+----------+---------+-------------------------+
8 rows in set (0.01 sec)
```

INNER and OUTER Joins

INNER JOIN **(versus** OUTER JOIN**)**

- Default type of join in a joined table
- Tuple is included in the result only if a matching tuple exists in the other relation

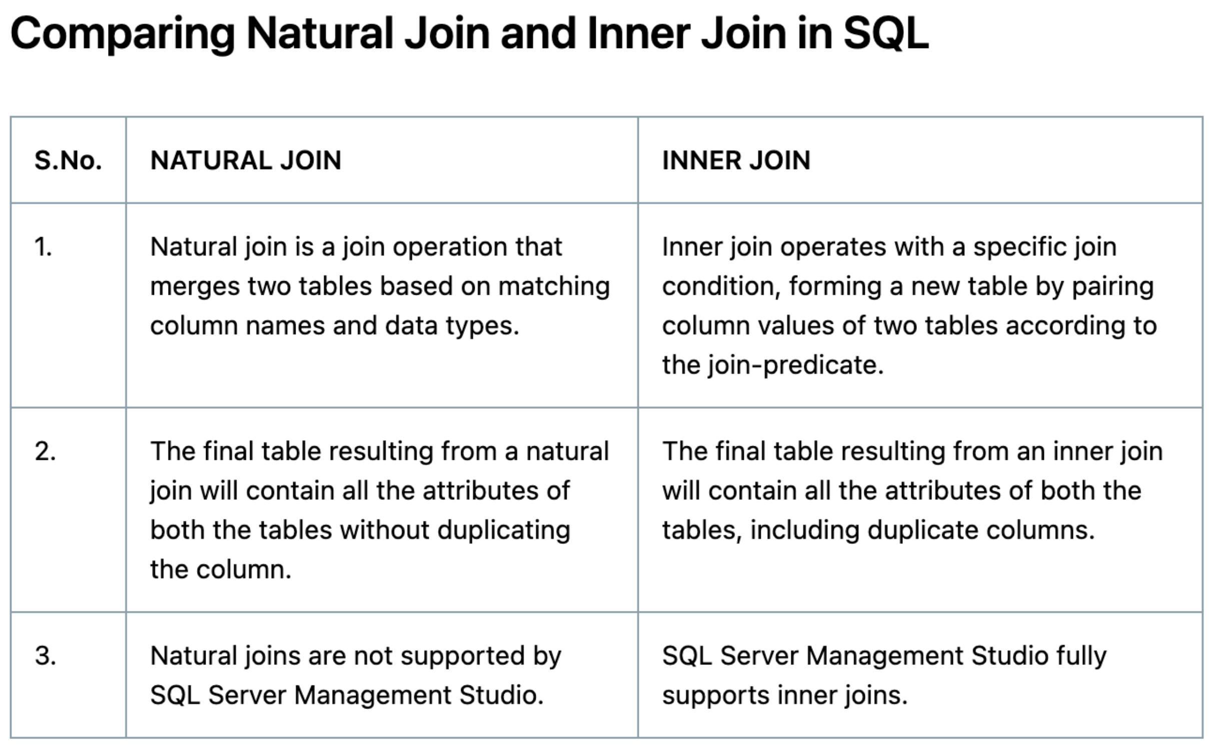
LEFT OUTER JOIN

- Every tuple in left table must appear in result
- If no matching tuple
  - Padded with NULL values for attributes of right table

RIGHT OUTER JOIN

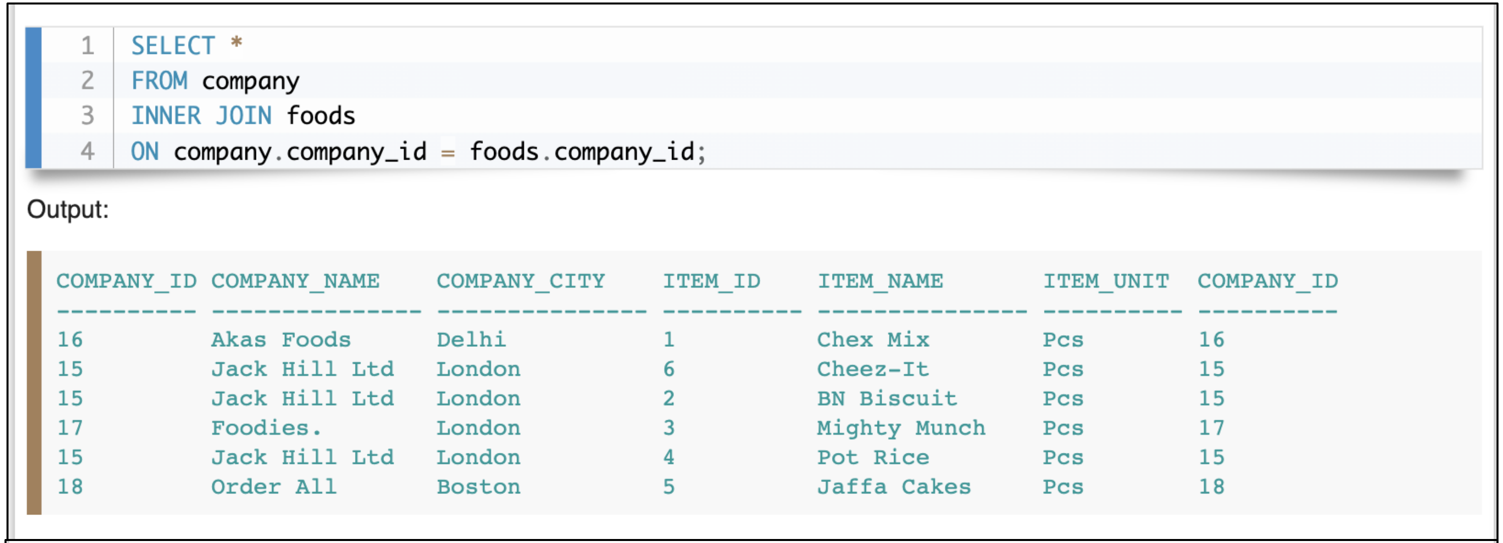
- Every tuple in right table must appear in result
- If no matching tuple
  - Padded with NULL values for attributes of left table

Natural join & Inner join difference



Comparing Natural Join and Inner Join in SQL

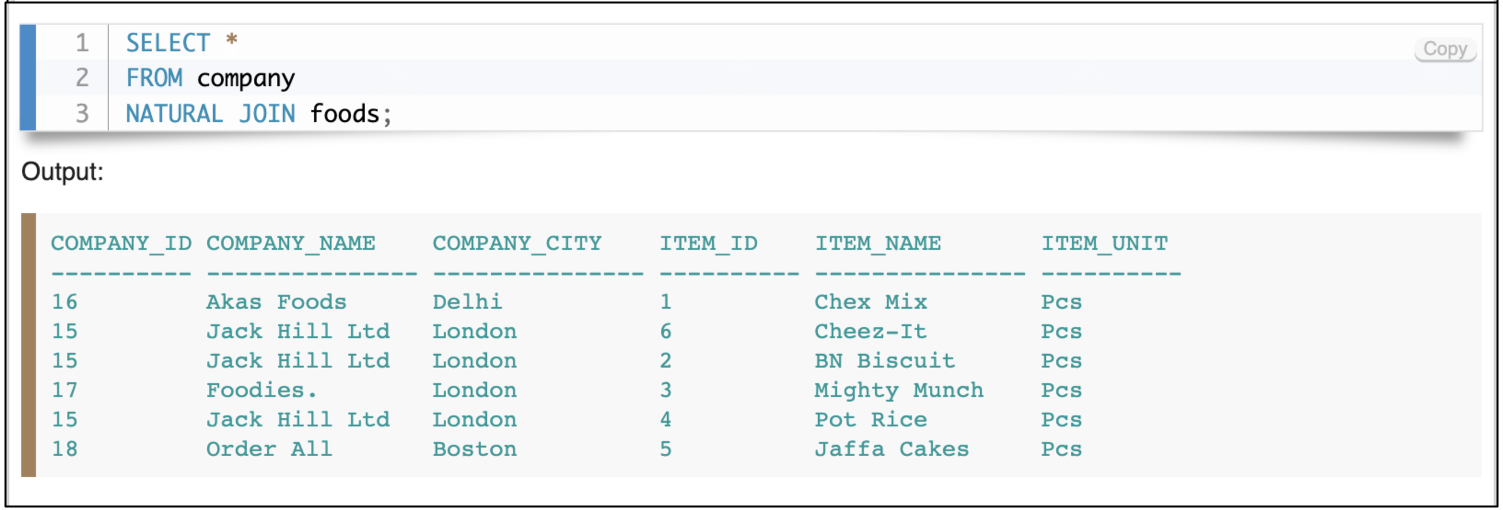
S.No.	NATURAL JOIN	INNER JOIN
1.	Natural join is a join operation that merges two tables based on matching column names and data types.	Inner join operates with a specific join condition, forming a new table by pairing column values of two tables according to the join-predicate.
2.	The final table resulting from a natural join will contain all the attributes of both the tables without duplicating the column.	The final table resulting from an inner join will contain all the attributes of both the tables, including duplicate columns.
3.	Natural joins are not supported by SQL Server Management Studio.	SQL Server Management Studio fully supports inner joins.



```sql
SELECT *
FROM company
INNER JOIN foods
ON company.company_id = foods.company_id;
```

Output:

```
COMPANY_ID COMPANY_NAME    COMPANY_CITY    ITEM_ID    ITEM_NAME       ITEM_UNIT  COMPANY_ID
---------- --------------- --------------- ---------- --------------- ---------- ----------
16         Akas Foods      Delhi           1          Chex Mix        Pcs        16
15         Jack Hill Ltd   London          6          Cheez-It        Pcs        15
15         Jack Hill Ltd   London          2          BN Biscuit      Pcs        15
17         Foodies.        London          3          Mighty Munch    Pcs        17
15         Jack Hill Ltd   London          4          Pot Rice        Pcs        15
18         Order All       Boston          5          Jaffa Cakes     Pcs        18
```

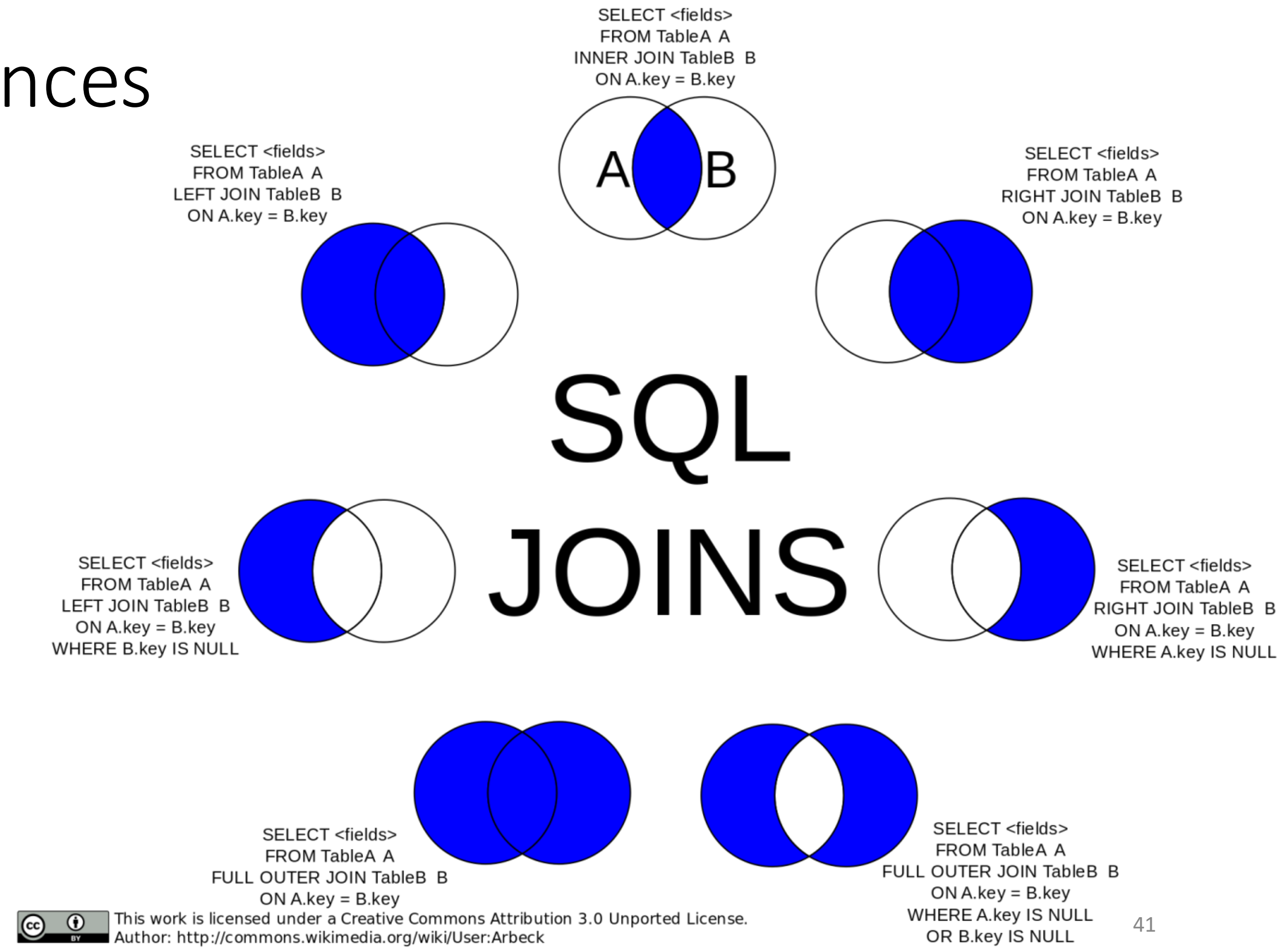


```sql
SELECT *
FROM company
NATURAL JOIN foods;
```

Output:

```
COMPANY_ID COMPANY_NAME    COMPANY_CITY    ITEM_ID    ITEM_NAME       ITEM_UNIT
---------- --------------- --------------- ---------- --------------- ----------
16         Akas Foods      Delhi           1          Chex Mix        Pcs
15         Jack Hill Ltd   London          6          Cheez-It        Pcs
15         Jack Hill Ltd   London          2          BN Biscuit      Pcs
17         Foodies.        London          3          Mighty Munch    Pcs
15         Jack Hill Ltd   London          4          Pot Rice        Pcs
18         Order All       Boston          5          Jaffa Cakes     Pcs
```

Joins differences



```
SELECT <fields>
FROM TableA A
INNER JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
```

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

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

```
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
```

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

https://i.stack.imgur.com/3bs7C.png

Bibliography / Acknowledgements

Instructor materials from Elmasri & Navathe 7e

pk.profgiri

Ponnurangam.kumaraguru

/in/ponguru

ponguru

pk.guru@iiit.ac.in

Thank you
for attending
the class!!!