Database System

北京交通大学软件学院

王方石 教授

E-mail: fshwang@bjtu.edu.cn

Chapter 3-Contents

3.1 Introduction to SQL

3.2 Data Definition Statements

3.3 Data Query Statements

3.4 Data Modification Statements

3.5 Views

3.6 Programmatic SQL

Chapter 3-2 - Objectives

How to retrieve data from database using SELECT and:

- Use compound WHERE conditions.
- Sort query results using ORDER BY.
- Use aggregate functions.
- Group data using GROUP BY and HAVING.
- Use subqueries.
- Join tables together.
- Perform set operations (UNION, INTERSECT, EXCEPT).

3.3 Data Query Statements

3.3.1 Select statement

- SELECT [DISTINCT | ALL]
- {* | [columnExpression [AS newName]] [,...] }
- FROM TableName [alias] [, ...]
- [WHERE condition]
- [GROUP BY columnList]
- [HAVING condition]
- [ORDER BY columnList]

SELECT Statement

FROM	Specifies table(s) to be used.
WHERE	Filters rows.
GROUP BY	Forms groups of rows with same column value.
HAVING	Filters groups subject to some condition.
SELECT	Specifies which columns are to appear in output.
ORDER BY	Specifies the order of the output.

SELECT Statement

- Order of the clauses cannot be changed.

- Only SELECT and FROM are mandatory.

Case

Estates Agency-Dream of Home

- **Branch** (branchNo, street, city, postcode)
- **Staff** (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
- **Property** (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)
- **Client** (clientNo, fName, lName, telNo, prefType, maxRent)
- **PrivateOwner** (ownerNo, fName, lName, address, telNo)
- **Viewing** (clientNo, propertyNo, viewDate, comment)

Example 3.8 All Columns, All Rows

List full details of all staff.

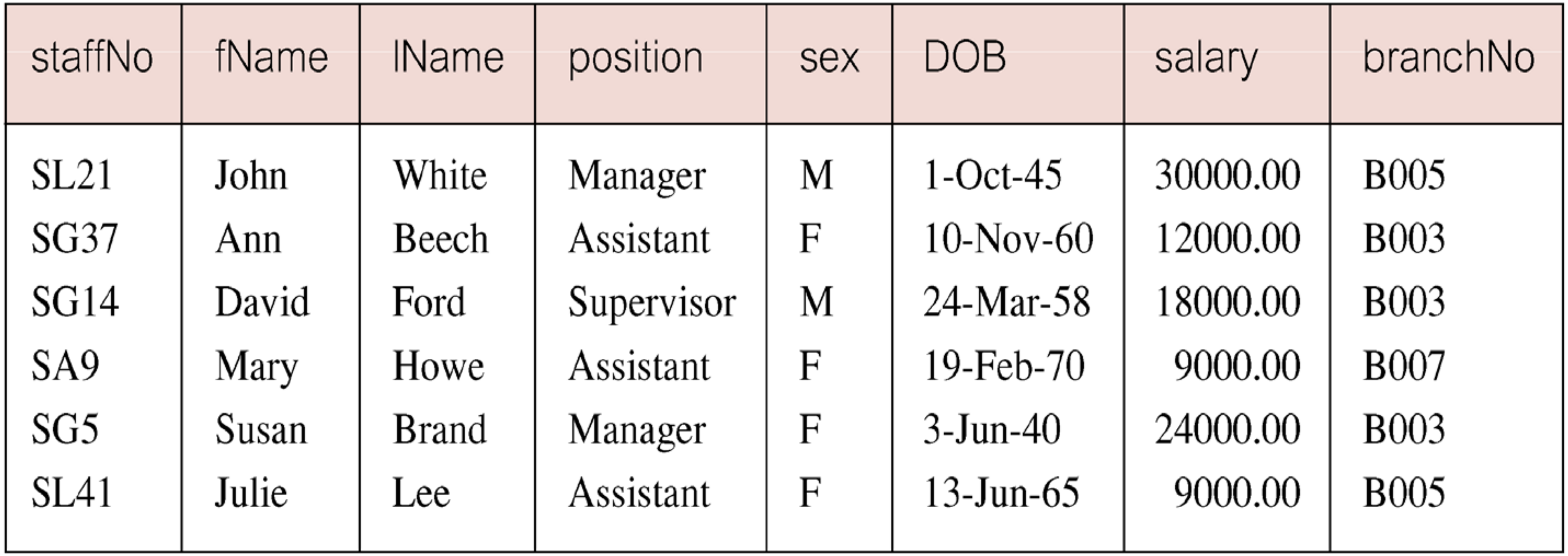
```
SELECT staffNo, fName, lName, position,
       sex, DOB, salary, branchNo
FROM Staff;
```

- Can use * as an abbreviation for ‘all columns’:

```
SELECT *
FROM Staff;
```

Example All Columns, All Rows

Table 5.1 Result table for Example 5.1.



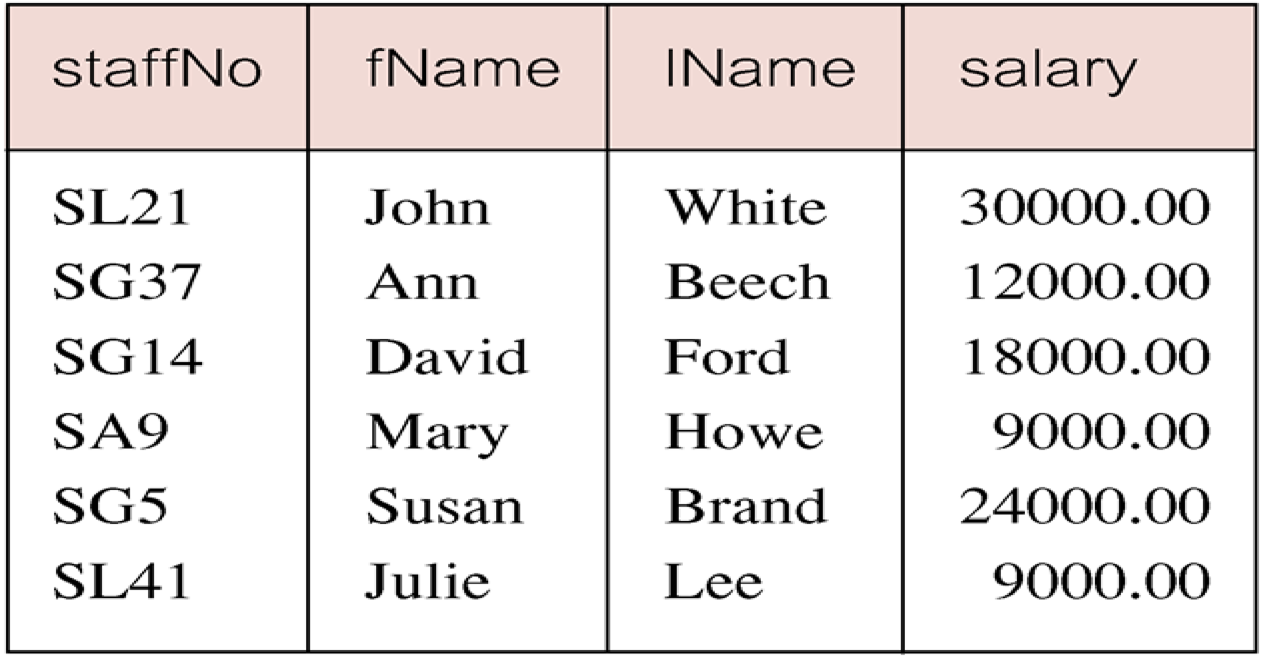
staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

Example 3.9 Specific Columns, All Rows
检索指定的列或属性

Produce a list of **salaries** for all staff, showing only the **staff number**, **first** and **last names**, and **salary**.

```
SELECT staffNo, fName, lName, salary
FROM Staff;
```

Result table



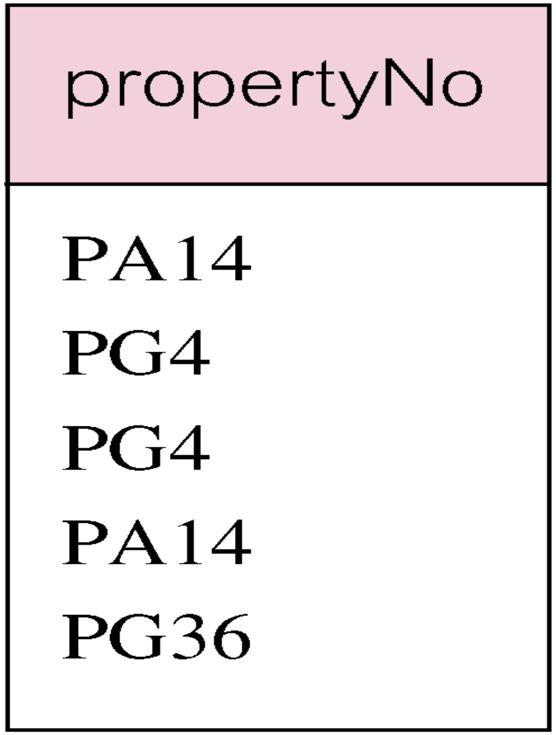
staffNo	fName	lName	salary
SL21	John	White	30000.00
SG37	Ann	Beech	12000.00
SG14	David	Ford	18000.00
SA9	Mary	Howe	9000.00
SG5	Susan	Brand	24000.00
SL41	Julie	Lee	9000.00

Example 3.10 Not Use of DISTINCT

It can be used to eliminate the duplicates.

e.g. List the property numbers of all properties that have been viewed.

```
SELECT propertyNo
FROM Viewing;
```



propertyNo
PA14
PG4
PG4
PA14
PG36

Example 3.11 Use of DISTINCT

- Use DISTINCT to eliminate duplicates:

```
SELECT DISTINCT propertyNo
FROM Viewing;
```

propertyNo
PA14
PG4
PG36

若select 后面有多个列：
select **distinct sno, cno** from sc
= distinct (sno,cno)

select **cno, distinct sno** from sc ？

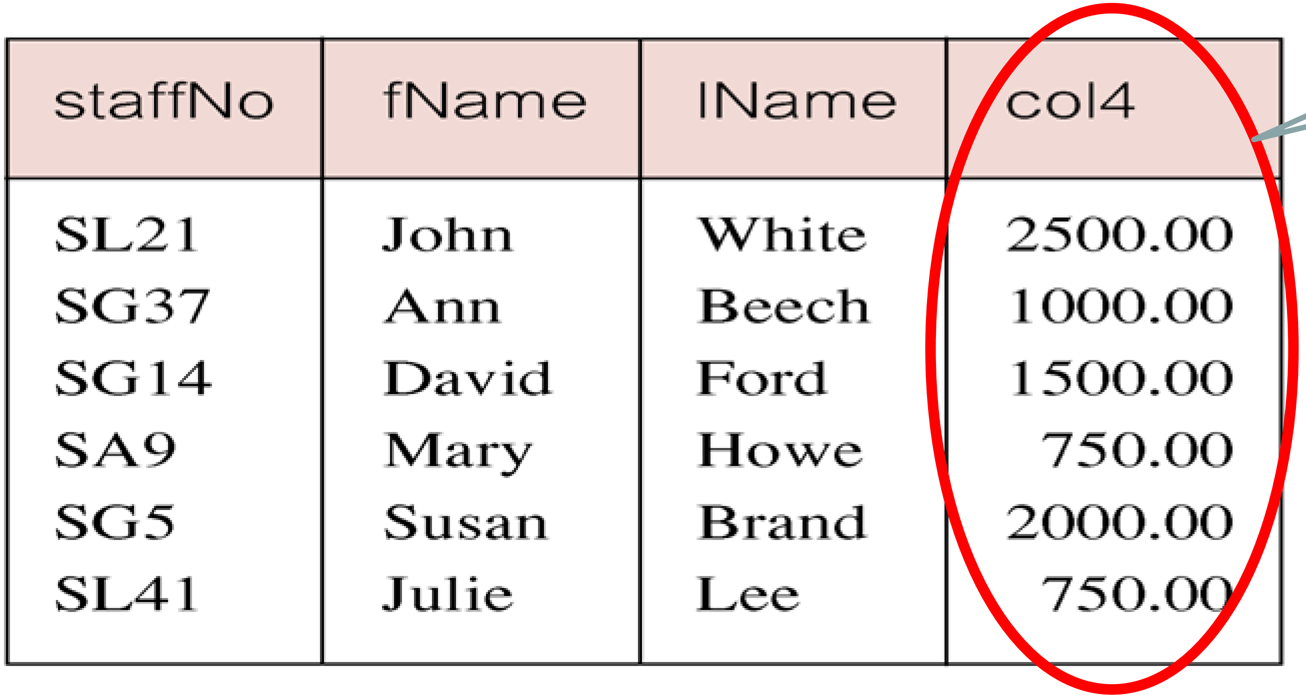
关键字 'distinct' 附近有语法错误。

Example 3.12 Calculated Fields

Produce a list of monthly salaries for all staff, showing staff number, first and last names, and salary details.

```
SELECT staffNo, fName, lName, salary/12
FROM Staff;
```

+, -, *, /, aggregate functions can be used in SELECT clause.



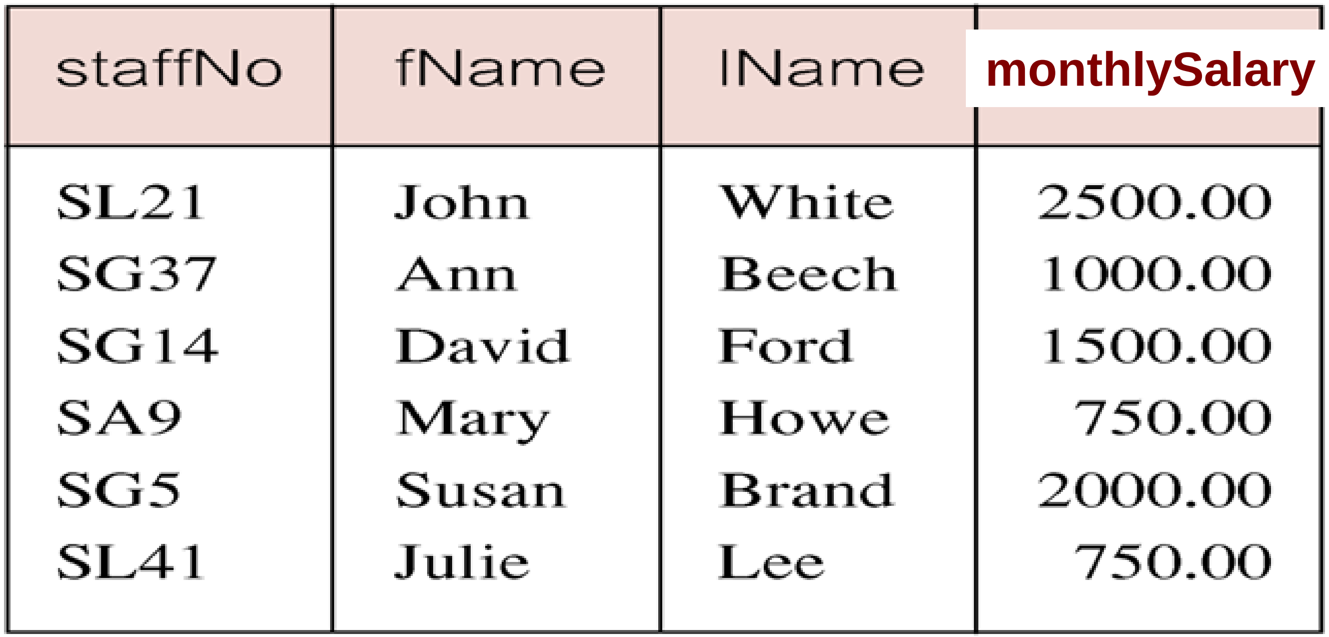
staffNo	fName	lName	col4
SL21	John	White	2500.00
SG37	Ann	Beech	1000.00
SG14	David	Ford	1500.00
SA9	Mary	Howe	750.00
SG5	Susan	Brand	2000.00
SL41	Julie	Lee	750.00

“虚”列

Example 3.12 Calculated Fields

- To name column, use AS clause:

```
SELECT staffNo, fName, lName,
            salary/12  AS monthlySalary
FROM Staff;
```



staffNo	fName	lName	monthlySalary
SL21	John	White	2500.00
SG37	Ann	Beech	1000.00
SG14	David	Ford	1500.00
SA9	Mary	Howe	750.00
SG5	Susan	Brand	2000.00
SL41	Julie	Lee	750.00

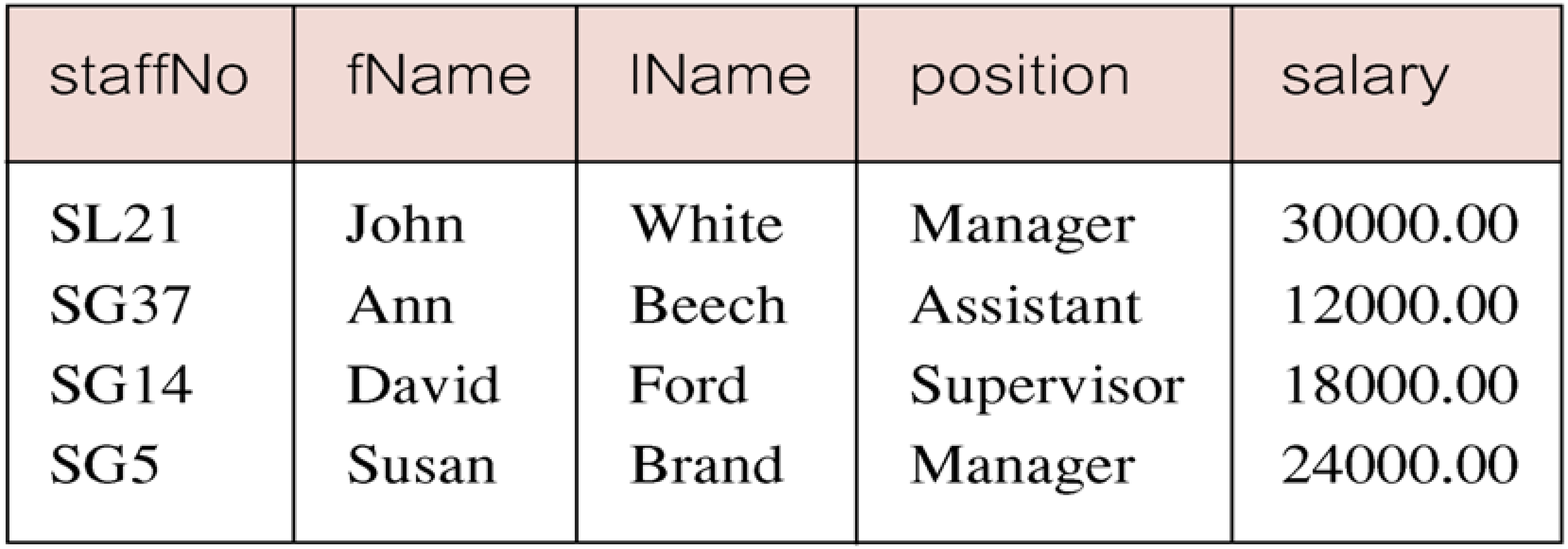
Row selection (WHERE clause)

	Predicates(谓词)	Function （功能）	Notes（说明）
1	=,<>(!=),<,<=,>,>=	Comparison search	
	AND, OR,NOT	Compound Comparison search （复合比较查询）	From left to right, NOT>AND>OR () has the highest priority
2	BETWEEN…AND; NOT BETWEEN…AND	Range test （范围查找）	includes the endpoints; Not include the endpoints
3	IN, NOT IN	Set membership	
4	LIKE, NOT LIKE	Pattern match	
5	NULL, IS NULL	NULL test	IS cannot be replaced by ‘=‘

Example 3.13 Comparison Search Condition

List all staff with a salary greater than 10,000.

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary > 10000;
```



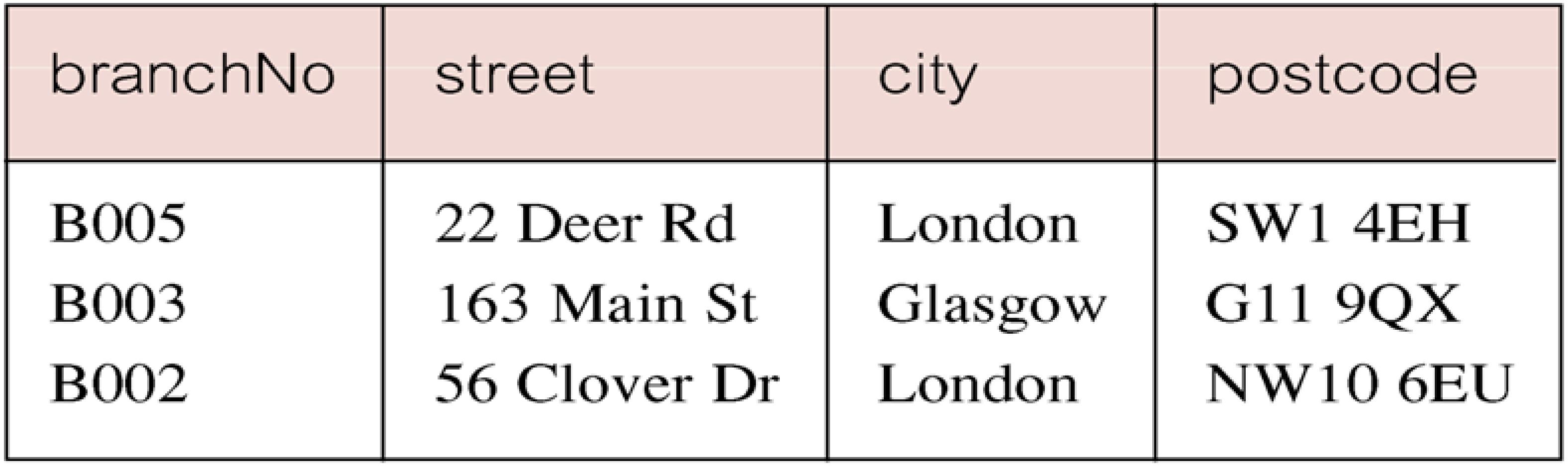
staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG37	Ann	Beech	Assistant	12000.00
SG14	David	Ford	Supervisor	18000.00
SG5	Susan	Brand	Manager	24000.00

Example 3.13

Compound Comparison Search Condition

List addresses of all branch offices in London or Glasgow.

```
SELECT *
FROM Branch
WHERE city = ‘London’ OR city = ‘Glasgow’;
```



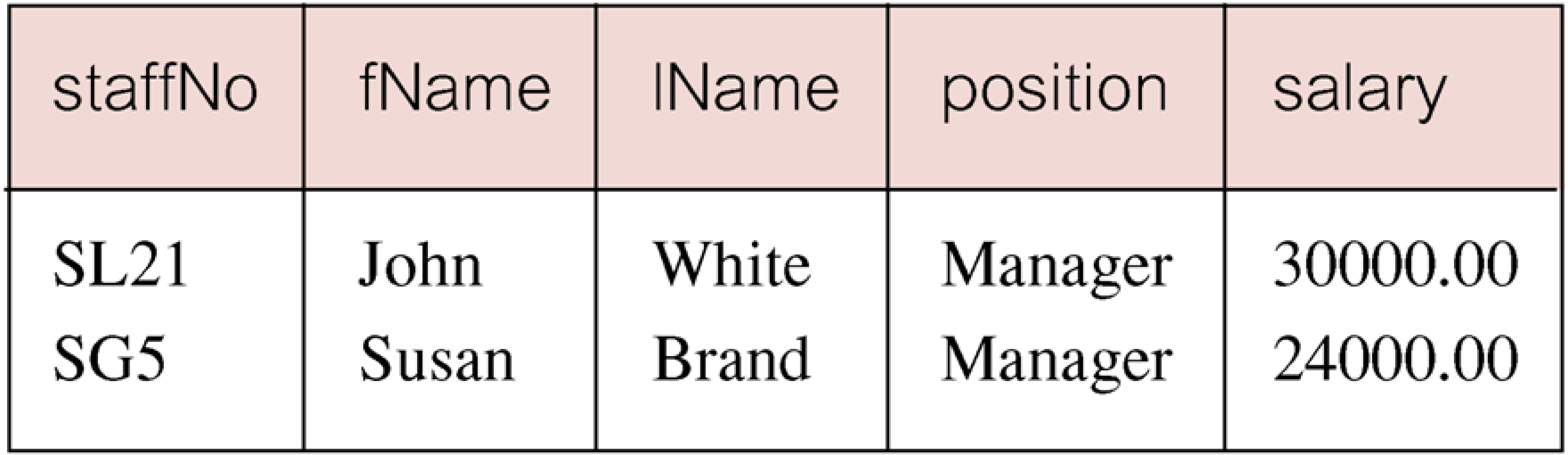
branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B003	163 Main St	Glasgow	G11 9QX
B002	56 Clover Dr	London	NW10 6EU

Example 3.14 Range Search Condition

List all staff with a salary between 20,000 and 30,000.

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary BETWEEN 20000 AND 30000;
```

- **BETWEEN test includes the endpoints of range.**



staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG5	Susan	Brand	Manager	24000.00

Example 3.14 Range Search Condition

- **BETWEEN does not add much to SQL’s expressive power. Could also write:**

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary>=20000 AND salary <= 30000;
```

- **Useful, though, for a range of values.**

Example 3.14 Range Search Condition

- Also a negated version NOT BETWEEN.

```
SELECT staffNo, fName, lName, position, salary
 FROM Staff
 WHERE salary NOT BETWEEN 20000 AND  30000;
```

- NOT BETWEEN also does not add much to SQL's expressive power. Could also write:

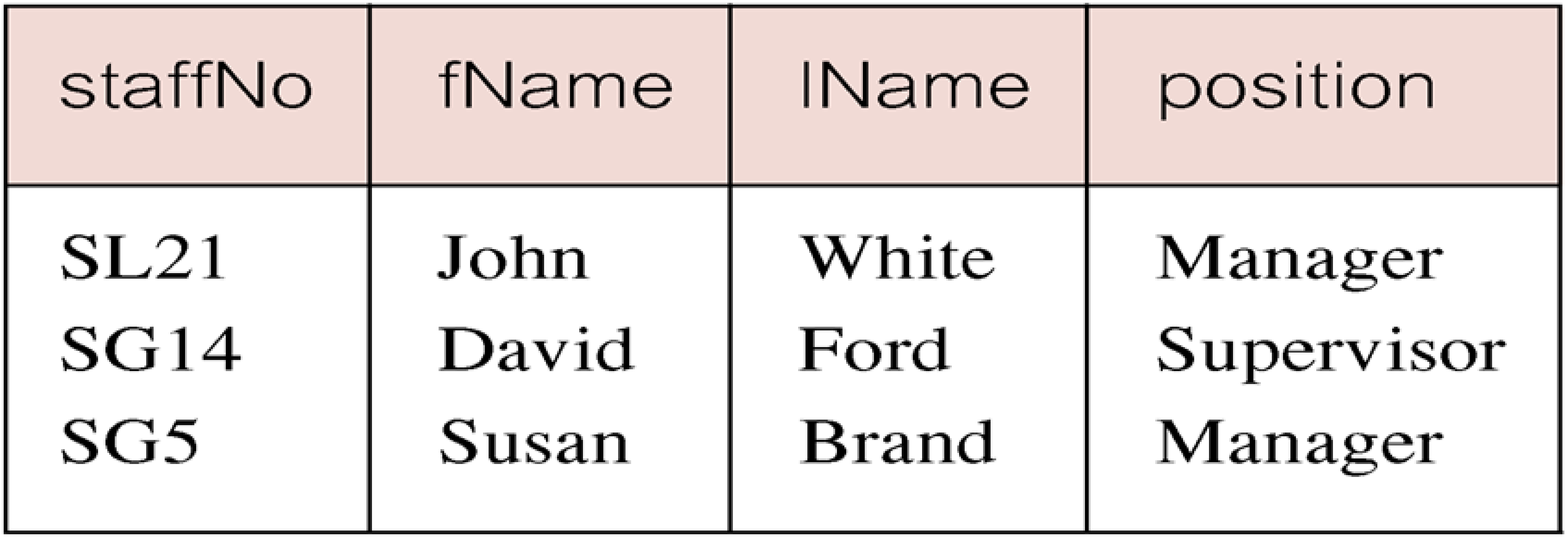
```
 SELECT staffNo, fName, lName, position, salary
 FROM Staff
 WHERE salary<20000 OR salary > 30000;
```

- NOT BETWEEN test does not include the endpoints of range.

Example 3.15 Set Membership

List all managers and supervisors.

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE position IN (‘Manager’, ‘Supervisor’);
```



staffNo	fName	lName	position
SL21	John	White	Manager
SG14	David	Ford	Supervisor
SG5	Susan	Brand	Manager

Example 3.15 Set Membership

- IN does not add much to SQL's expressive power.
- Could have expressed this as:

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE position='Manager' OR
              position='Supervisor';
```

- IN is more efficient when a set contains many values.

Example 3.15 Set Membership

- There is a negated version (NOT IN).

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE position NOT IN
          (‘Manager’, ‘Supervisor’);
```

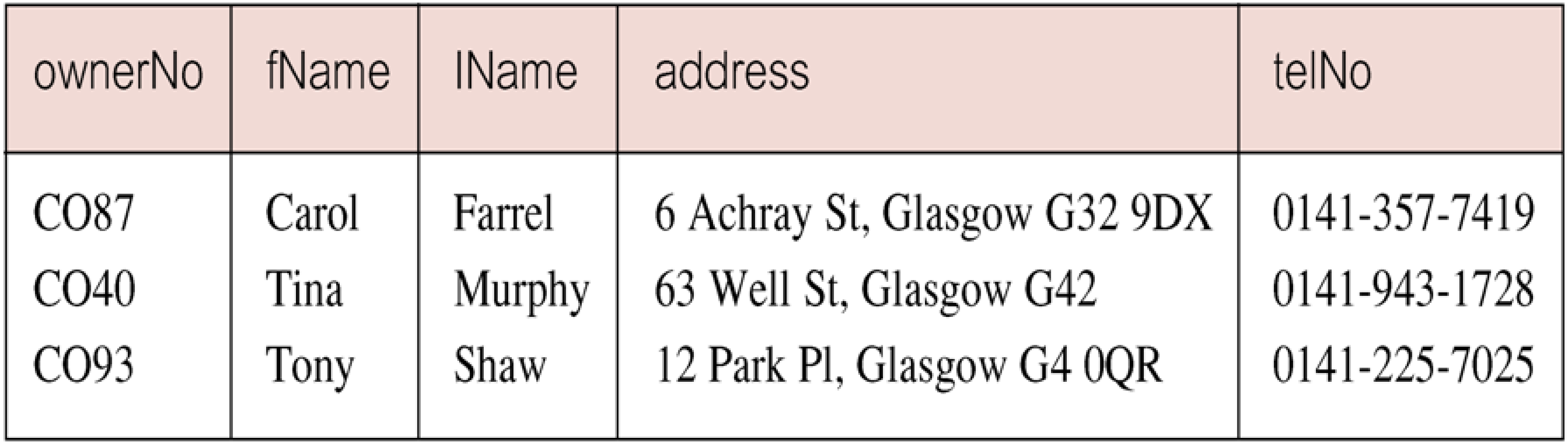
- NOT IN also does not add much to SQL’s expressive power. Could have expressed this as:

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE position<>‘Manager’ AND
          position<>‘Supervisor’;
```

Example 3.16 Pattern Matching

Find all owners with the string ‘Glasgow’ in their address.

```
SELECT ownerNo, fName, lName, address, telNo
FROM PrivateOwner
WHERE address LIKE ‘%Glasgow%’;
```



ownerNo	fName	lName	address	telNo
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025

Example 3.16 Pattern Matching

- SQL has two special pattern matching symbols:
  - %: sequence of zero or more characters;
  - _ (underscore): any single (0 or 1) character.
- LIKE ‘%Glasgow%’ means a sequence of characters of any length containing ‘*Glasgow*’.

Example 3.16 Pattern Matching

- Sno LIKE ‘20131678’

  i.e.   Sno = ‘20131678’

If no Pattern-Matching symbol（% , _）：

- Sno NOT LIKE ‘20131678’

  i.e.   Sno <> ‘20131678’

Example 3.16 Pattern Matching

```
SELECT Sname, Sno, Ssex
FROM Student
WHERE Sname LIKE ‘王%’; (姓王的)

SELECT Sname
FROM Student
WHERE Sname LIKE ‘夏侯__’;
                    (two underscores，三个汉字)
```

Example 3.16 Pattern Matching

```
SELECT Sname，Sno
FROM Student
WHERE Sname LIKE ‘__丽%’; (第2个字为“丽“)

SELECT Sname
FROM Student
WHERE Sname NOT LIKE ‘王%’; (不姓王)
```

Example 3.16 Pattern Matching

- ◆**Cname LIKE ‘DB_Design’ ESCAPE '\‘**

“_” itself

- ◆Search for the course named ‘**DB_Design**’

  查找课程名为“**DB_Design**”的课程

```
SELECT   Cno, credit
FROM       C
WHERE    Cname LIKE 'DB\_Design' ESCAPE '\' ;
```

```
select *  from c
where cname like 'DB！_%'  ESCAPE '！’
```

OK!

在**SQL Server** 中，单引号里 “大小写不敏感”。
转义符只对紧随其后的通配符起作用！
如：’\%%’ 其中，第1个%是其本身，第2个%是仍通配符

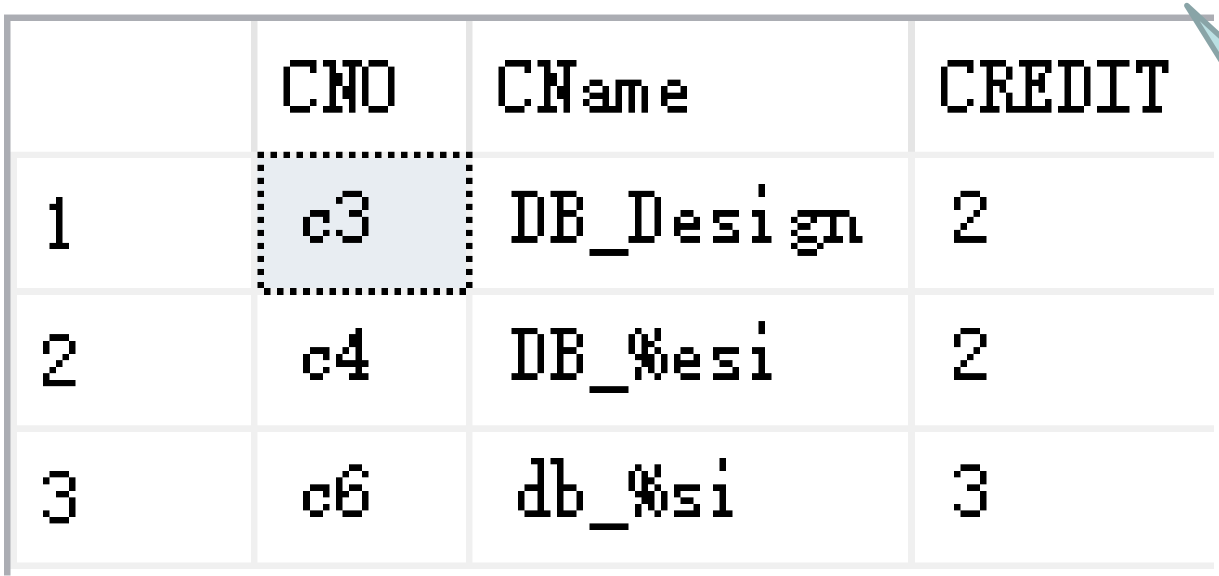
Example 3.16 Pattern Matching

Search for the course whose name begins with “DB_” and the last letter but two is ”**i**”.

查找以**"DB_"**开头，且倒数第**3**个字符为**i**的课程的详细情况。

```
SELECT *
FROM C
WHERE Cname LIKE 'DB\_%i__' ESCAPE '\' ;
```

此处的%仍然是通配符



	CNO	CName	CREDIT
1	c3	DB_Design	2
2	c4	DB_%esi	2
3	c6	db_%si	3

此结果是正确的
因为‘_’可以匹配一个或0个字符

Example 3.16 Pattern Matching

- **LIKE ’15\%’ ESCAPE’\’   i.e.   =’15%’**

- Search for the course whose name begins with “DB_” and the last letter but two is ”**i**”.

查找以**"DB_"**开头，且倒数第**3**个字符为**i**的课程的详细情况。

```
SELECT *
FROM C
WHERE Cname LIKE 'DB\_%i__' ESCAPE '\' ;
```

其中只有_被转义为其本身，%不转义，仍然表示通配符，若要将%也转义，需写成:

LIKE ‘DB_\%i__' ESCAPE '\'

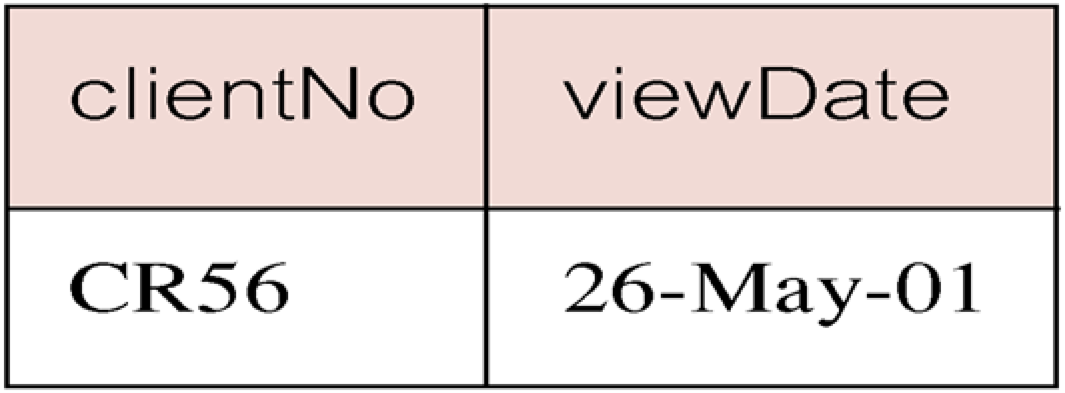
Example 3.17 NULL Search Condition

List details of all viewings on property PG4 where a comment has not been supplied.

- There are 2 viewings for property PG4, one with and one without a comment.
- Have to test for null explicitly using special keyword IS NULL:

```
SELECT   clientNo, viewDate
FROM     Viewing
WHERE    propertyNo = ‘PG4’
         AND comment IS NULL;
```

Example 3.17 NULL Search Condition



clientNo	viewDate
CR56	26-May-01

- Negated version (IS NOT NULL) can test for non-null values.
- IS can not be replaced by “=“

Example 3.18 Single Column Ordering

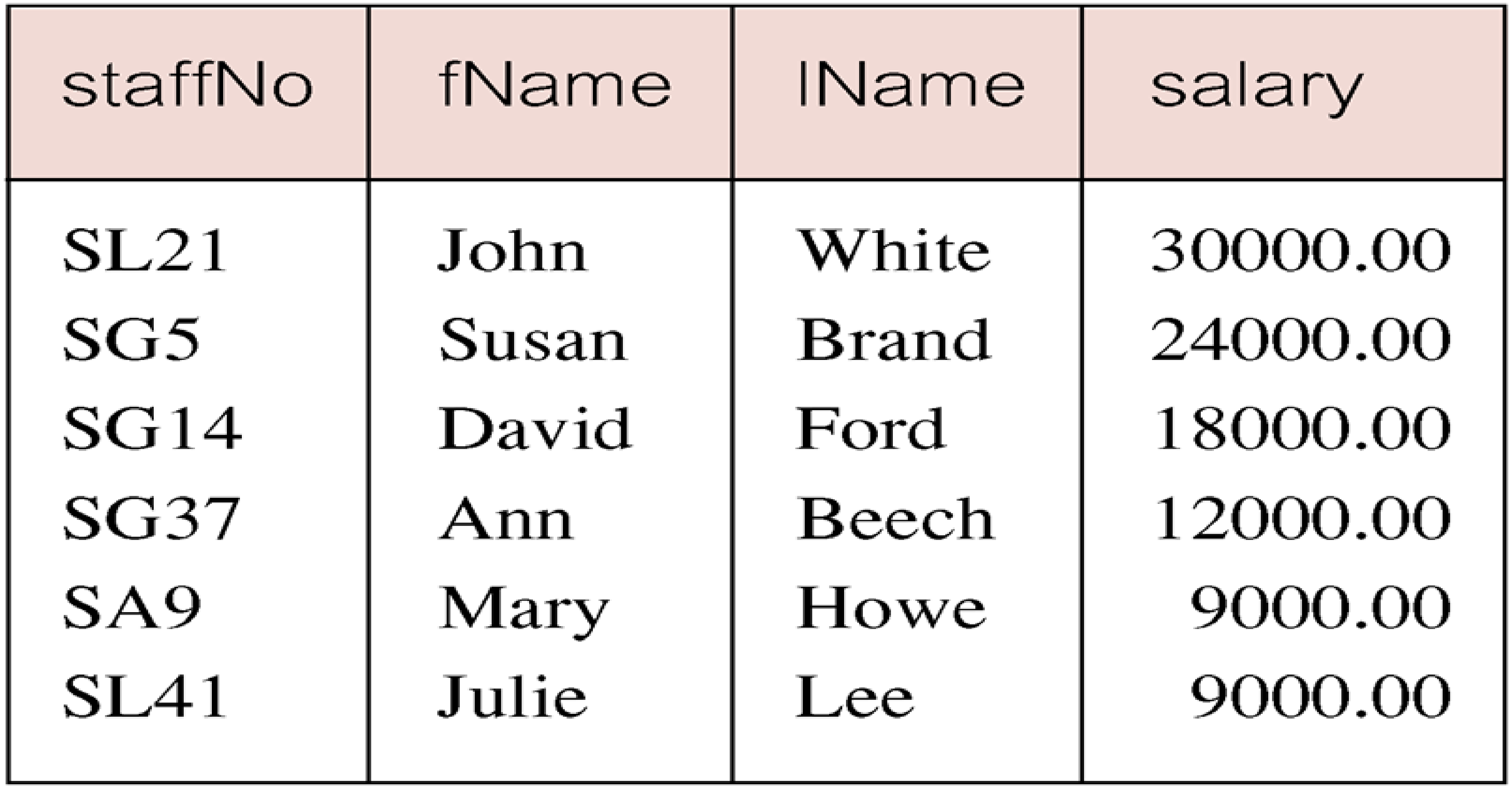
- **ORDER BY can make the result sort according to one or more columns.**
- Ascending (**ASC, default**) or descending (**DESC**)
- When sorting, the order of **null** value is determined by the chosen **DBMS**.

List the salaries for all staff, arranged in descending order of salary.

```
SELECT staffNo, fName, lName, salary
FROM Staff
ORDER BY salary DESC;
```

Example 3.18 Single Column Ordering

```
SELECT staffNo, fName, lName, salary
FROM Staff
ORDER BY salary DESC;
```



staffNo	fName	lName	salary
SL21	John	White	30000.00
SG5	Susan	Brand	24000.00
SG14	David	Ford	18000.00
SG37	Ann	Beech	12000.00
SA9	Mary	Howe	9000.00
SL41	Julie	Lee	9000.00

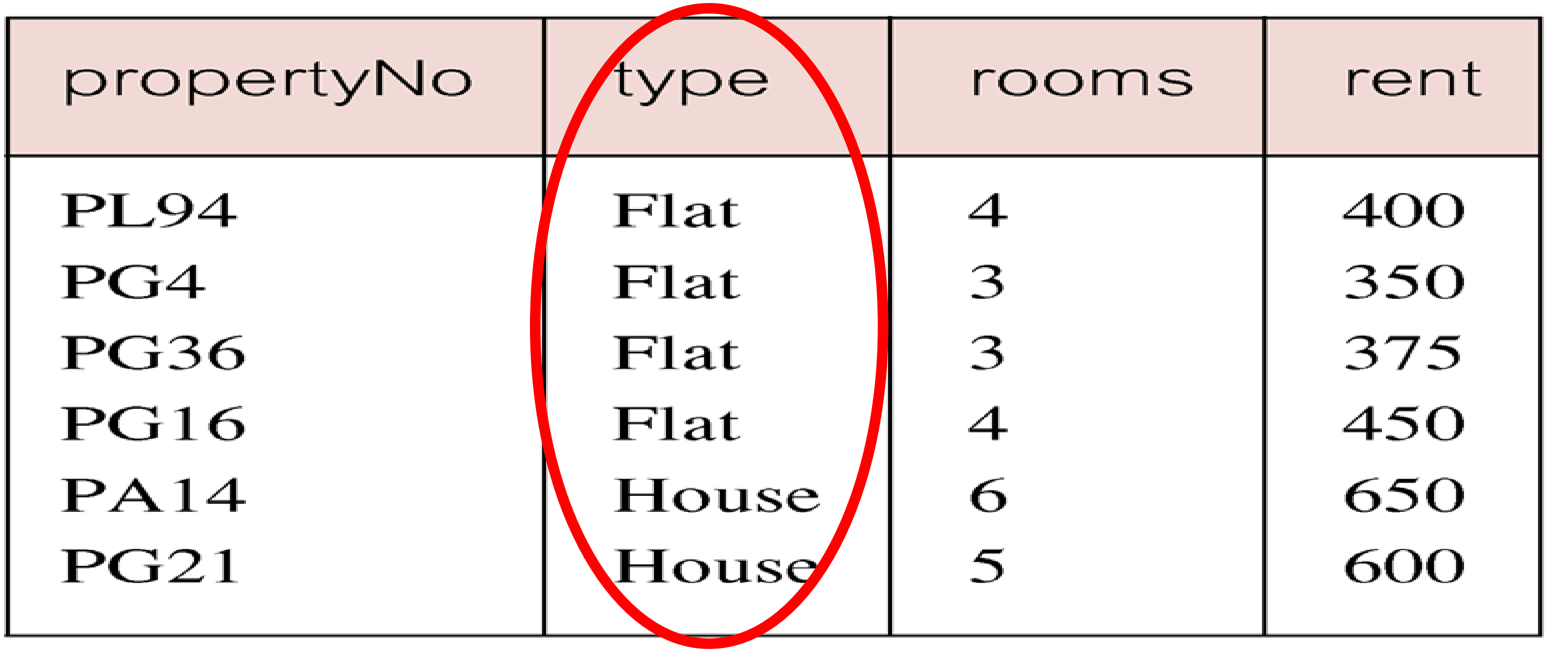
Example 3.19 Multiple Column Ordering

Produce abbreviated list（簡表）of properties in order of property type.

```
SELECT propertyNo, type, rooms, rent
FROM Property
ORDER BY type;
```

Example 3.19 Multiple Column Ordering

Result table with only one Sort key “type”



propertyNo	type	rooms	rent
PL94	Flat	4	400
PG4	Flat	3	350
PG36	Flat	3	375
PG16	Flat	4	450
PA14	House	6	650
PG21	House	5	600

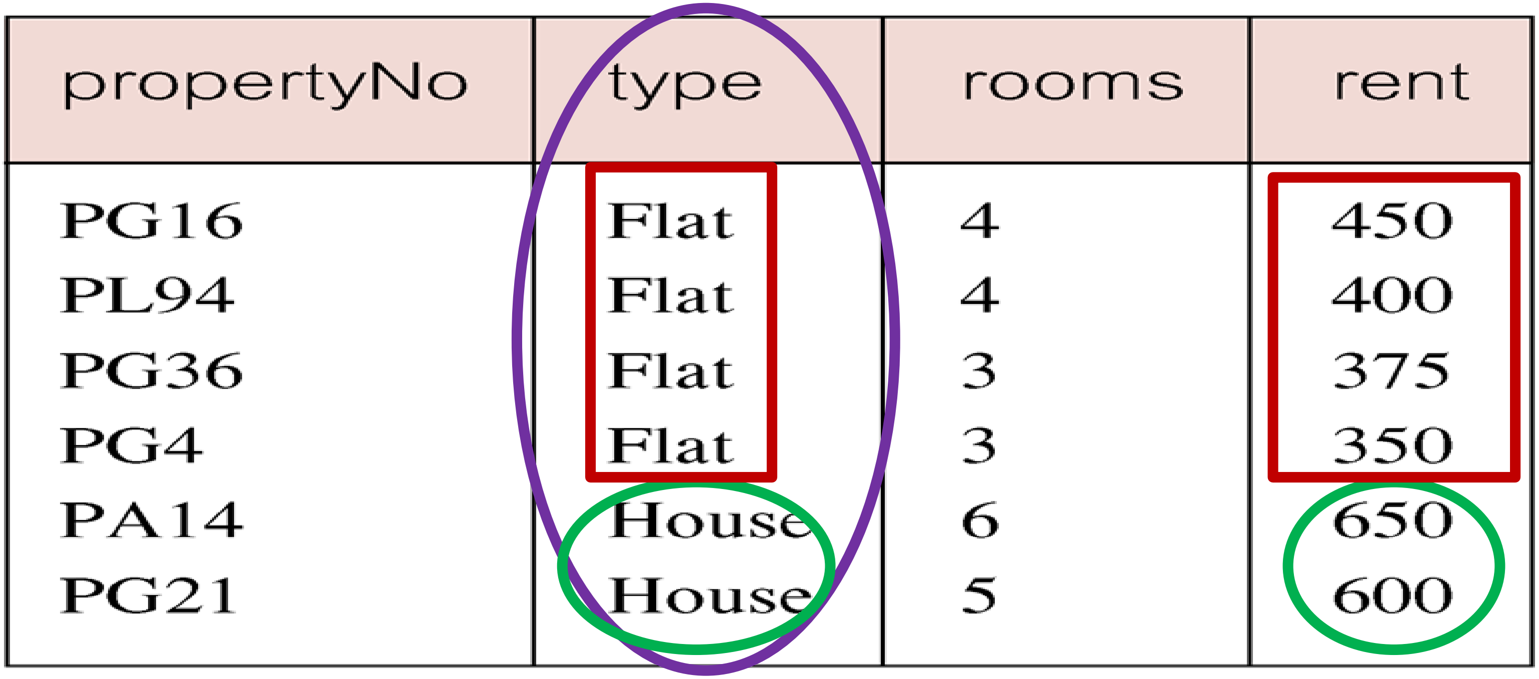
Note: Four flats in this list - as no minor sort key specified, system arranges these rows in any order it chooses.

Example 3.19 Multiple Column Ordering

To arrange in order of rent, specify minor order:

```
SELECT propertyNo, type, rooms, rent
FROM Property
ORDER BY type, rent DESC;
```

Result table with two Sort keys “type” & “rent”



propertyNo	type	rooms	rent
PG16	Flat	4	450
PL94	Flat	4	400
PG36	Flat	3	375
PG4	Flat	3	350
PA14	House	6	650
PG21	House	5	600

SELECT Statement - Aggregates

- ISO standard defines five aggregate functions:

COUNT returns the number of values in a specified column.

SUM returns the sum of values in a specified column.

AVG returns the average of values in a specified column.

MIN returns the smallest value in a specified column.

MAX returns the largest value in a specified column.

SELECT Statement - Aggregates

- Each operates on a single column of a table and returns a single value.
- COUNT, MIN, and MAX apply to numeric and non-numeric fields, but SUM and AVG may be used on numeric fields only.
- Can use DISTINCT before column name to eliminate duplicates.
- DISTINCT has no effect with MIN/MAX, but may have with SUM/AVG.

SELECT Statement - Aggregates

- Apart from COUNT(*), each function eliminates nulls first and operates only on remaining non-null values.
- COUNT(*) counts all rows of a table, regardless of whether nulls or duplicate values occur.

SELECT Statement - Aggregates

- Examples of COUNT

sno	sname	age
s1	ss1	18
s2	ss2	17
s3	ss3	null
s4	ss4	18

```
Select  COUNT(age) as num_age
from  s
```

col1
4

num_age
4

```
Select  Count (*)
from  s
```

col1
3

col1
4

```
Select  COUNT(sname)
from  s
```

col1
4

SELECT Statement - Aggregates

- Examples of AVG (sum)

sno	sname	age
s1	ss1	19
s2	ss2	17
s3	ss3	null
s4	ss4	18

```
Select   AVG(age)   as  avg_age
from   s
```

col1
17

avg_age
17

col1
18

AVG (age)=17.75 → floor(17.75) =17

```
Select  AVG(sno)
from   s
```

Wrong!

```
Select   AVG(distinct age)
from   s
```

col1
18

SELECT Statement - Aggregates

- Examples of MIN (MAX)

sno	sname	age
s1	ss1	18
s2	ss2	17
s3	ss3	null
s4	ss4	18

```
Select   MIN(age)   as  min_age
from   s
```

col1
17

min_age
17

```
Select   MIN(sno)
from   s
```

col1
s1

```
Select   MIN(distinct age)
from   s
```

col1
17

DISTINCT has no effect with MIN/MAX

SELECT Statement - Aggregates

- **Aggregate functions can be used only in SELECT list and in HAVING clause.**
- **Aggregate functions cannot be used in WHERE clause.**
- **The following sentence is wrong!**

```
SELECT Sno, AVG(Grade)
FROM SC
WHERE AVG(Grade)>=90
```

SELECT Statement - Aggregates

- If SELECT list includes an aggregate function and there is no GROUP BY clause, SELECT list cannot reference a column out of an aggregate function. For example, the following is illegal:

```
SELECT staffNo, COUNT(salary)
FROM Staff;
```

SELECT Statement - Aggregates

```
SELECT staffNo, COUNT(salary)
 FROM Staff;
```

staff

staffNo	…	salary
s1		8000
s2		6500
s3		5670
s4		9200

Result table

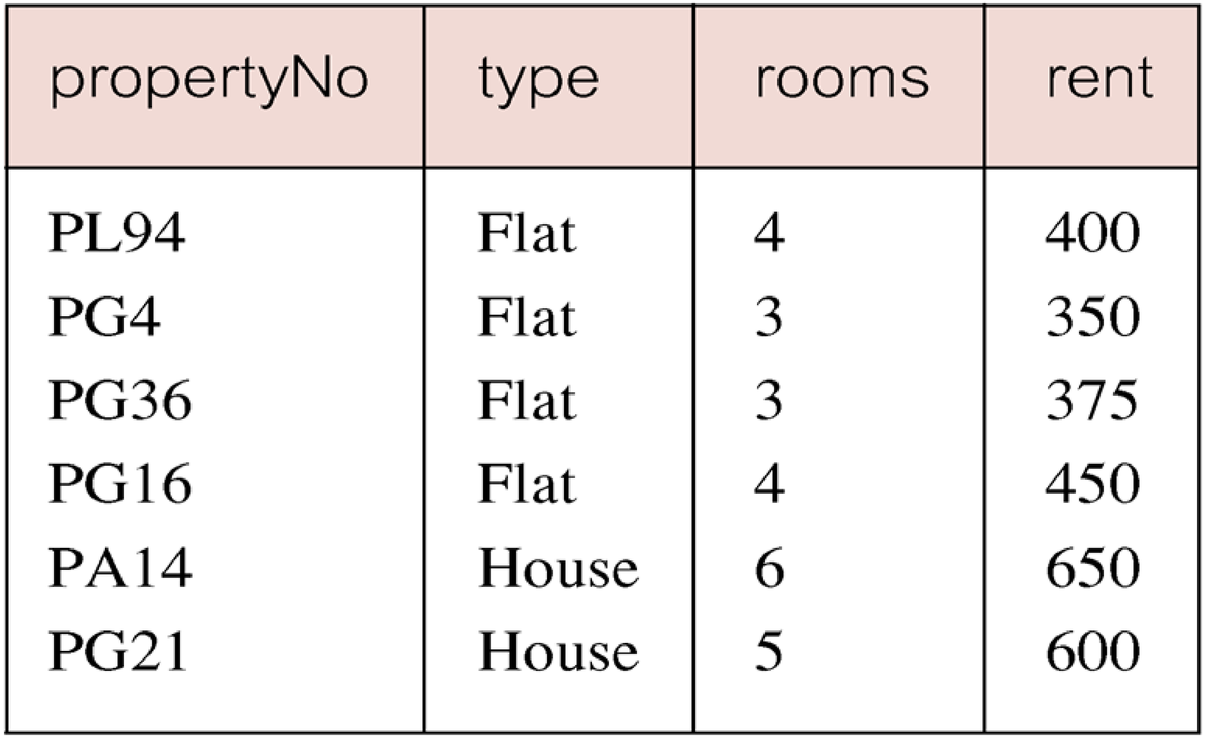
staffNo	col1
s1	4
s2	4
s3	4
s4	4

Such table has no semantic meaning!

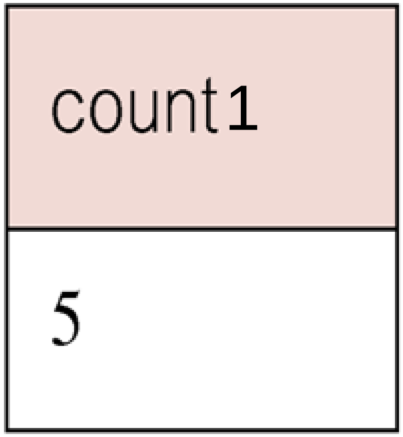
Example 3.20 Use of COUNT(*)

How many properties cost more than £350 per month to rent?

```
SELECT COUNT(*) AS count1
FROM Property
WHERE rent > 350;
```



propertyNo	type	rooms	rent
PL94	Flat	4	400
PG4	Flat	3	350
PG36	Flat	3	375
PG16	Flat	4	450
PA14	House	6	650
PG21	House	5	600

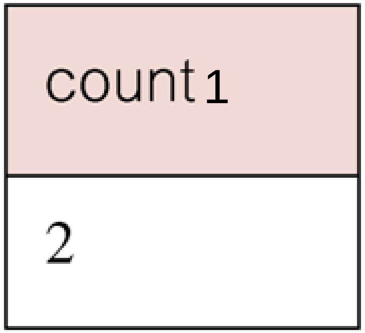


count1
5

Example 3.21 Use of COUNT (DISTINCT)

How many different properties viewed in May ’19 (2019)?

```
SELECT COUNT (DISTINCT propertyNo)
             AS count1
FROM Viewing
WHERE viewDate BETWEEN ‘1-May-2019’
        AND ‘31-May-2019’ ;
```

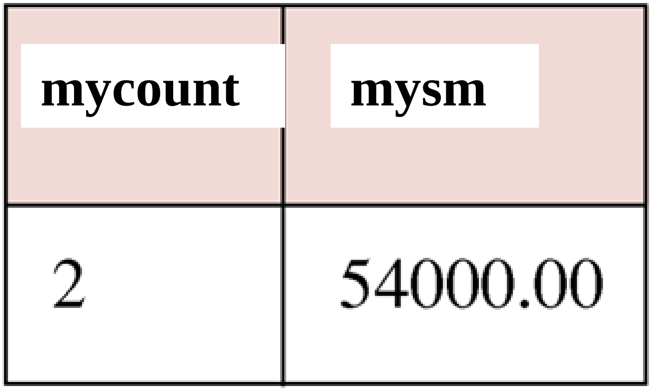


count1
2

Example 3.22 Use of COUNT and SUM

Find the number of Managers and the sum of their salaries.

```
SELECT COUNT(staffNo) AS mycount,
        SUM(salary) AS mysum
FROM Staff
WHERE position = ‘Manager’;
```

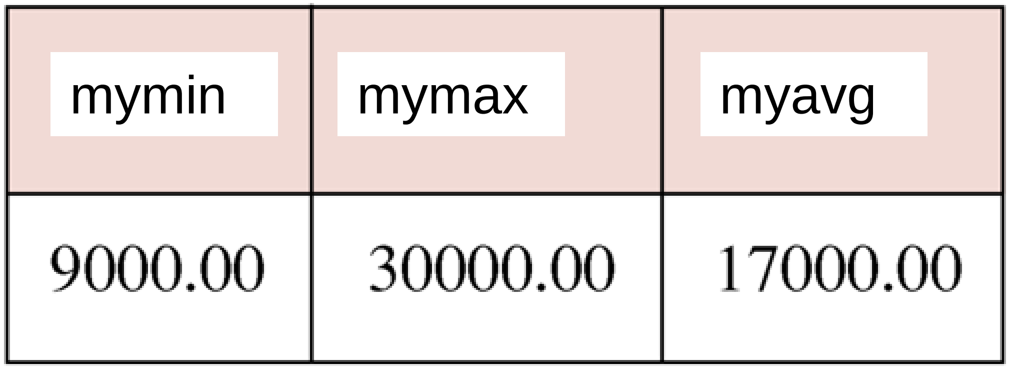


mycount	mysm
2	54000.00

Example 3.23 Use of MIN, MAX, AVG

Find the minimum, the maximum, and the average staff salary.

```
SELECT MIN(salary) AS mymin,
          MAX(salary) AS mymax,
          AVG(salary) AS myavg
 FROM Staff;
```



mymin	mymax	myavg
9000.00	30000.00	17000.00

SELECT Statement – Grouping

- Use GROUP BY clause to get sub-totals.
- SELECT and GROUP BY closely integrated: each item in SELECT list must be *single-valued per group*, and SELECT clause may only contain:
  - column names
  - aggregate functions
  - constants
  - expression involving combinations of the above.

SELECT Statement - Grouping

- **All column names in SELECT list must appear in GROUP BY clause unless the name is used only in an aggregate function.**

```
select c.cno, cname, avg(grade)
from sc,course c
where sc.cno=c.cno
group by c.cno
```

Column 'course.cname' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

SELECT Statement - Grouping

- If WHERE is used with GROUP BY, WHERE is applied first, then groups are formed from the remaining rows satisfying predicate.

- ISO considers two nulls to be equal for purposes of GROUP BY.

Example 3.24 Use of GROUP BY

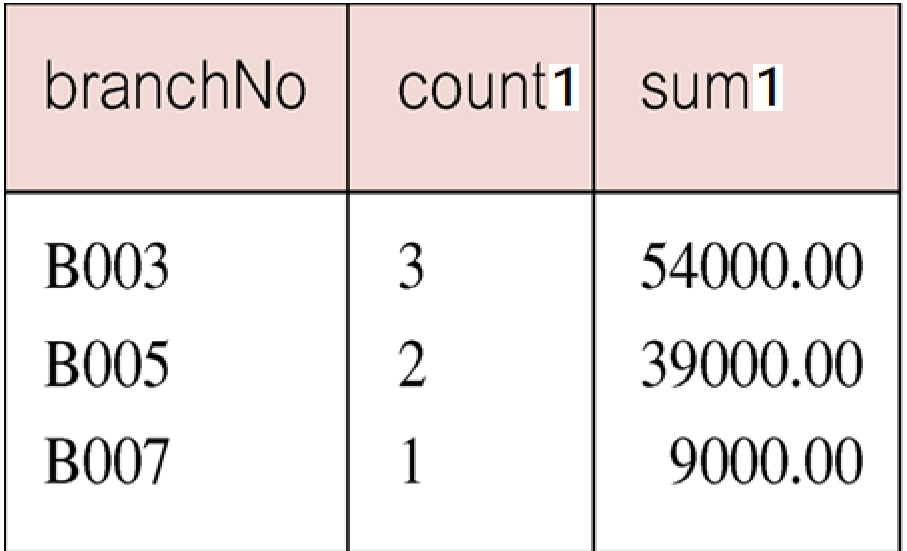
Find the number of staff in each branch and their total salaries.

staffNo	name	salary	branchNo	….
s1	John	14000	B003	
s2	Jane	15000	B003	
s3	Lily	28000	B005	
s4	Mary	25000	B003	
s5	Holy	11000	B005	
s6	Kane	9000	B007	

Example 3.24 Use of GROUP BY

staffNo	name	salary	branchNo	….
s1	John	14000	B003	
s2	Jane	15000	B003	
s3	Lily	28000	B005	
s4	Mary	25000	B003	
s5	Holy	11000	B005	
s6	Kane	9000	B007	

Result table for Example 3.24



branchNo	count1	sum1
B003	3	54000.00
B005	2	39000.00
B007	1	9000.00

```
SELECT   branchNo,
      COUNT(staffNo) AS count1,
      SUM(salary) AS sum1
FROM Staff
GROUP BY branchNo
ORDER BY branchNo;
```

Restricted Groupings – HAVING clause

- HAVING clause is designed for use with GROUP BY to restrict groups that appear in final result table.

- Similar to WHERE, but WHERE filters individual rows whereas HAVING filters groups.

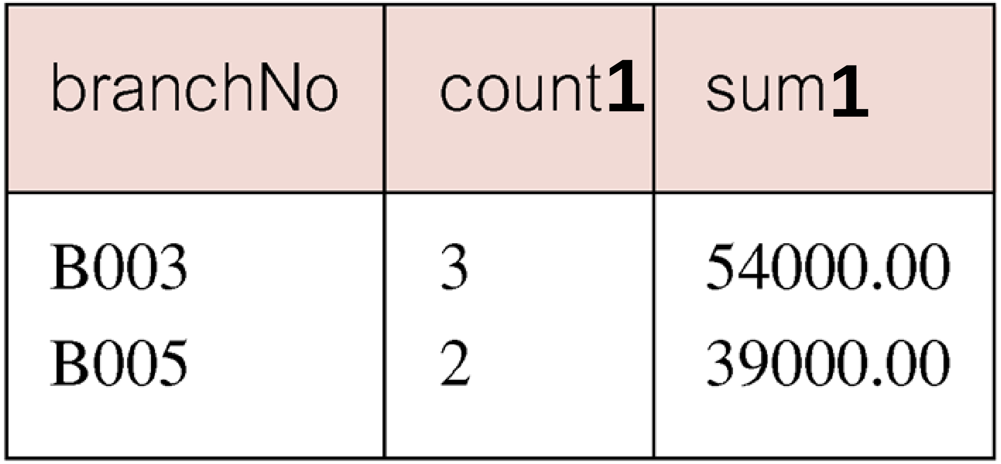
- Column names in HAVING clause must also appear in the GROUP BY list or be contained within an aggregate function.

Example 3.25 Use of HAVING

For each branch with more than 1 member of staff, find the number of staff in each branch and sum of their salaries.

staffNo	name	salary	branchNo	….
s1	John	14000	B003	
s2	Jane	15000	B003	
s3	Lily	28000	B005	
s4	Mary	25000	B003	
s5	Holy	11000	B005	
s6	Kane	9000	B007	

Result table for Example 3.25



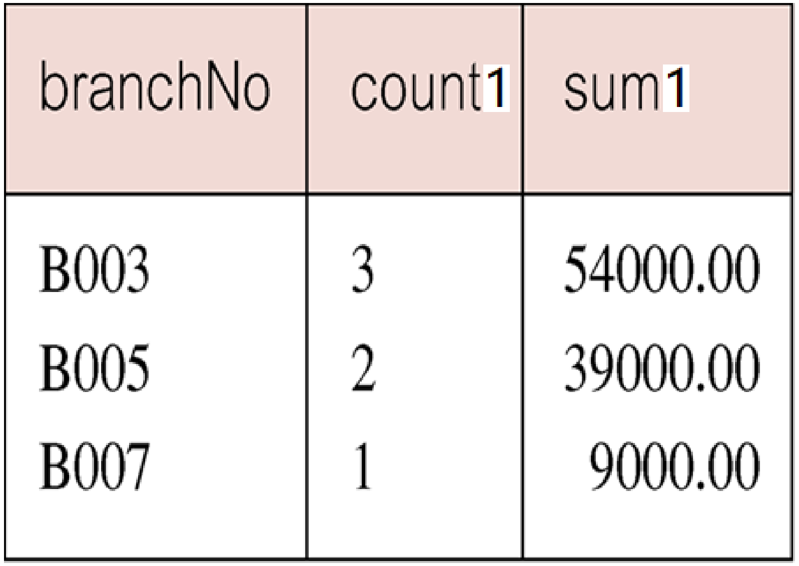
branchNo	count1	sum1
B003	3	54000.00
B005	2	39000.00

Example 3.25 Use of HAVING

staffNo	name	salary	branchNo	….
s1	John	14000	B003	
s2	Jane	15000	B003	
s3	Lily	28000	B005	
s4	Mary	25000	B003	
s5	Holy	11000	B005	
s6	Kane	9000	B007	

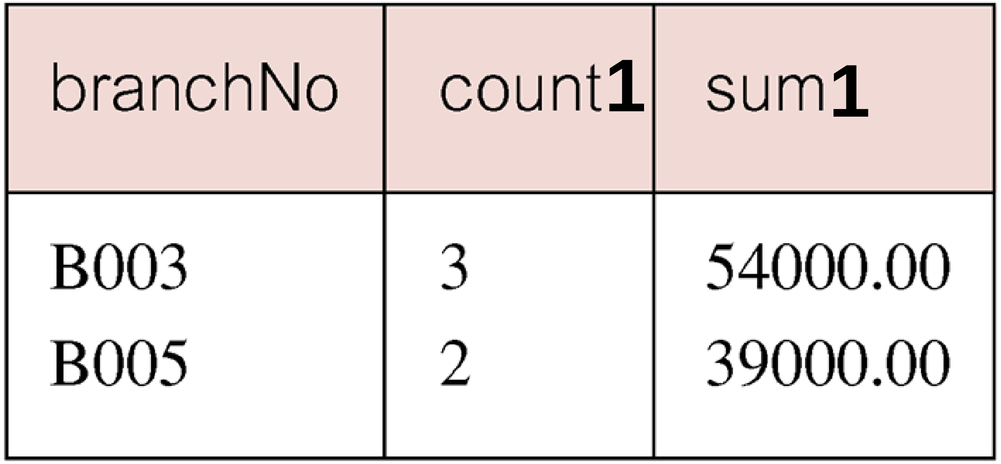
```
SELECT branchNo,
        COUNT(staffNo) AS count1,
        SUM(salary) AS sum1
FROM Staff
GROUP BY branchNo
HAVING COUNT(staffNo) > 1
ORDER BY branchNo;
```

HAVING COUNT1 > 1 ?   No !



branchNo	count1	sum1
B003	3	54000.00
B005	2	39000.00
B007	1	9000.00

Result table for Example 3.25



branchNo	count1	sum1
B003	3	54000.00
B005	2	39000.00

3.3.2 Subqueries

- Some SQL statements can have a search block of ‘SELECT-FROM-WHERE’ embedded within them.
- A subselect can be used in WHERE and HAVING clauses of an outer SELECT, where it is called a subquery or nested query(子查询或嵌套查询).
- Subselects may also appear in INSERT, UPDATE, and DELETE statements.

Example 3.26 Subquery with Equality

List staff who work in branch at ‘163 Main St’.

Outer SELECT

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE branchNo =
          ( SELECT branchNo
            FROM Branch
            WHERE street = ‘163 Main St’);
```

use IN?

Inner SELECT

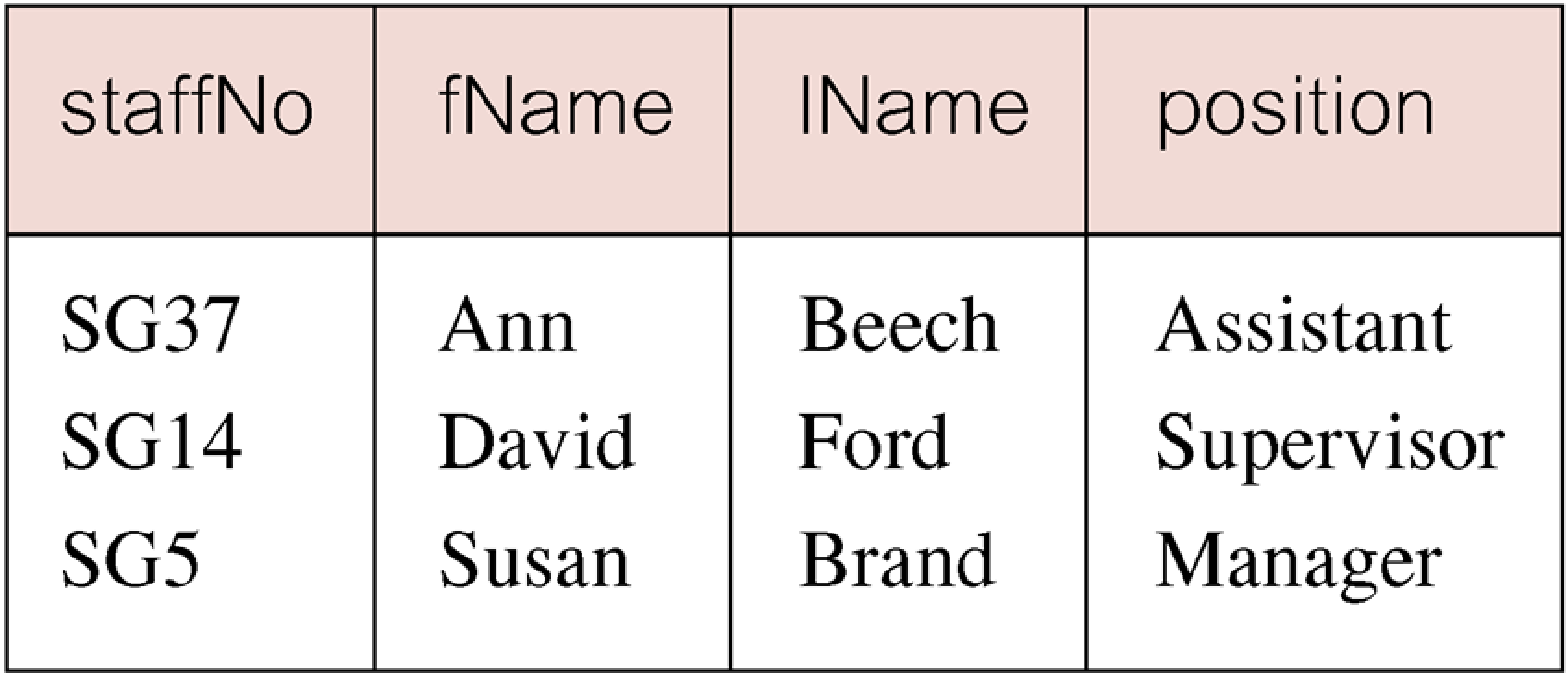
Example 3.26 Subquery with Equality

- Inner SELECT finds branch number for branch at ‘163 Main St’ (‘B003’).
- Outer SELECT then retrieves details of all staff who work at this branch.
- Outer SELECT then becomes:

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE branchNo = ‘B003’;
```

Example 3.26 Subquery with Equality

Result table for Example 3.26

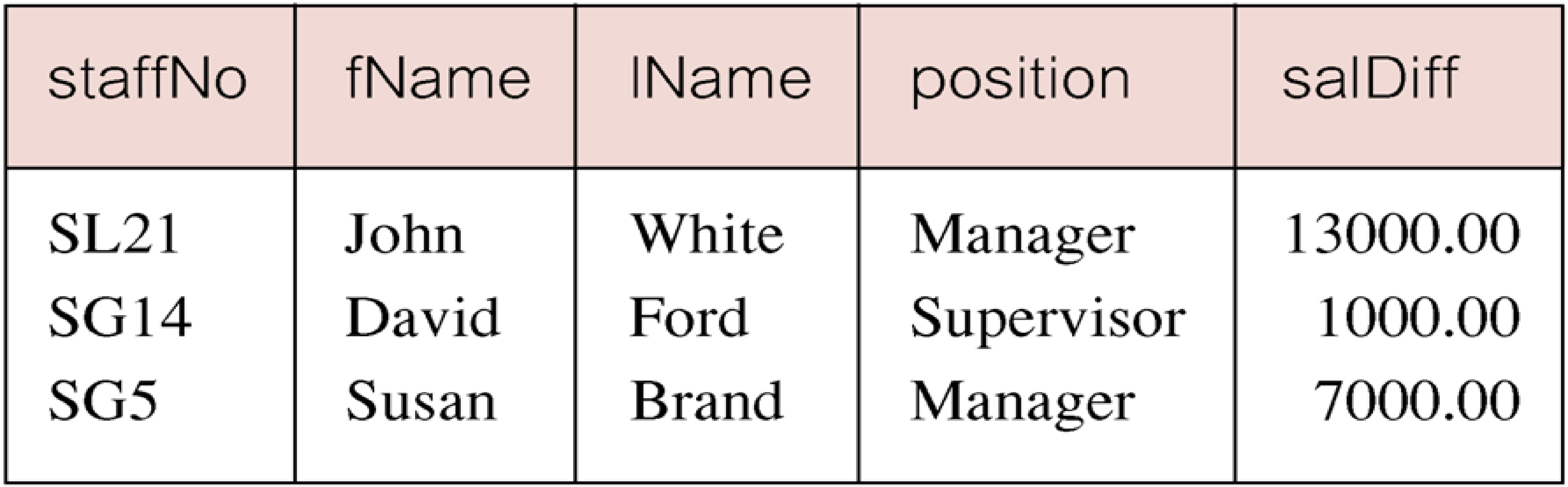


staffNo	fName	lName	position
SG37	Ann	Beech	Assistant
SG14	David	Ford	Supervisor
SG5	Susan	Brand	Manager

Example 3.27 Subquery with Aggregate

List all staff whose salary is greater than the average salary, and show by how much.

```
SELECT staffNo, fName, lName, position,
    salary – (SELECT AVG(salary) FROM Staff)  As SalDiff
FROM Staff
WHERE salary >  ( SELECT AVG(salary)   FROM Staff );
```



staffNo	fName	lName	position	salDiff
SL21	John	White	Manager	13000.00
SG14	David	Ford	Supervisor	1000.00
SG5	Susan	Brand	Manager	7000.00

Example 3.27 Subquery with Aggregate

- Cannot write ‘WHERE salary > AVG(salary)’
- Instead, use subquery to find the average salary (17000), and then use outer SELECT to find those staff with salary greater than this:

```
SELECT staffNo, fName, lName, position,
                salary – 17000 As salDiff
FROM Staff
WHERE salary > 17000;
```

```
SELECT AVG(salary)
FROM Staff
```

Example 3.27 Subquery with Aggregate

```
SELECT staffNo, fName, lName, position,
            salary – AVG(salary)  As SalDiff
FROM Staff
WHERE salary >AVG(salary)
```

Wrong!

Subquery Rules

- ORDER BY clause may not be used in a subquery (although it may be used in outermost SELECT).
- Subquery SELECT list must consist of a single column name or expression, except for subqueries that use EXISTS.
- By default, column names refer to table name in FROM clause of the same level subquery. It can refer to a table in FROM using an *alias*.
- When subquery is an operand in a comparison, subquery must appear on right-hand side.

Alias

If you specify an *alias* for a table in FROM clause, you must use it, not its original name.

```
select sno, cname, grade
from sc, course c
where sc.cno = c.cno
```

Correct！

```
select sno, cname, grade
from sc, course c
where sc.cno = course.cno
```

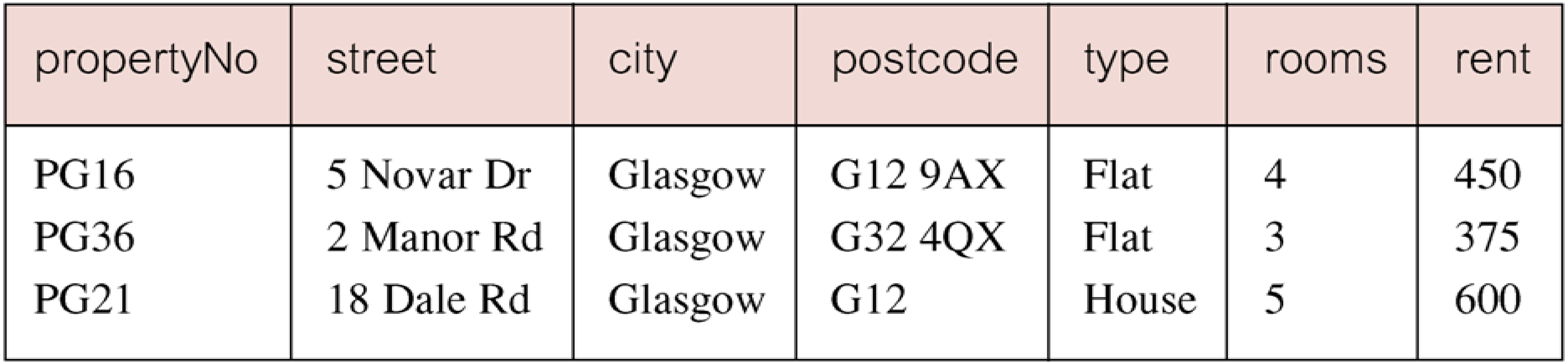
wrong！

无法绑定由多个部分组成的标识符 "course.cno"。

Example 3.28 Nested subquery: use of IN

List the properties handled by the staff at ‘163 Main St’.

```
SELECT propertyNo, street, city, postcode, type, rooms, rent
FROM Property
WHERE staffNo IN
      (SELECT staffNo
       FROM Staff
       WHERE branchNo =
                      (SELECT branchNo
                       FROM Branch
                       WHERE street = ‘163 Main St’));
```



propertyNo	street	city	postcode	type	rooms	rent
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375
PG21	18 Dale Rd	Glasgow	G12	House	5	600

ALL and ANY
universal quantifier & existing quantifier

- ANY and ALL may be used with subqueries that produce a single column, not ‘ * ’ .
- With ALL, condition will only be true if it is satisfied by *all* values produced by subquery.
- With ANY, condition will be true if it is satisfied by *any* values produced by subquery.
- If subquery is empty, ALL returns true, ANY returns false.
- SOME may be used in place of ANY in ISO standard.

ANY/SOME and ALL

	=	<> or !=	<	<=	>	>=
ANY/**SOME**	IN	Meaning-less	<MAX	<=MAX	>MIN	>=MIN
ALL	Meaning-less	NOT IN	<MIN	<=MIN	>MAX	>=MAX

★ <>all ≡ not in

★ =some ≡ in

★ =all、<>some meaningless

IF A=10

A<>all (12,10,34) F

A<>all (12,16,34) T

A=all (12,10,34) **meaningless**

A>=all (12,10,34) F

A<=all (12,10,34) T

A=any (12,10,34) T

A<any (8,16,34) T

A>any (6,16,34) T

A=any (6,16,34) F

A<>any (12,16,34) **meaningless**

Example 3.29 Use of ANY/SOME

Find the staff whose salary is larger than the salary of at least one member of staff at branch B003.

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary > SOME
          (SELECT salary    FROM Staff
           WHERE branchNo = ‘B003’);
```

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary >  ( SELECT min (salary)
                  FROM Staff
                  WHERE branchNo = ‘B003’);
```

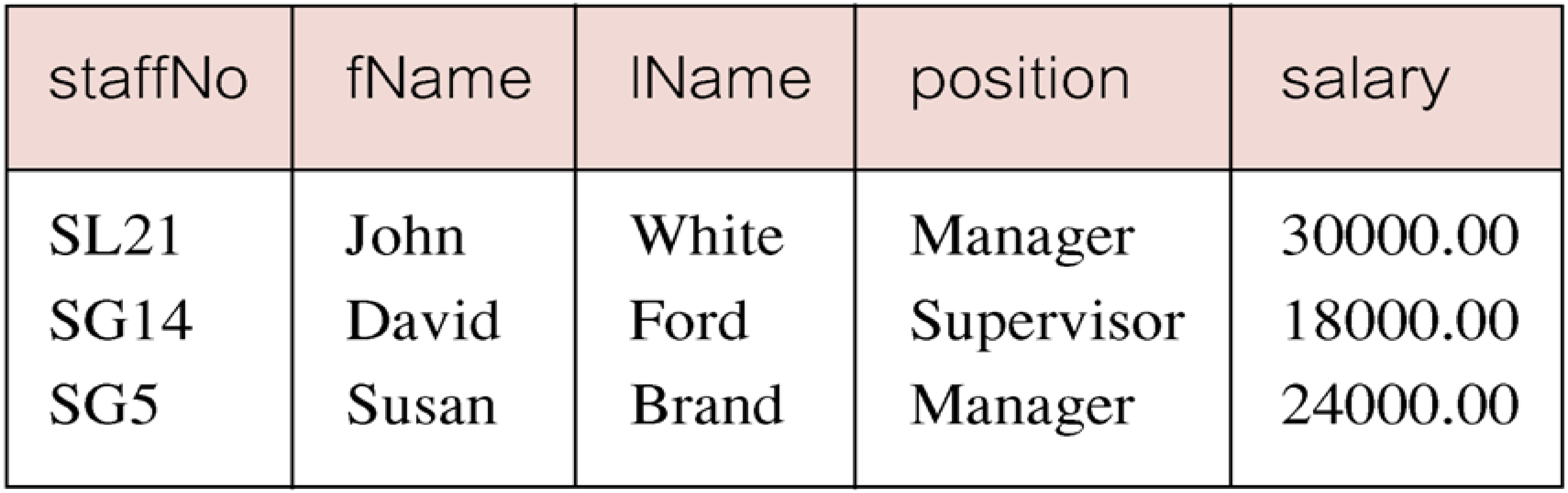
Example 3.29 Use of ANY/SOME

- **Inner query :**

```
SELECT salary    FROM Staff
WHERE branchNo = ‘B003’;
```

- **produces set {12000, 18000, 24000} and outer query selects those staff whose salaries are greater than any of the values in this set.**

The Result Set



staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG14	David	Ford	Supervisor	18000.00
SG5	Susan	Brand	Manager	24000.00

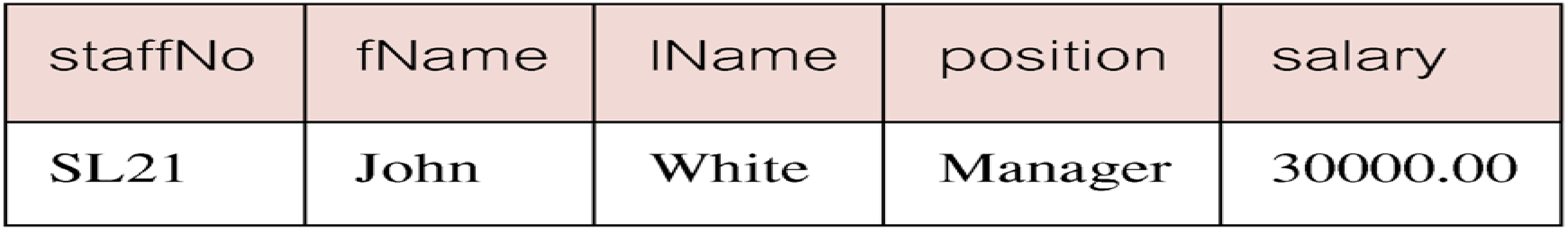
Example 3.30 Use of ALL

Find the staff whose salary is larger than salary of every member of staff at branch B003.

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary > ALL
                (SELECT salary
                 FROM Staff
                 WHERE branchNo = 'B003');
```

{12000, 18000, 24000}

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary >  (SELECT max (salary)
                    FROM Staff
                    WHERE branchNo = 'B003');
```

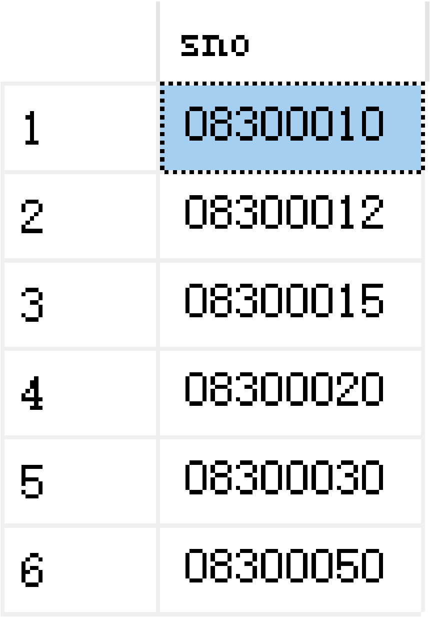


staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00

```
select * from sc
where sno > all ( select sno
                  from sc
                  group by sno
                   having count(cno)>3)
```

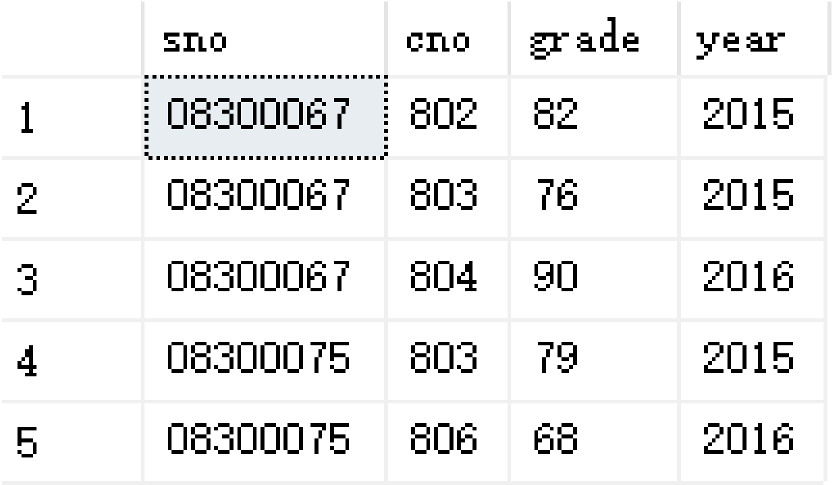
Not a numeric value,
Is it correct?

Result of subquery



	sno
1	08300010
2	08300012
3	08300015
4	08300020
5	08300030
6	08300050

Result of the above query



	sno	cno	grade	year
1	08300067	802	82	2015
2	08300067	803	76	2015
3	08300067	804	90	2016
4	08300075	803	79	2015
5	08300075	806	68	2016

Non-Correlation subquery

不相关子查询

It has the following features:

1. The non-Correlation subquery can be executed independently.

2. The innermost subquery is executed first, and then its outer query is executed until going to the outermost query one by one level.

3.3.3 Multi-Table Queries

- A major limitation of all previous examples is that the columns of the result table must all come from a single table.
- If we need to obtain information from more than one table, we can use subquery or use a join.
- If the result columns come from more than one table, we must use a join.

3.3.3 Multi-Table Queries

- ◆ To perform join, include more than one table in FROM clause, using comma as separator, and typically including a WHERE clause to specify join column(s) of join conditions.
- ◆ Also possible to use an alias for a table named in FROM clause.
- ◆ Alias is separated from table name with a space or ‘as’.
- ◆ Alias can be used to qualify column names when there is ambiguity (歧义).

Alias

If you specify an *alias* for a table in FROM clause, you must use it, not its original name.

```
select sno, cname, grade
from sc, course c
where sc.cno = c.cno
```

Correct！

```
select sno, cname, grade
from sc, course c
where sc.cno = course.cno
```

wrong！

无法绑定由多个部分组成的标识符"course.cno"。

Unable to bind the identifier made of multiple parts

Example 3.31 Simple Join

List the names of all clients who have viewed a property along with any comment supplied.

```
SELECT c.clientNo, fName, lName,
               propertyNo, comment
FROM Client c, Viewing v
WHERE c.clientNo = v.clientNo;
```

Example 3.31 Simple Join

```
SELECT c.clientNo, fName, lName,  propertyNo, comment
FROM Client c, Viewing v
WHERE c.clientNo = v.clientNo;
```

Client

clientNo	fName	lName	…….
c1	cc1	ccc1	
c2	cc2	ccc2	
c3	cc3	ccc3	
c4	cc4	ccc4	
c5	cc5	ccc5	

Viewing

pNo	clientNo	staffNo	comment	……
p1	c1	s1	good	
p3	c2	s1	Not bad	
p5	c1	s1	OK	
p2	c3	s2		
p4	c2	s2	OK	
p6	c2	s3		
p9	c1	s4	Very good	
p7	c3	s4		
p8				

Example 3.31 Simple Join

Joined Table **Result table for Example 3.31**

clientNo	fName	lName	pno	comment
c1	cc1	ccc1	p1	good
c2	cc2	ccc2	p3	Not bad
c1	cc1	ccc1	p5	OK
c3	cc3	ccc3	p2	
c2	cc2	ccc2	p4	OK
c2	cc2	ccc2	p6	
c1	cc1	ccc1	p9	Very good
c3	cc3	ccc3	p7	

Example 3.31 Simple Join

```
SELECT c.clientNo, fName, lName,
                propertyNo, comment
FROM Client c, Viewing v
WHERE c.clientNo = v.clientNo;
```

The SQL standard provides the following alternative ways to specify this join:

```
FROM Client c JOIN Viewing v
          ON c.clientNo = v.clientNo

FROM Client JOIN Viewing USING clientNo

FROM Client NATURAL JOIN Viewing
```

Example 3.32 Sorting a join

排序连接结果

For each branch, list the numbers and names of staff who manage properties, and the properties they manage.

```
SELECT s.branchNo, s.staffNo, fName,
                lName,   propertyNo
 FROM Staff s, Property p
 WHERE s.staffNo = p.staffNo
 ORDER BY s.branchNo, s.staffNo, propertyNo;
```

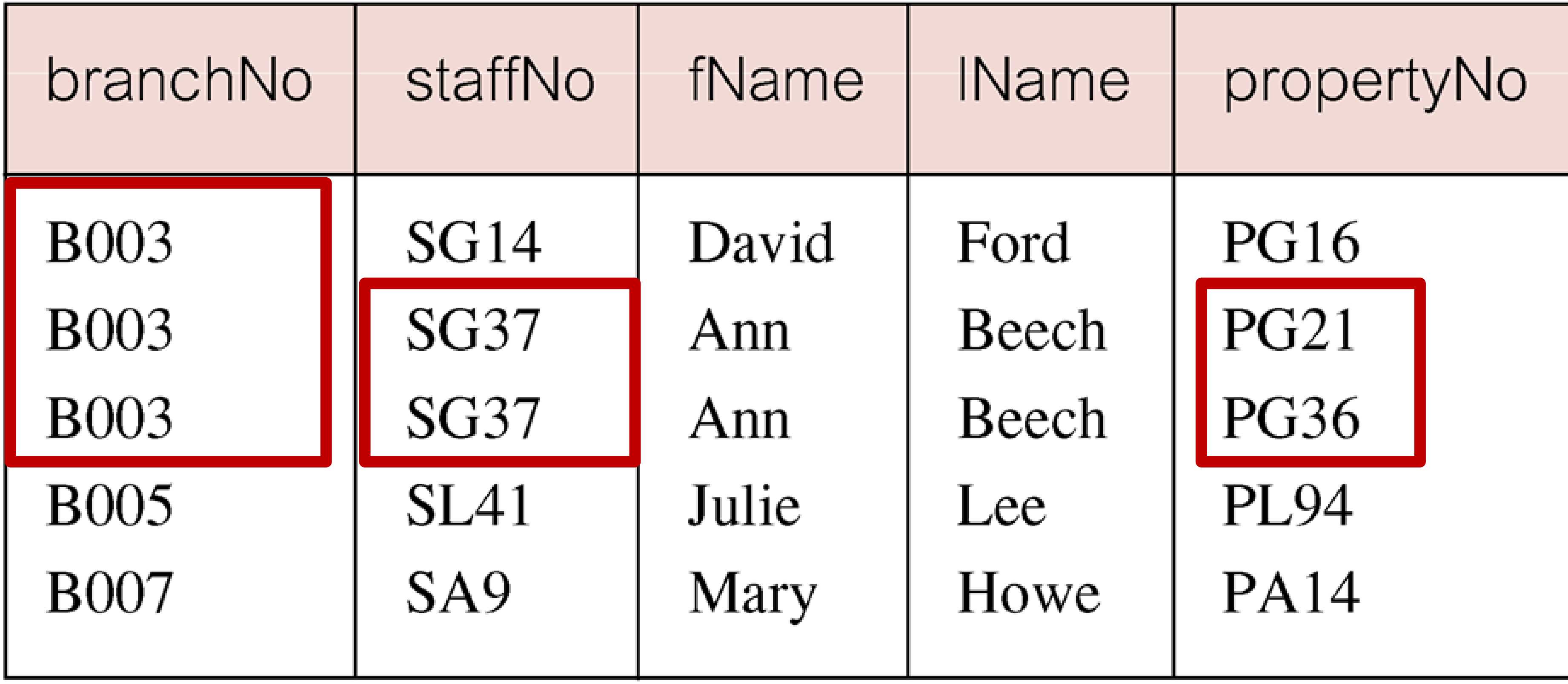
Major sort key

Minor sort key

Minor sort key

Example 3.32 Sorting a join

Result table for Example 3.32



branchNo	staffNo	fName	lName	propertyNo
B003	SG14	David	Ford	PG16
B003	SG37	Ann	Beech	PG21
B003	SG37	Ann	Beech	PG36
B005	SL41	Julie	Lee	PL94
B007	SA9	Mary	Howe	PA14

ORDER BY s.branchNo, s.staffNo, propertyNo

Example 3.33 Three-Table Join

For each branch, list staff who manage properties, including city in which branch is located and properties they manage.

```
SELECT b.branchNo, b.city, s.staffNo, fName,
          lName,  propertyNo
FROM Branch b, Staff s, Property p
WHERE b.branchNo = s.branchNo
         AND  s.staffNo = p.staffNo
ORDER BY b.branchNo, s.staffNo, propertyNo;
```

Example 3.33 Three-Table Join

Branch

branchNo	Street	City	Postcode
b1	Rd11	**London**	…
b2	St20	**Glasgow**	…
b3	Rd 234	**Glasgow**	…
b4	St33	**Bristol**	…

Property

pno	staffNo	……
p1	s1	
p3	s1	
p5	s1	
p2	s2	
p4	s2	
p6	s3	
p7	s4	
p9	s4	
p8		

Staff

staffNo	branchNo	…….
s1	b1	
s2	b1	
s3	b2	
s4	b3	
s5	b4	

b.branchNo = s.branchNo

AND s.staffNo = p.staffNo

Example 3.33 Three-Table Join

Joined table

Result table for Example 3.33

branchNo	City	staffNo	name	pno
b1	London	s1	Ann	p1
b1	London	s1	Ann	p3
b1	London	s1	Ann	p5
b1	London	s2	Jane	p2
b1	London	s2	Jane	p4
b2	Glasgow	s3	Mark	p6
b3	Glasgow	s4	David	p7
b3	Glasgow	s4	David	p9

Example 3.34 Multiple Grouping Columns
按多个列分组

Find the number of properties handled by each staff member.

```
SELECT  staffNo, COUNT(*) AS count2
FROM    Property
GROUP BY  staffNo
ORDER BY staffNo;
```

Desired result table

staffNo	count2
s1	3
s2	2
s3	1
s4	2
s5	1

Example 3.34 Multiple Grouping Columns

Find the number of properties handled by each staff member in each branch.

Staff

staffNo	branchNo	……
s1	b1	
s2	b1	
s3	b2	
s4	b3	
s5	b4	

Property

pno	staffNo	……
p1	s1	
p3	s1	
p5	s1	
p2	s2	
p4	s2	
p6	s3	
p7	s4	
p9	s4	
p8	s5	

Example 3.34 Multiple Grouping Columns

Join property and staff table.

branchNo	staffNo	pno
b1	s1	p1
b1	s1	p3
b1	s1	p5
b1	s2	p2
b1	s2	p4
b2	s3	p6
b3	s4	p7
b3	s4	p9
b4	s5	p8

Example 3.34 Multiple Grouping Columns

```
SELECT s.branchNo, s.staffNo, COUNT(*) AS count1
FROM Staff s, Property p
WHERE s.staffNo = p.staffNo
GROUP BY s.branchNo, s.staffNo      == GROUP BY  s.staffNo
ORDER BY s.branchNo, s.staffNo;
```

branchNo	staffNo	count1
b1	s1	3
b1	s2	2
b2	s3	1
b3	s4	2
b4	s5	1

Computing a Join

The procedure for generating results of a join are:

1. Form Cartesian product of the tables named in FROM clause.
2. If there is a WHERE clause, apply the search condition to each row of the product table, retaining those rows that satisfy the condition.
3. For each remaining row, determine the value of each item in SELECT list to produce a single row in the result table.
4. If DISTINCT has been specified, eliminate any duplicate rows from the result table.
5. If there is an ORDER BY clause, sort the result table as required.

e.g. R×S

Number of tuples(rows) in R×S is called

Cardinality（基数）, is equal to the product of two numbers of rows.

Number of attributes (columns) is called

Degree（元数）, is equal to the sum of two numbers of columns.

R

A	B	C
a	b	c
b	c	e
e	d	c

S

C	D
c	d
e	f

R×S

A	B	R.C	S.C	D
a	b	c	c	d
a	b	c	e	f
b	c	e	c	d
b	c	e	e	f
e	d	c	c	d
e	d	c	e	f

Outer joins

- The join operation combines data from two tables by forming pairs of related rows where the matching columns in each table have the same value.
- If one row of a table is **unmatched**, the row is omitted from the result table. This has been the case for the joins we examined above.
- The ISO standard provides another set of join operators called **outer joins** (see Chapter 2).
- The **Outer join retains rows that do not satisfy the join condition**.

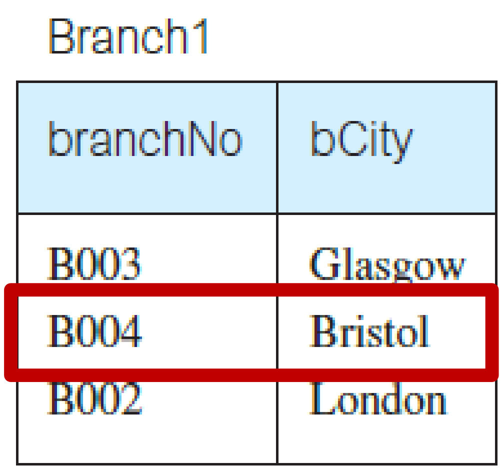
Example of Outer joins

- The (Inner) join of these two tables:

SELECT b.*, p.*

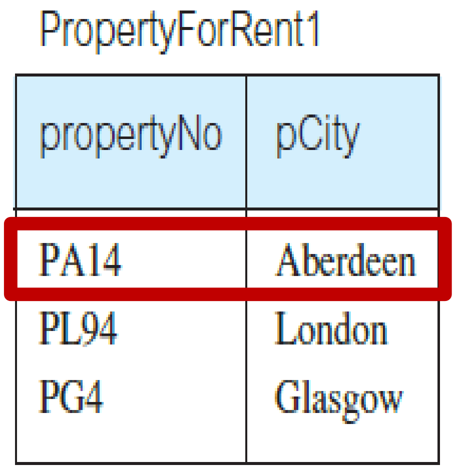
FROM Branch1 b, PropertyForRent1 p

WHERE b.bCity = p.pCity;



Branch1

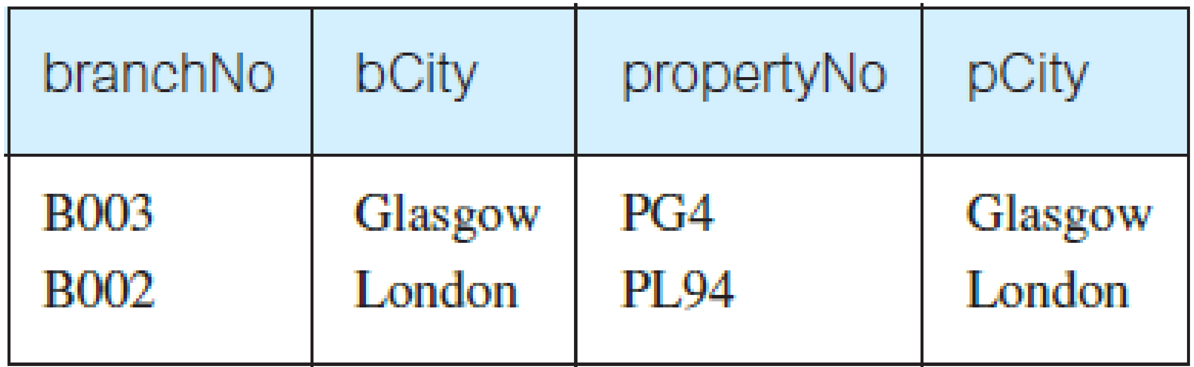
branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London



PropertyForRent1

propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow

result table



branchNo	bCity	propertyNo	pCity
B003	Glasgow	PG4	Glasgow
B002	London	PL94	London

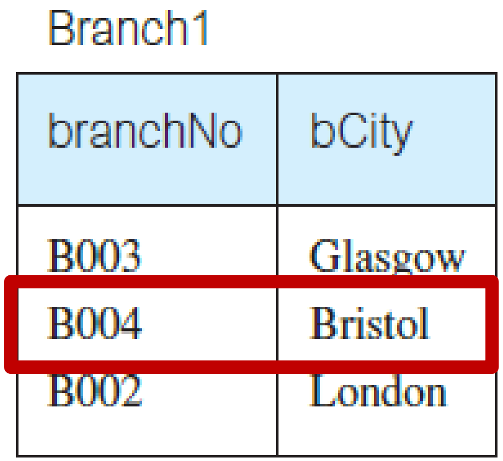
Example of Outer joins

- The (Inner) join of these two tables:

  SELECT b.*, p.*

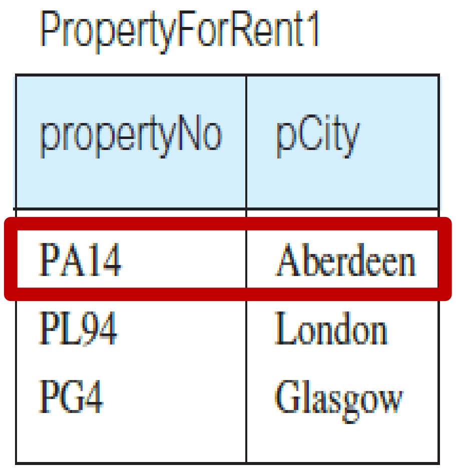
  FROM Branch1 b, PropertyForRent1 p

  WHERE b.bCity = p.pCity;



Branch1

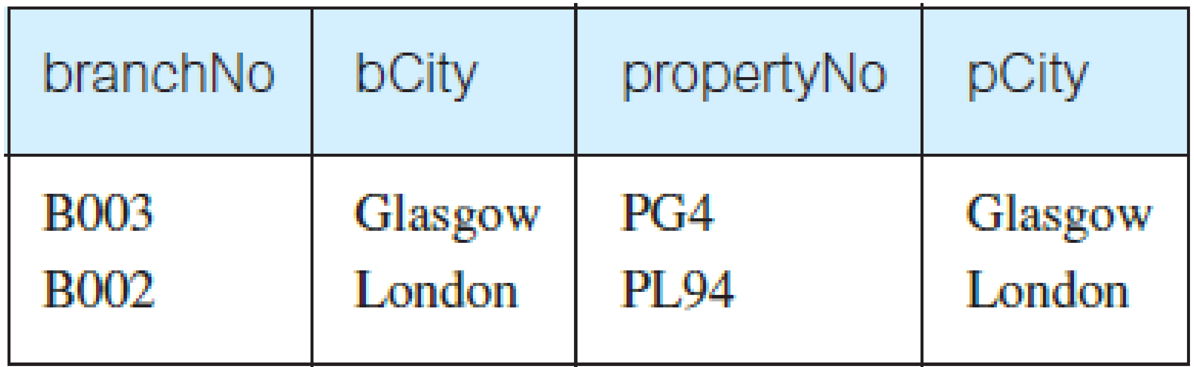
branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London



PropertyForRent1

propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow

result table



branchNo	bCity	propertyNo	pCity
B003	Glasgow	PG4	Glasgow
B002	London	PL94	London

Example of Left Outer join

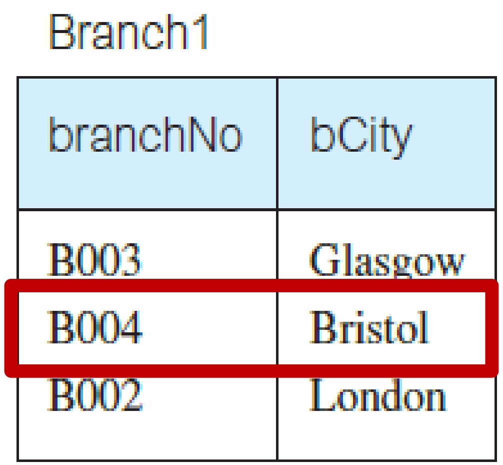
List all branch offices and any properties that are in the same city.

The **Left Outer join** of these two tables：

SELECT b.*, p.*

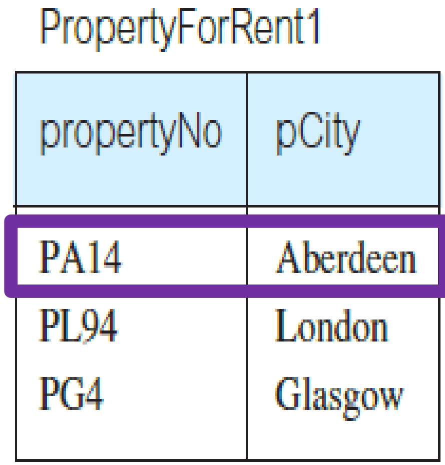
FROM Branch1 b **LEFT JOIN** PropertyForRent1 p

ON b.bCity = p.pCity；



Branch1

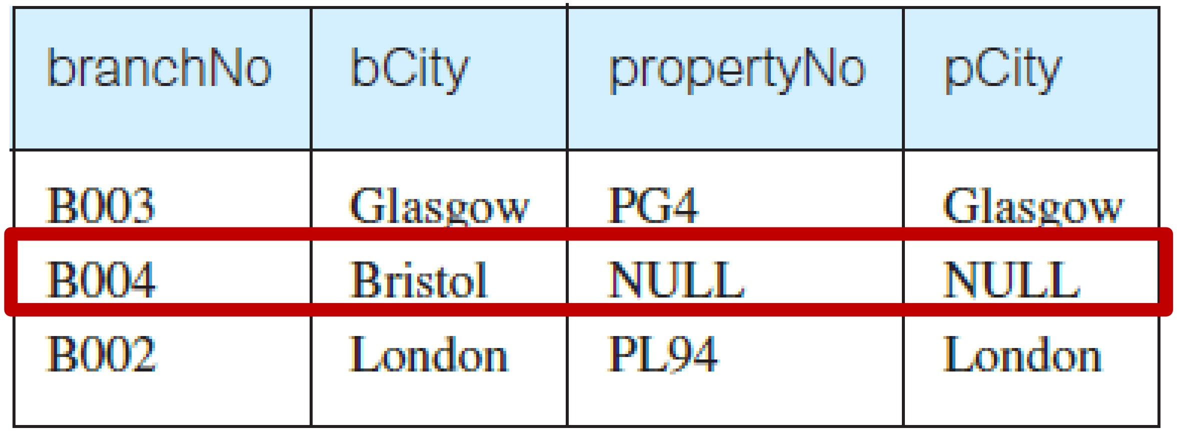
branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London



PropertyForRent1

propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow

result table



branchNo	bCity	propertyNo	pCity
B003	Glasgow	PG4	Glasgow
B004	Bristol	NULL	NULL
B002	London	PL94	London

Example of Right Outer join

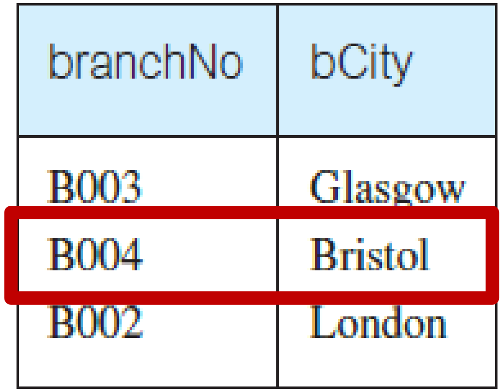
List all properties offices and any branch offices that are in the same city. The **Right Outer join** of these two tables：

SELECT b.*, p.*

FROM Branch1 b **RIGHT JOIN** PropertyForRent1 p

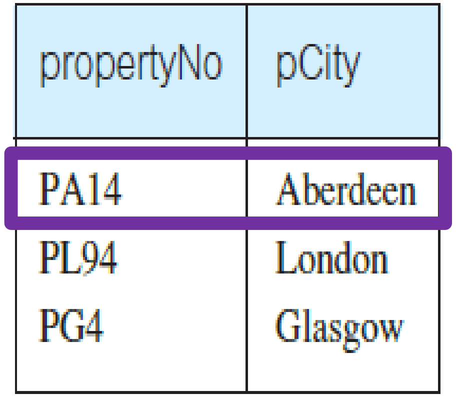
ON b.bCity = p.pCity；

Branch1



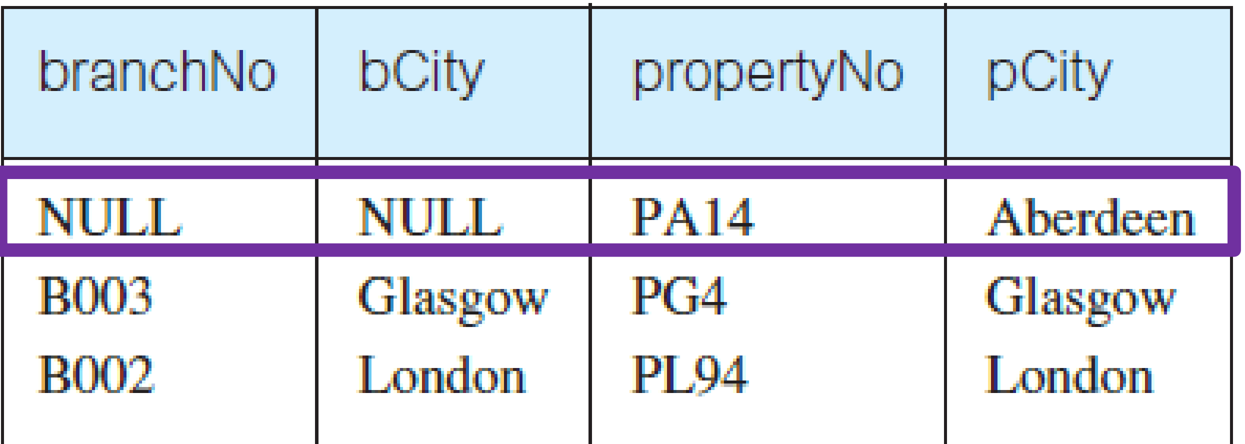
branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London

PropertyForRent1



propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow

result table



branchNo	bCity	propertyNo	pCity
NULL	NULL	PA14	Aberdeen
B003	Glasgow	PG4	Glasgow
B002	London	PL94	London

Example of Full Outer join

List the branch offices and properties that are in the same city along with any unmatched branches or properties.

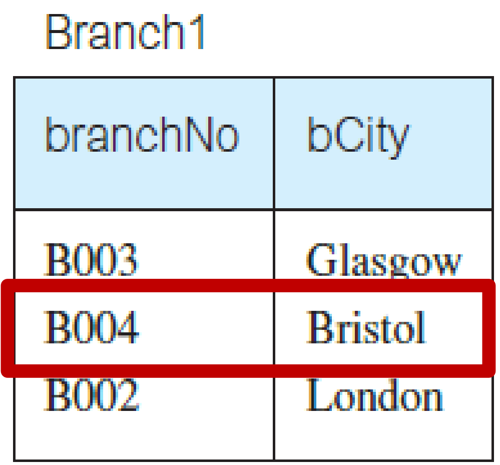
The **Full Outer join** of these two tables：

SELECT b.*, p.*

FROM Branch1 b **FULL JOIN**

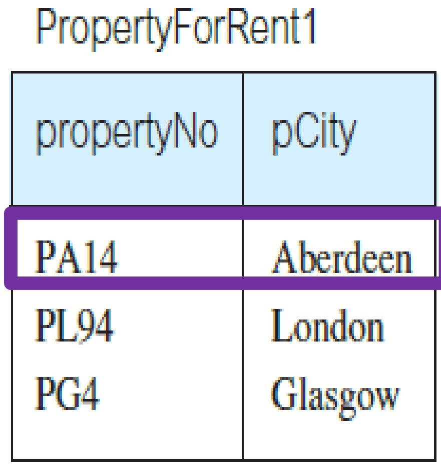
PropertyForRent1 p

ON b.bCity = p.pCity；



Branch1

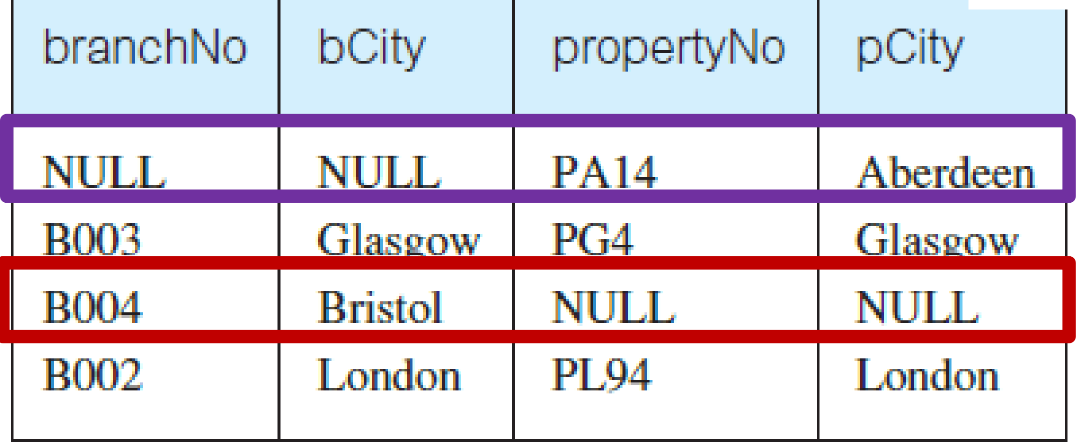
branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London



PropertyForRent1

propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow

result table



branchNo	bCity	propertyNo	pCity
NULL	NULL	PA14	Aberdeen
B003	Glasgow	PG4	Glasgow
B004	Bristol	NULL	NULL
B002	London	PL94	London

3.3.4 EXISTS and NOT EXISTS

- EXISTS and NOT EXISTS are designed for use only with subqueries.
- They produce a simple true/false result.
- True if and only if there exists at least one row in the result table returned by the subquery.
- False if the subquery returns an empty result table.
- NOT EXISTS is the opposite of EXISTS.

EXISTS and NOT EXISTS
(the third person singular)

- **Since (NOT) EXISTS check only for existence or non-existence of rows in the subquery result table, the column name of subquery is meaningless.**

- **It is common for subqueries following the form:**

  (NOT) EXISTS (SELECT * FROM ...)

带有 EXISTS 谓词的子查询

带有EXISTS谓词的子查询不返回任何数据，只产生逻辑真值“true”或假值“false”。

若子查询结果为非空，则父查询的WHERE子句返回真值，否则，返回假值。

由EXISTS引出的子查询（相关子查询），其目标列表达式通常都用*，因为带EXISTS的子查询只返回真值或假值，给出列名无实际意义。

与EXISTS对应的是NOT EXISTS谓词。

Example 3.35 Query using EXISTS

Find all staff who work in a London branch office.

```
SELECT staffNo, fName, lName, position
FROM Staff s
WHERE EXISTS
          ( SELECT *
            FROM Branch b
            WHERE s.branchNo = b.branchNo
                  AND  city = ‘London’);
```

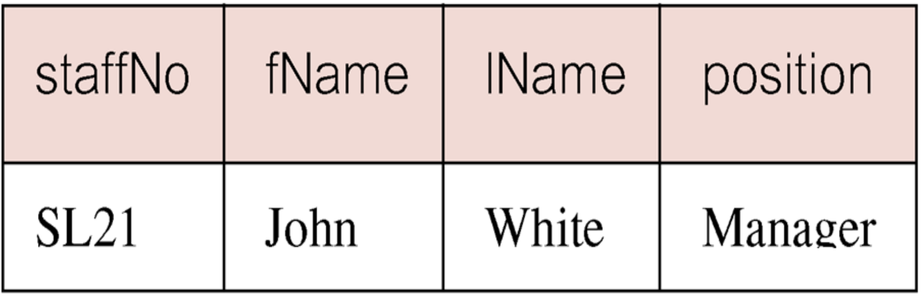
s.branchNo = b.branchNo AND city = ‘London

Staff

staffNo	fName	lName	branchNo	position
SL19	丁岩	Crane	B001	……
SL20	王爽	Kite	B003	……
SL21	John	White	B002	Manager
SL23	赵立	Lee	B001	……

Branch

branchNo	city	……
B001	Paris	
B002	London	
B003	Bristol	



staffNo	fName	lName	position
SL21	John	White	Manager

Correlation subquery

It has the following features:

1. The Correlation subquery can not be executed independently.

2. The execution procedure is similar to that of loop statement in high-level programming language. The outermost subquery is executed first, then go inside the second level outer query until going to the innermost query one by one level. （in the opposite direction to Non-Correlation subquery）

Example 3.35 Query using EXISTS

```
  ( SELECT *  FROM Branch b
    WHERE s.branchNo = b.branchNo  AND  city = ‘London’);
```

- **Note, the first search condition s.branchNo = b.branchNo is necessary.**
- **If omitted, we would get all staff records listed out because subquery:**

```
SELECT * FROM Branch
WHERE city=‘London’
```

would always be true and query would be:

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE true;
```

Example 3.35 Query using EXISTS

We could also write this query using join construct:

```
SELECT staffNo, fName, lName, position
FROM Staff s, Branch b
WHERE s.branchNo = b.branchNo AND
            city = ‘London’;
```

Example 3.36：Find the students who take all the courses and list their names.（相当于查询这样的学生，没有一门课是他不选的）

```
select sn   from s
where not exists
   ( select *  from c
     where not exists
          (select *  from sc
           where s.sno=sc.sno
             and sc.cno=c.cno));
```

S (1st level)

sno	sn	age	sex
s1	丁岩	19	M
s2	王爽	17	F
s3	李红	18	F
s4	赵立	21	M

C (2nd level)

cno	cn	T
c1	DB	li
c2	Maths	liu
c3	DS	zhang

SC (3rd level)

sno	cno	G
s1	c1	79
s1	c3	85
s2	c1	60
s2	c2	83
s2	c3	90
s3	c1	95
s3	c2	80
s4	c1	75
s4	c2	85

Result set is：

sno	sn
s2	王爽

Another solution?

```
Select s.sno, sname
From s, sc
Where s.sno=sc.sno
Group by s.sno, sname
Having count(cno) =
               (select count(*) from c)
```

Example 3.37：Find the students who take all the courses which S3 takes, and list their student numbers.

(相当于查询学号X，对所有课程Y，只要S3选修了课程Y，则学生X也选修了Y）

```
    select distinct x.sno
    from  sc as x   (Here the alias is necessary)
    where  not exists
      ( select *
        from sc as y
         where y.sno=‘s3’ and not exists
             （select *    from  sc as  z
                   where x.sno=z.sno
                       and z.cno=y.cno))；
```

x (1st level)

sno	cno	G
s1	c1	79
s1	c3	85
s2	c1	60
s2	c2	83
s2	c3	90
s3	c1	95
s3	c2	80
s4	c1	75
s4	c2	85

y (2nd level)

sno	cno	G
s1	c1	79
s1	c3	85
s2	c1	60
s2	c2	83
s2	c3	90
s3	c1	95
s3	c2	80
s4	c1	75
s4	c2	85

z (3rd level)

sno	cno	G
s1	c1	79
s1	c3	85
s2	c1	60
s2	c2	83
s2	c3	90
s3	c1	95
s3	c2	80
s4	c1	75
s4	c2	85

Another solution?

```
select sno
from sc
where cno in
      ( select cno from sc
        where sno='s3')
group by sno
having count(cno)= (> does not work)
      ( select count(cno)
        from sc
        where sno='s3');
```

Result set is:

sno
s2
s3
s4

where cno in
(select cno from sc
where sno='s3')

sc

sno	cno	G
s1	c1	79
s1	c3	85
s2	c1	60
s2	c2	83
s2	c3	90
s3	c1	95
s3	c2	80
s4	c1	75
s4	c2	85

After the above where clause

Filtered out

Filtered out

sno	cno	G
s1	c1	79
s2	c1	60
s2	c2	83
s3	c1	95
s3	c2	80
s4	c1	75
s4	c2	85

3.3.5 Union, Intersect, and Difference (Except)

- SQL can use normal set operations of Union, Intersection, and Difference to combine results of two or more queries into a single result table.
- Union of two tables, A and B, is a table containing all rows in either A or B or both.
- Intersection is a table containing all rows common to both A and B.
- Difference is a table containing all rows in A but not in B.

3.3.5 Union, Intersect, and Difference (Except)

- **Two tables must be *union compatible*.**
  - **The numbers of columns in two tables should be the same.**
  - **The corresponding columns should get values from the same domain.**

S1 (sno: **char(4)**, sname:char (10), age: smallint)
S2 (sno: **char(5)**, sname:char (10), age: smallint)

No!

S1 (sno: char(4), sname:char (10), age: smallint, **sex: char**)
S2 (sno: char(4), sname:char (10), age: smallint)

No!

S1 (sno: char(4), sname:char (10), age: smallint)
S2 (snum: char(4), name:char (10), s_age: smallint)

Yes!

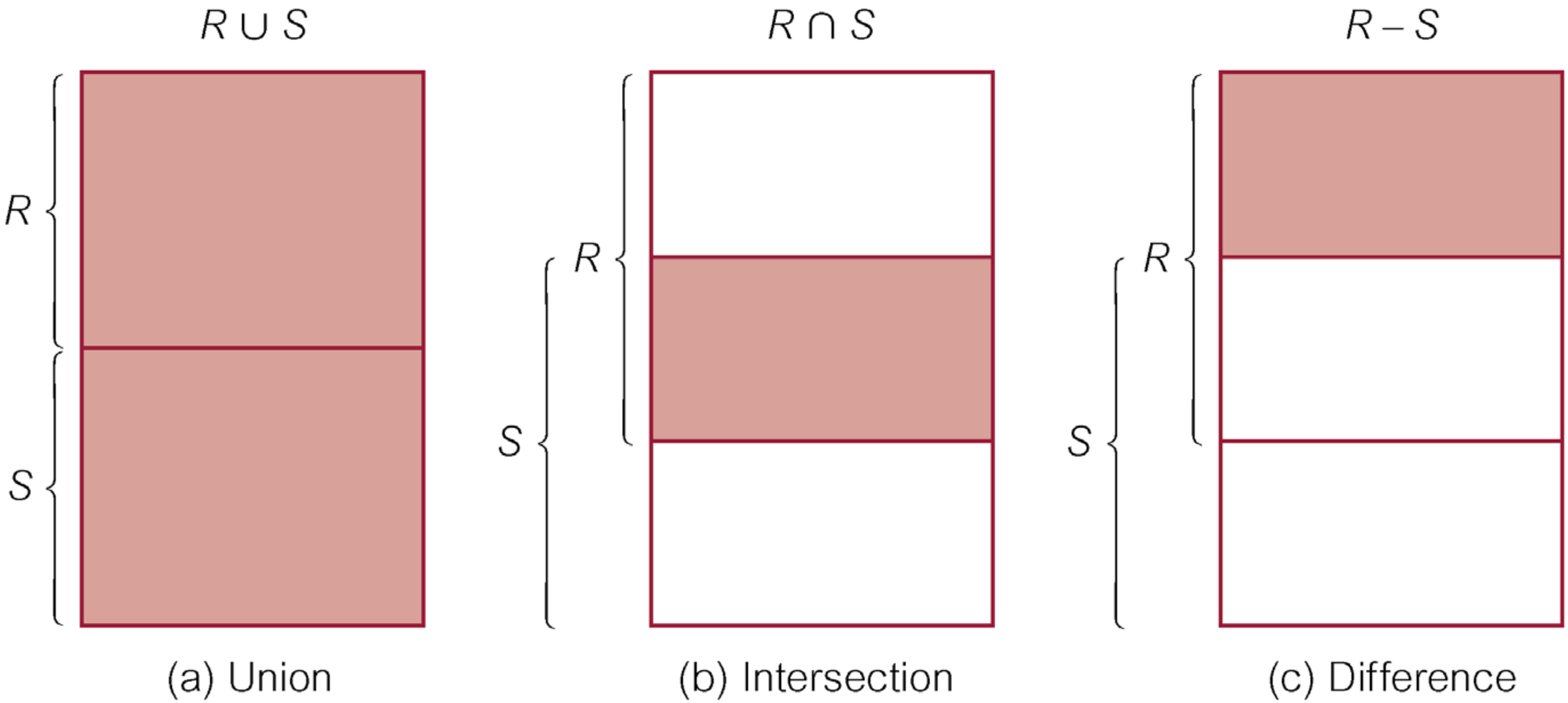
Union, Intersect, and Difference (Except)

- **Format of set operator clause in each case is:**

  ***op* [ALL] [CORRESPONDING [BY {column1 [, ...]}]]**

- **If CORRESPONDING BY specified, set operation performed on the named column(s).**
- **If CORRESPONDING specified but not BY clause, operation performed on common columns.**
- **If ALL specified, result can include duplicate rows.**

Union, Intersect, and Difference (Except)



(a) Union

(b) Intersection

(c) Difference

Example 3.38 Use of UNION

List all cities where there is either a branch office or a property.

```
(SELECT city
FROM Branch
WHERE city IS NOT NULL)
 UNION
(SELECT city
FROM  Property
WHERE city IS NOT NULL);
```

Example 3.38 Use of UNION

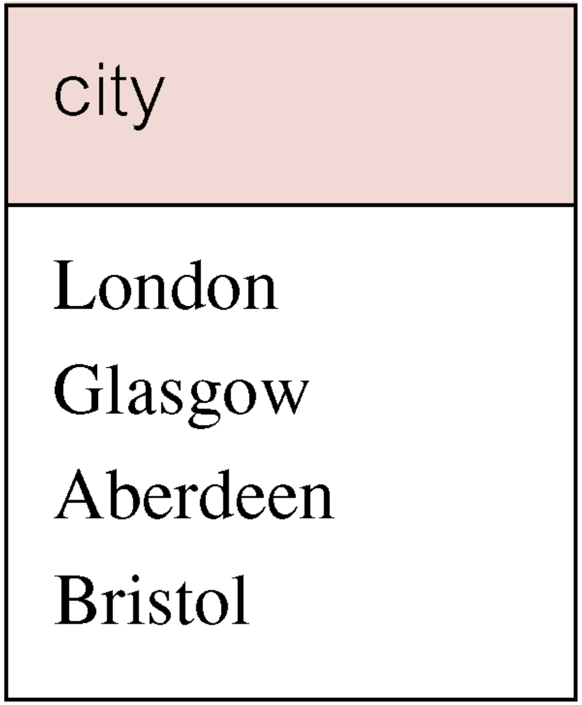
- Or

```
(SELECT *
 FROM Branch
 WHERE city IS NOT NULL)
 UNION CORRESPONDING BY city
 (SELECT *
 FROM Property
 WHERE city IS NOT NULL);
```

Example 3.38 Use of UNION

- **Produces result tables from both queries and merges both tables together.**

Table 5.32 Result table for Example 5.32.



city
London
Glasgow
Aberdeen
Bristol

Example 3.39 Use of INTERSECT

(SQL Server 2000 does not support,
but version 2008 does. DB2 dose.)

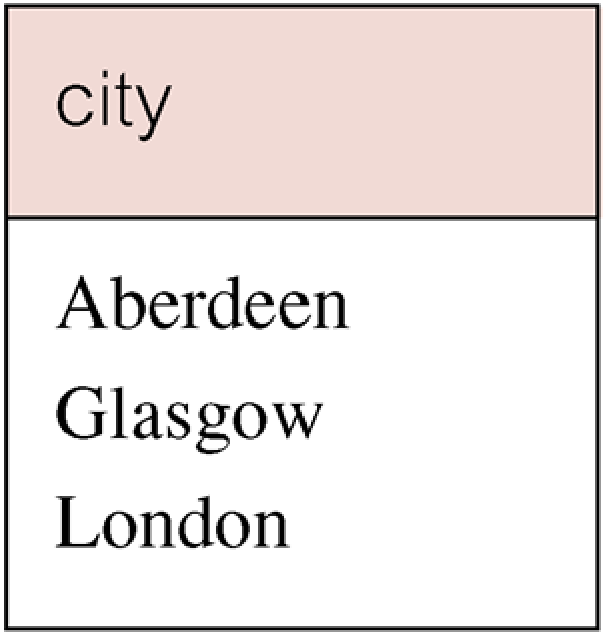
List all cities where there is both a branch office and a property.

```
 (SELECT city FROM Branch)
INTERSECT
(SELECT city FROM Property);
```

Example 3.39 Use of INTERSECT

- Or

```
(SELECT * FROM Branch)
INTERSECT CORRESPONDING BY city
(SELECT * FROM Property);
```



city
Aberdeen
Glasgow
London

Example 3.39 Use of INTERSECT

- Could rewrite this query without INTERSECT operator:

```
SELECT   b.city
FROM     Branch b, Property  p
WHERE    b.city = p.city;
```

- Or:

```
SELECT DISTINCT city FROM Branch b
WHERE EXISTS
  ( SELECT * FROM Property  p
    WHERE p.city = b.city );
```

Example 3.39 Use of INTERSECT

- Or

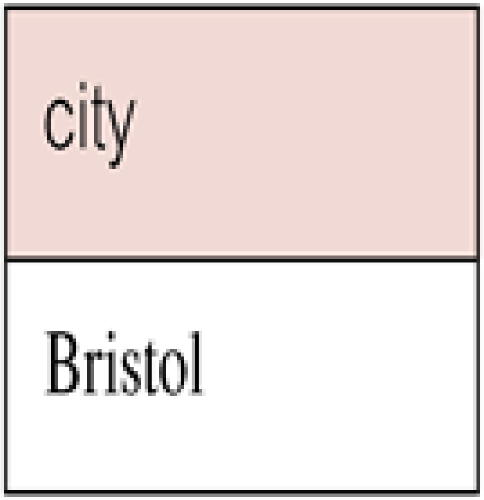
```
SELECT DISTINCT city
FROM Branch
WHERE city  IN
  (SELECT city FROM Property );
```

Example 3.40 Use of EXCEPT

(SQL Server 2000 does not support ,
but version 2008 does.
DB2 supports the first format.)

List of all cities where there is a branch office but no properties.

```
(SELECT city FROM Branch)
EXCEPT
(SELECT city FROM Property );
```



city
Bristol

- Or

```
(SELECT * FROM Branch)
EXCEPT CORRESPONDING BY city
(SELECT * FROM Property );
```

Example 3.40 Use of EXCEPT

- Could rewrite this query without EXCEPT:

```
SELECT DISTINCT city FROM Branch
WHERE city NOT IN
    (SELECT city FROM Property );
```

- Or

```
SELECT DISTINCT city FROM Branch b
WHERE NOT EXISTS
     ( SELECT * FROM Property  p
        WHERE p.city = b.city );
```