SELECT Statement - Grouping

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

SELECT Statement - Grouping

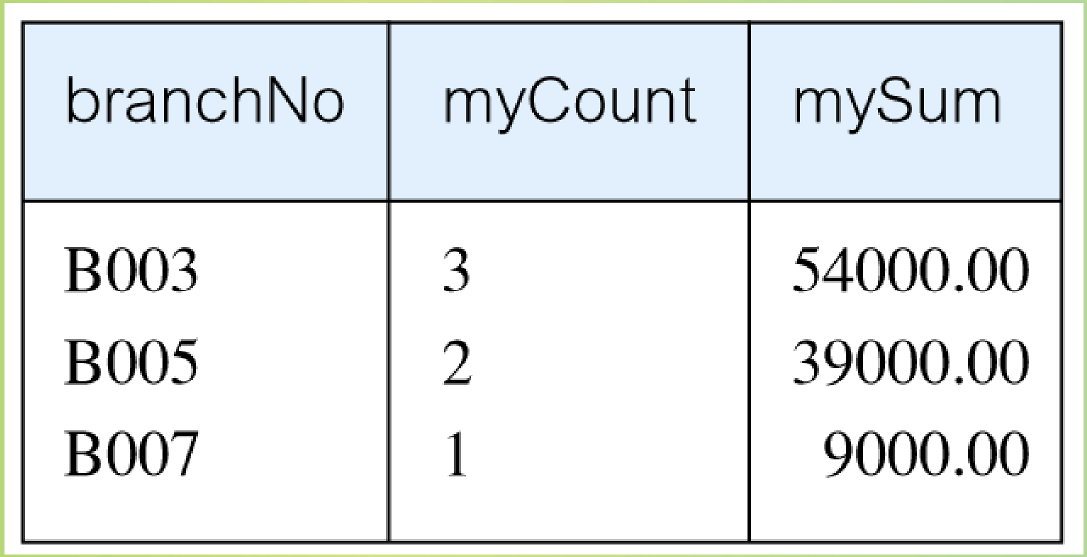
- All column names in SELECT list must appear in GROUP BY clause unless the name is used only in an aggregate function.
- If WHERE is used with GROUP BY, WHERE is applied first, then groups are formed from remaining rows satisfying predicate.
- ISO considers two nulls to be equal for purposes of GROUP BY.

Example 6.17 Use of GROUP BY

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

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

Example 6.17 Use of GROUP BY



branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00
B007	1	9000.00

Restricted Groupings – HAVING clause

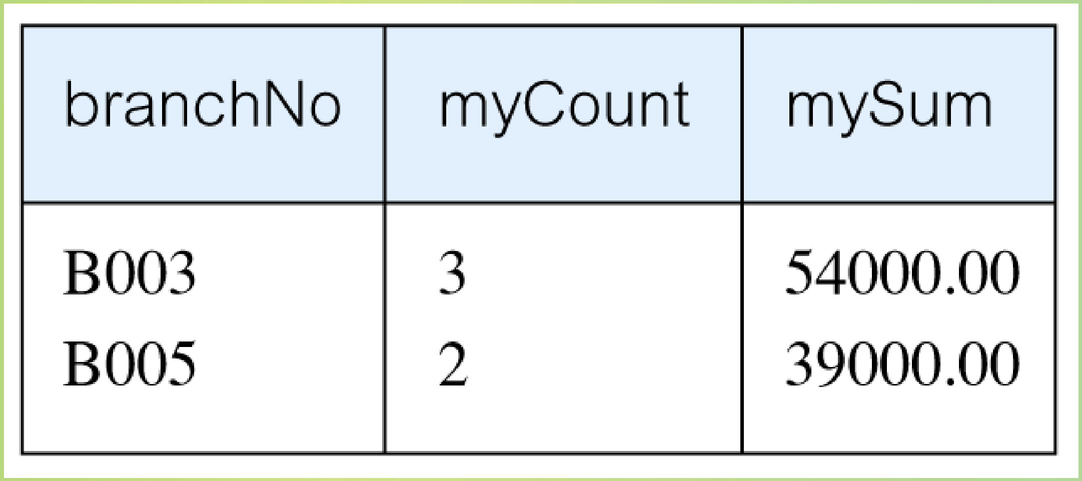
- **HAVING clause is designed for use with GROUP BY to restrict groups that appear in final result table.**
- **Similar to WHERE, but WHERE filters individual rows whereas HAVING filters groups.**
- **Column names in HAVING clause must also appear in the GROUP BY list or be contained within an aggregate function.**

Example 6.18 Use of HAVING

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

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

Example 6.18 Use of HAVING



branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00

Subqueries

- **Some SQL statements can have a SELECT 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*.**
- **Subselects may also appear in INSERT, UPDATE, and DELETE statements.**

Example 6.19 Subquery with Equality

List staff who work in branch at '163 Main St'.

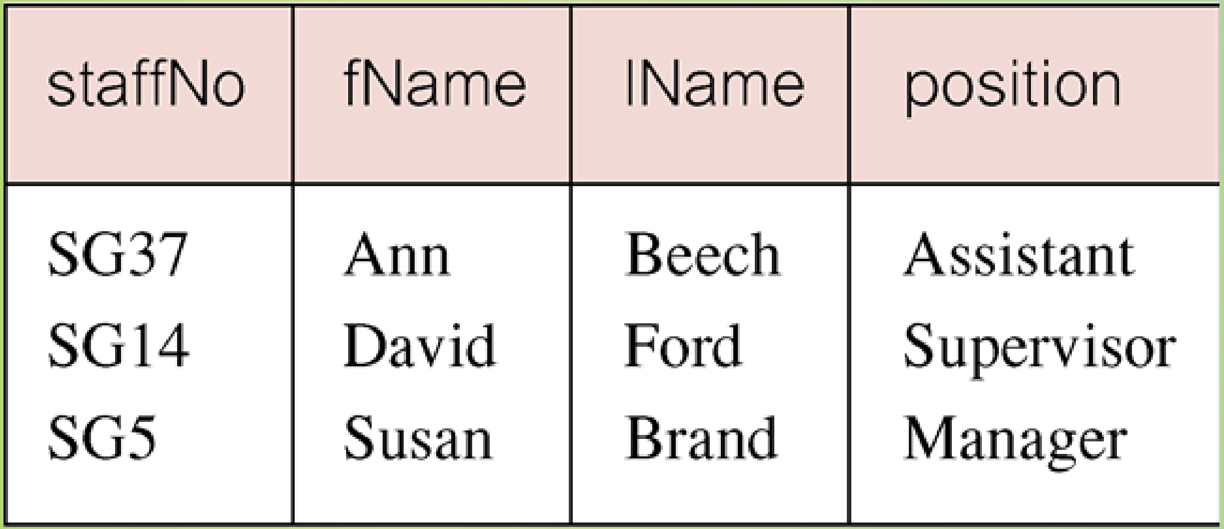
```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE branchNo =
                (SELECT branchNo
                 FROM Branch
                 WHERE street = '163 Main St');
```

Example 6.19 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 6.19 Subquery with Equality



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

Example 6.20 Subquery with Aggregate

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

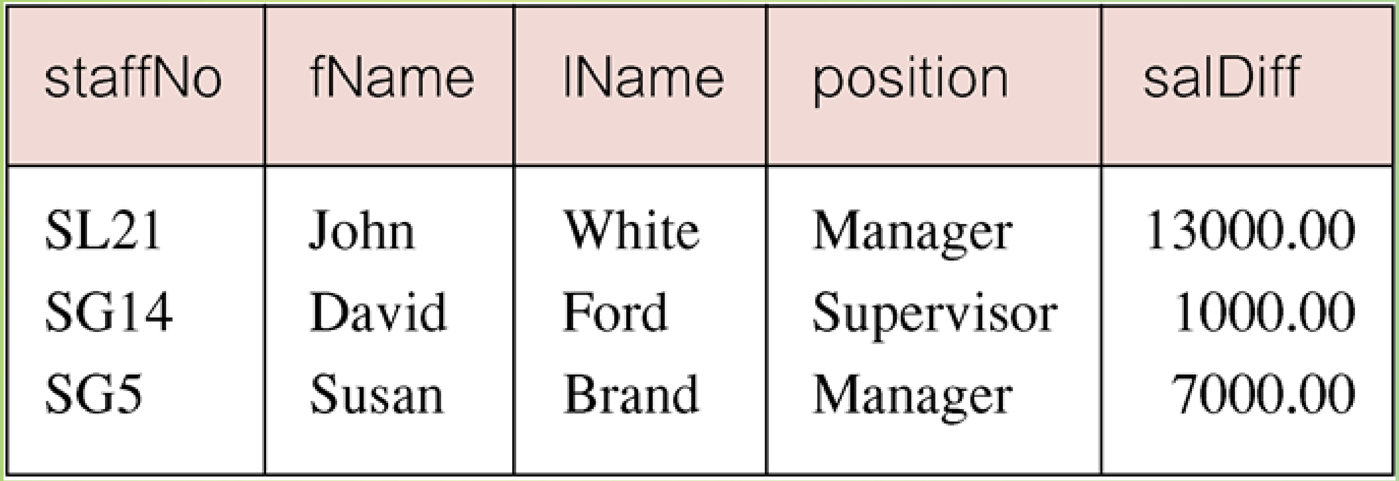
```
SELECT staffNo, fName, lName, position,
  salary – (SELECT AVG(salary) FROM Staff) As SalDiff
FROM Staff
WHERE salary >
                (SELECT AVG(salary)
                 FROM Staff);
```

Example 6.20 Subquery with Aggregate

- Cannot write ‘WHERE salary > AVG(salary)’
- Instead, use subquery to find 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;
```

Example 6.20 Subquery with Aggregate



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

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 in a subquery refer to the table name in the FROM clause of the subquery.

Subquery Rules

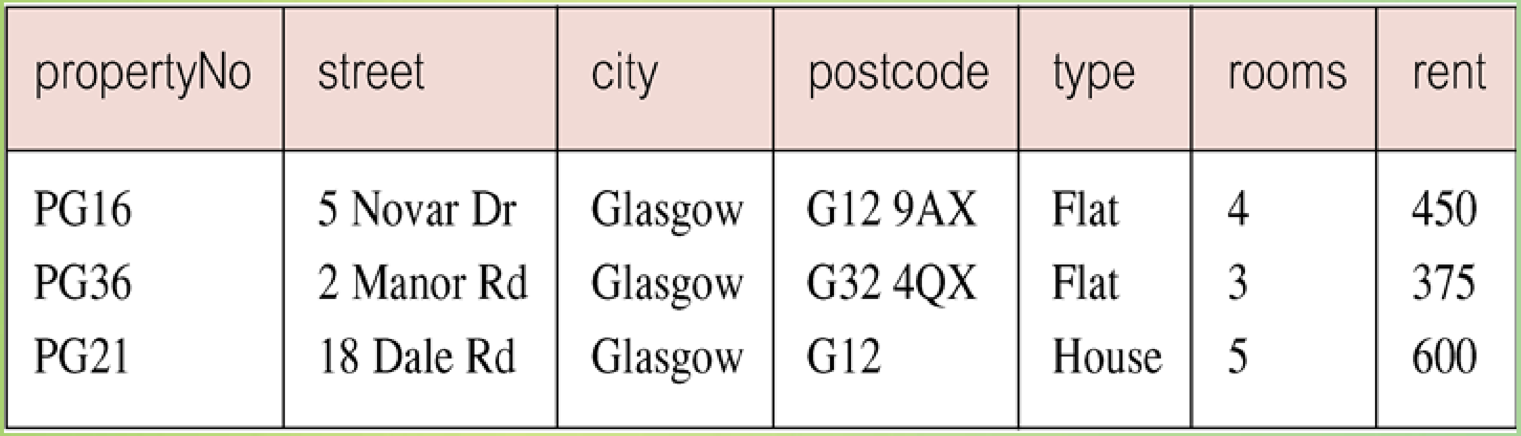
- When subquery is an operand in a comparison, subquery must appear on right-hand side.
- A subquery may not be used as an operand in an expression.

Example 6.21 Nested subquery: use of IN

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

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

Example 6.21 Nested subquery: use of IN



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

ANY and ALL

- ANY and ALL may be used with subqueries that produce a single column of numbers.
- 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.

Example 6.22 Use of ANY/SOME

Find staff whose salary is larger than 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');
```

Example 6.22 Use of ANY/SOME

- **Inner query produces set {12000, 18000, 24000} and outer query selects those staff whose salaries are greater than any of the values in this 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 6.23 Use of ALL

Find 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');
```

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00

Multi-Table Queries

- Subqueries provide result columns that come from same table.
- If result columns come from more than one table must use a join.
- To perform join, include more than one table in FROM clause.
- Use comma as separator and typically include WHERE clause to specify join column(s).

Multi-Table Queries

- **Also possible to use an alias for a table named in FROM clause.**
- **Alias is separated from table name with a space.**
- **Alias can be used to qualify column names when there is ambiguity.**

Example 6.24 Simple Join

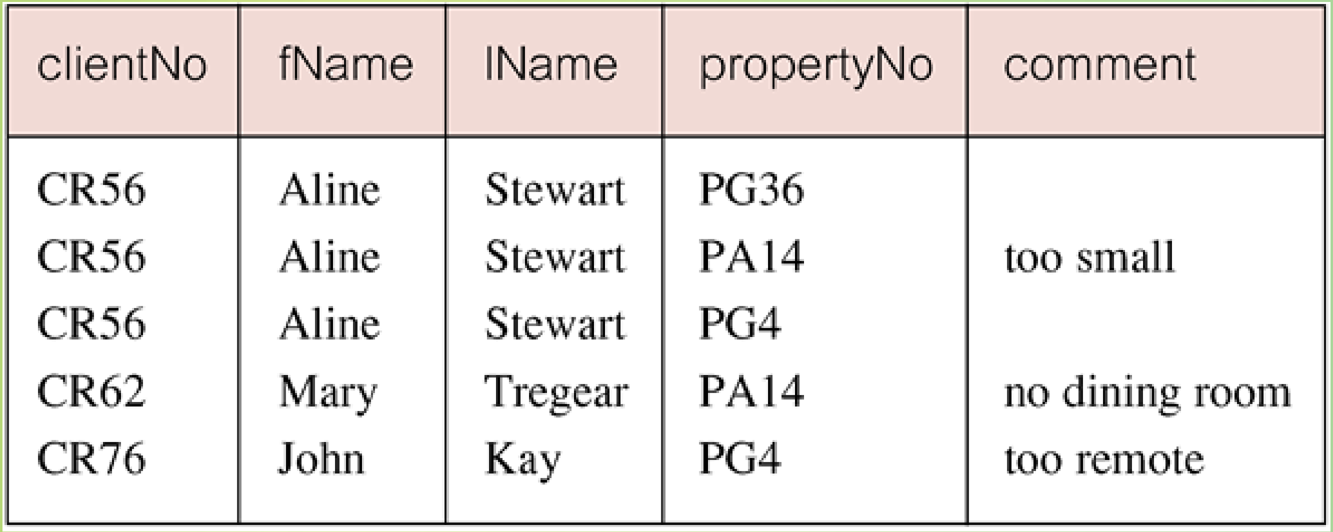
List names of all clients who have viewed a property along with any comment supplied.

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

Example 6.24 Simple Join

- **Only those rows from both tables that have identical values in the clientNo columns (c.clientNo = v.clientNo) are included in result.**

- **Equivalent to equi-join in relational algebra.**



clientNo	fName	lName	propertyNo	comment
CR56	Aline	Stewart	PG36	
CR56	Aline	Stewart	PA14	too small
CR56	Aline	Stewart	PG4	
CR62	Mary	Tregear	PA14	no dining room
CR76	John	Kay	PG4	too remote

Alternative JOIN Constructs

- **SQL provides alternative ways to specify joins:**

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

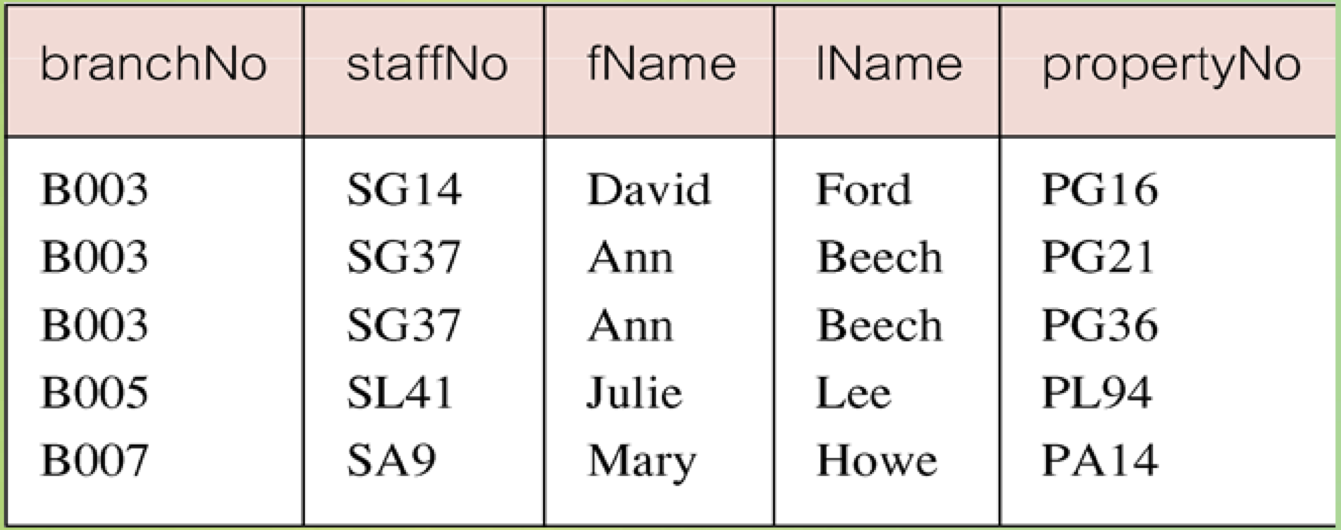
- **In each case, FROM replaces original FROM and WHERE. However, first produces table with two identical clientNo columns.**

Example 6.25 Sorting a join

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

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

Example 6.25 Sorting a join



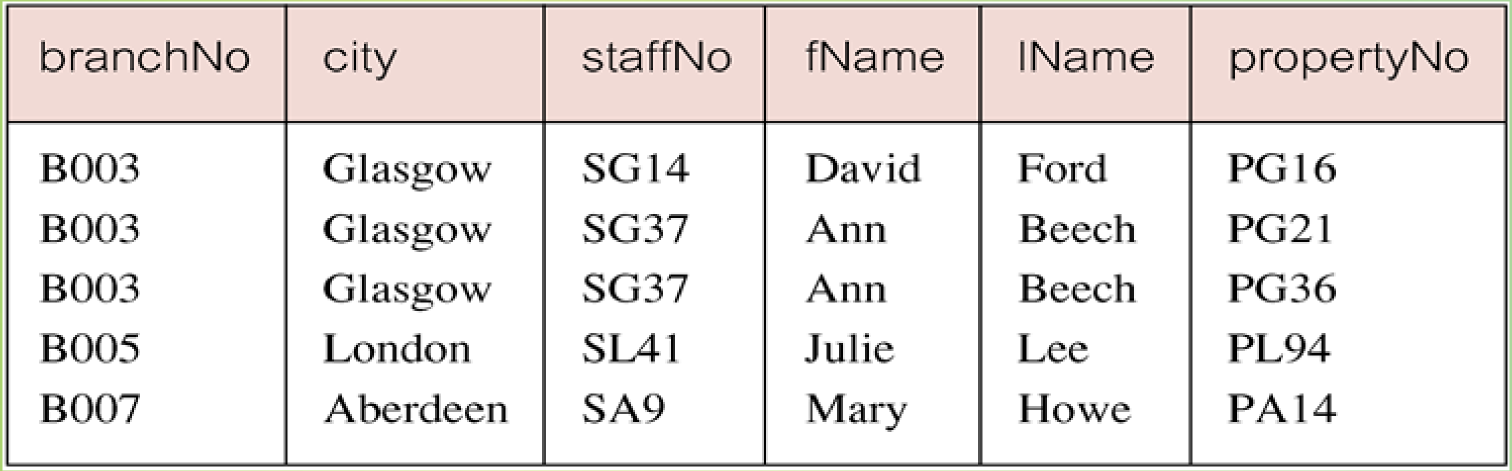
branchNo	staffNo	fName	lName	propertyNo
B003	SG14	David	Ford	PG16
B003	SG37	Ann	Beech	PG21
B003	SG37	Ann	Beech	PG36
B005	SL41	Julie	Lee	PL94
B007	SA9	Mary	Howe	PA14

Example 6.26 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, PropertyForRent p
 WHERE b.branchNo = s.branchNo AND
                s.staffNo = p.staffNo
 ORDER BY b.branchNo, s.staffNo, propertyNo;
```

Example 6.26 Three Table Join



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

- **Alternative formulation for FROM and WHERE:**

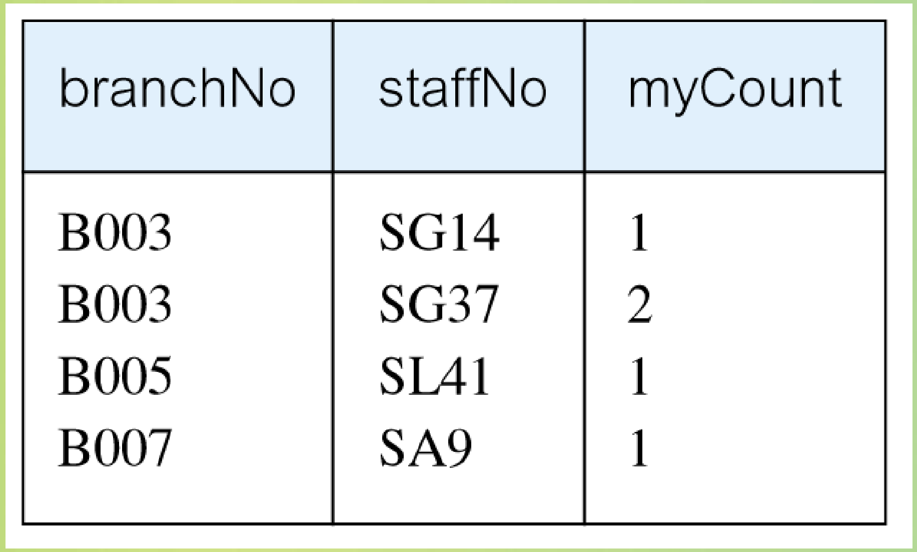
FROM (Branch b JOIN Staff s USING branchNo) AS bs JOIN PropertyForRent p USING staffNo

Example 6.27 Multiple Grouping Columns

Find number of properties handled by each staff member, along with the branch number of the member of staff.

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

Example 6.27 Multiple Grouping Columns



branchNo	staffNo	myCount
B003	SG14	1
B003	SG37	2
B005	SL41	1
B007	SA9	1

Computing a Join

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 value of each item in SELECT list to produce a single row in result table.**

Computing a Join

4. If DISTINCT has been specified, eliminate any duplicate rows from the result table.

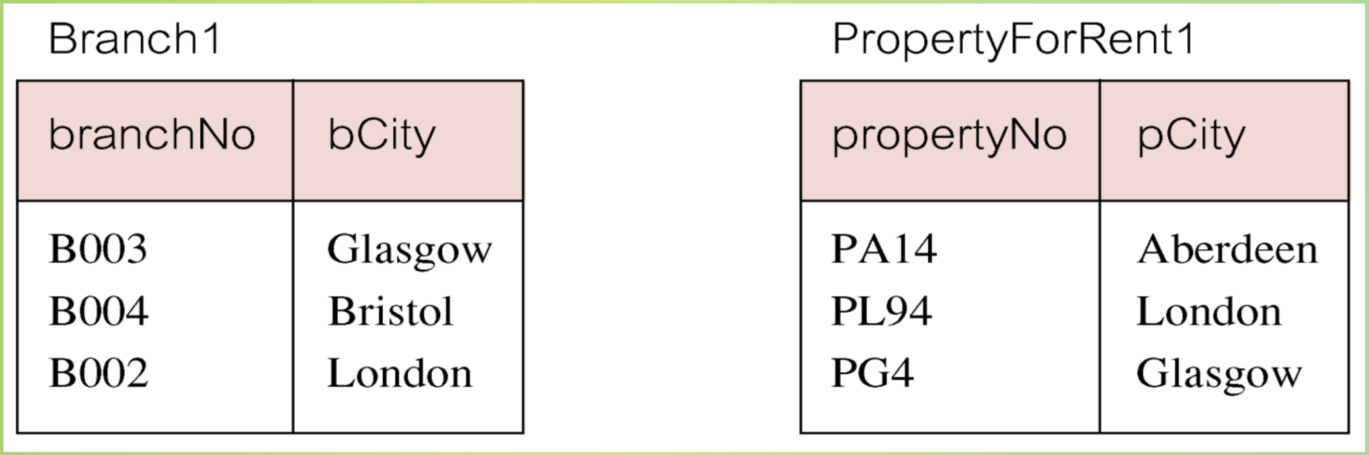
5. If there is an ORDER BY clause, sort result table as required.

- SQL provides special format of SELECT for Cartesian product:

```
SELECT    [DISTINCT | ALL]      {* | columnList}
FROM Table1Name CROSS JOIN Table2Name
```

Outer Joins

- If one row of a joined table is unmatched, row is omitted from result table.
- Outer join operations retain rows that do not satisfy the join condition.
- Consider following tables:



Branch1

branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London

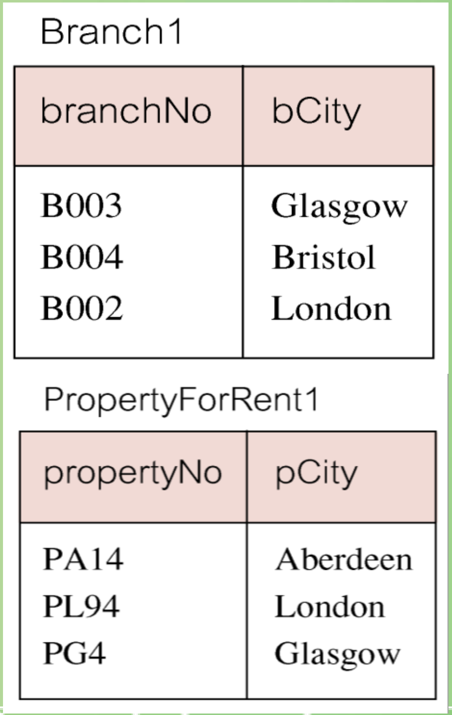
PropertyForRent1

propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow

Outer Joins

- The (inner) join of these two tables:

```
SELECT b.*, p.*
FROM Branch1 b, PropertyForRent1 p
WHERE b.bCity = p.pCity;
```

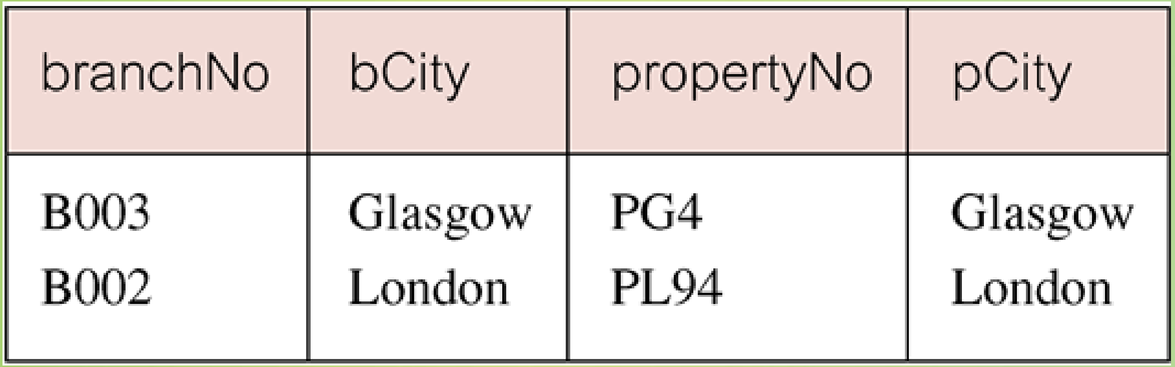


Branch1

branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London

PropertyForRent1

propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow



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

Outer Joins

- **Result table has two rows where cities are same.**
- **There are no rows corresponding to branches in Bristol and Aberdeen.**
- **To include unmatched rows in result table, use an Outer join.**

Example 6.28 Left Outer Join

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

```
SELECT b.*, p.*
FROM Branch1 b LEFT JOIN
                PropertyForRent1 p ON b.bCity = p.pCity;
```

Example 6.28 Left Outer Join

- Includes those rows of first (left) table unmatched with rows from second (right) table.
- Columns from second table are filled with NULLs.

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

Branch1

branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London

PropertyForRent1

propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow

Example 6.29 Right Outer Join

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

```
SELECT b.*, p.*
FROM Branch1 b RIGHT JOIN
        PropertyForRent1 p ON b.bCity = p.pCity;
```

Example 6.29 Right Outer Join

- Right Outer join includes those rows of second (right) table that are unmatched with rows from first (left) table.
- Columns from first table are filled with NULLs.

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

Branch1

branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London

PropertyForRent1

propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow

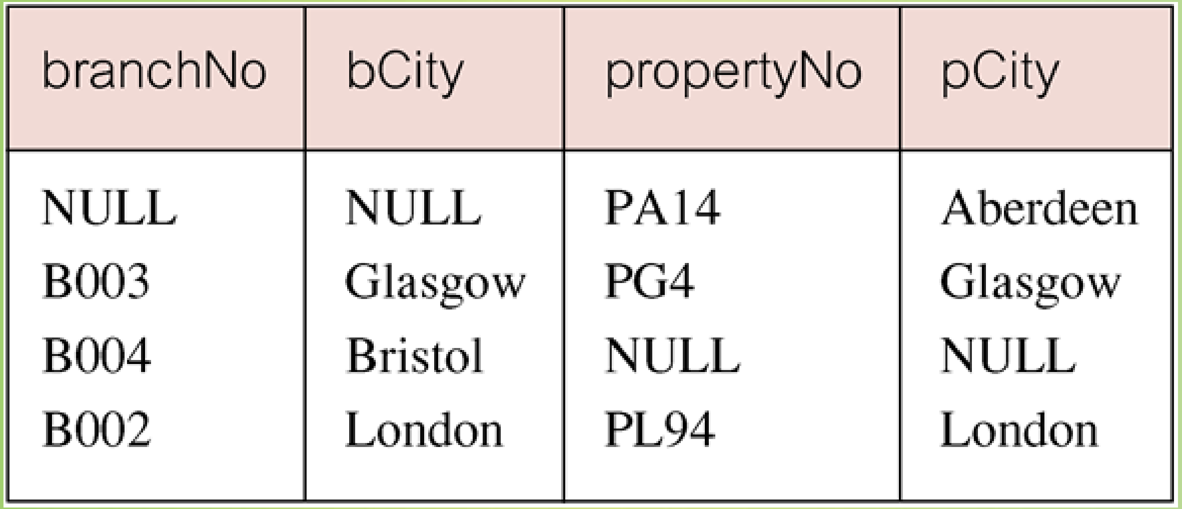
Example 6.30 Full Outer Join

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

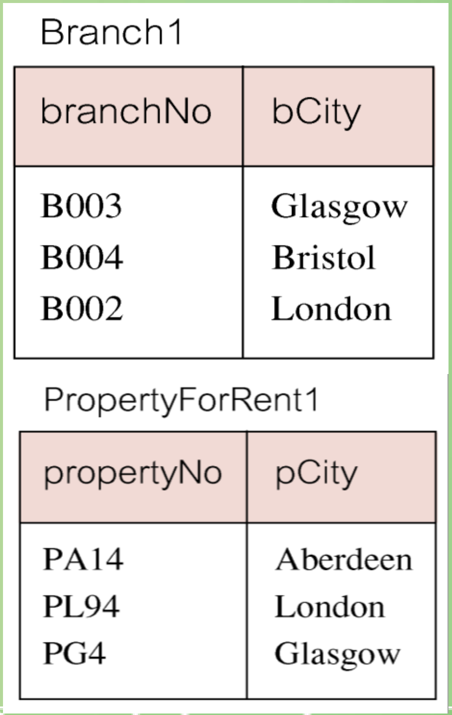
```
SELECT b.*, p.*
FROM Branch1 b FULL JOIN
        PropertyForRent1 p ON b.bCity = p.pCity;
```

Example 6.30 Full Outer Join

- **Includes rows that are unmatched in both tables.**
- **Unmatched columns are filled with NULLs.**



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



Branch1

branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London

PropertyForRent1

propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow

EXISTS and NOT EXISTS

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

EXISTS and NOT EXISTS

- **As (NOT) EXISTS check only for existence or non-existence of rows in subquery result table, subquery can contain any number of columns.**

- **Common for subqueries following (NOT) EXISTS to be of form:**

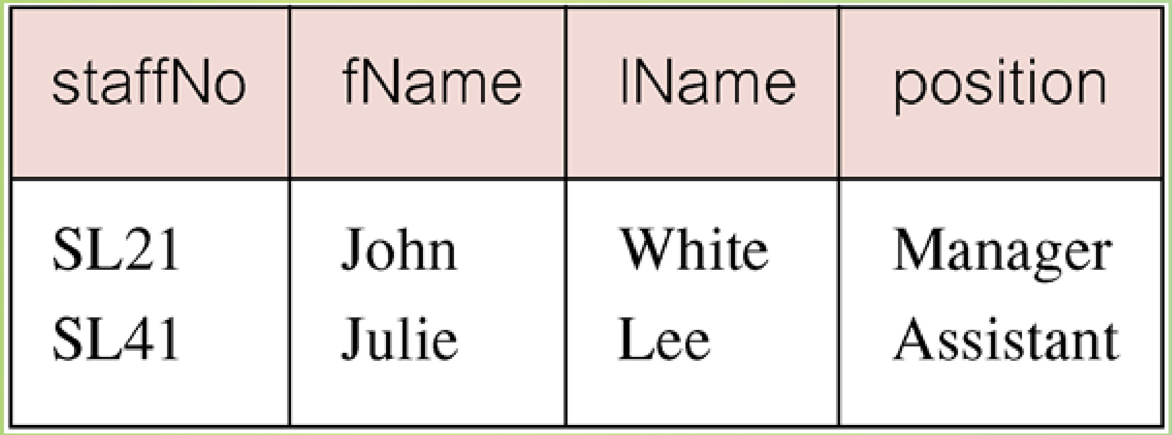
```
(SELECT * FROM ...)
```

Example 6.31 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’);
```

Example 6.31 Query using EXISTS



staffNo	fName	lName	position
SL21	John	White	Manager
SL41	Julie	Lee	Assistant

Example 6.31 Query using EXISTS

- Note, search condition s.branchNo = b.branchNo is necessary to consider correct branch record for each member of staff.
- If omitted, 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 6.31 Query using EXISTS

- 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’;
```

Union, Intersect, and Difference (Except)

- 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 table containing all rows in either A or B or both.
- Intersection is table containing all rows common to both A and B.
- Difference is table containing all rows in A but not in B.
- Two tables must be *union compatible*.

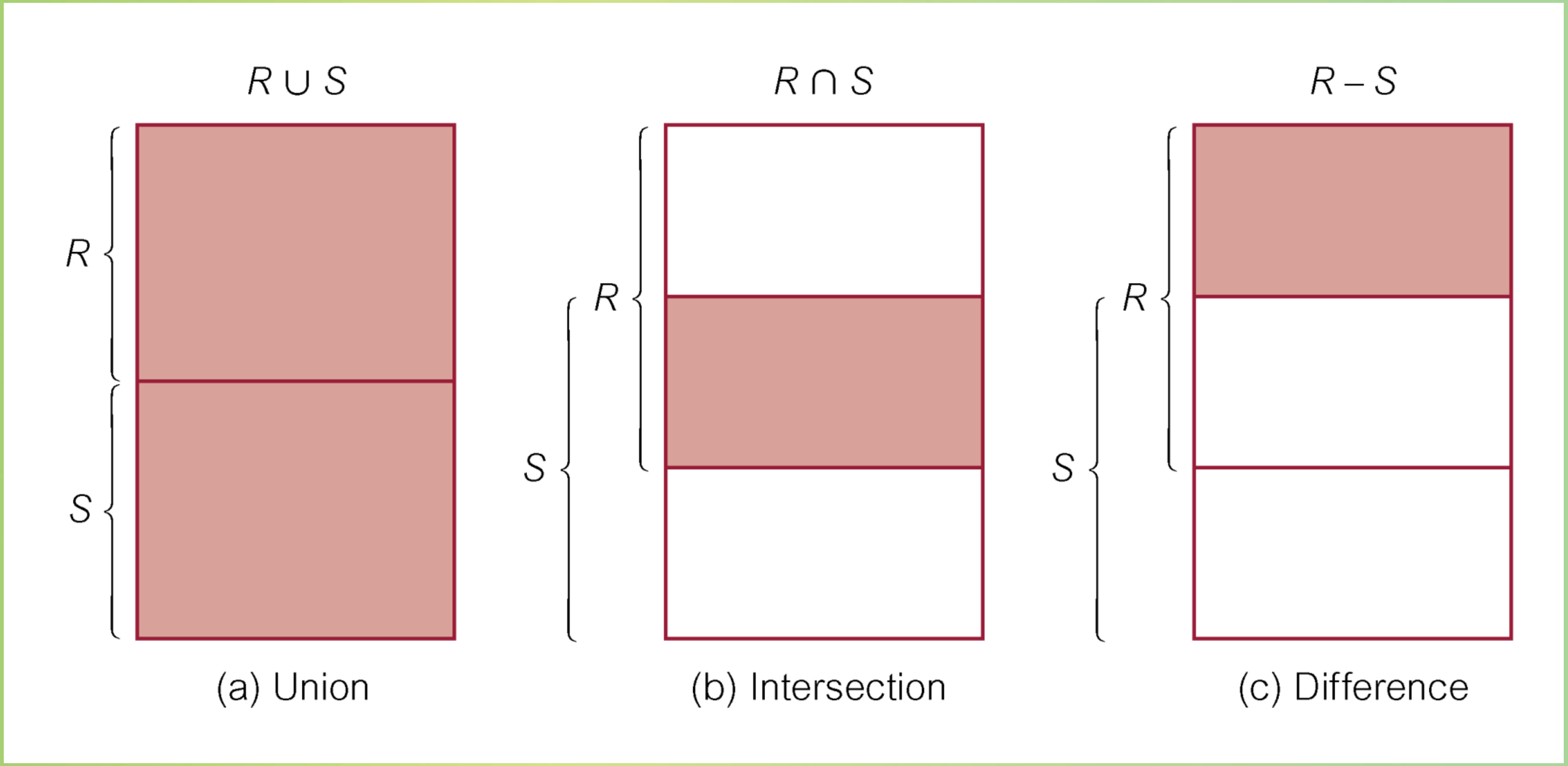
Union, Intersect, and Difference (Except)

- Format of set operator clause in each case is:

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

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

Union, Intersect, and Difference (Except)



Example 6.32 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 PropertyForRent
WHERE city IS NOT NULL);
```

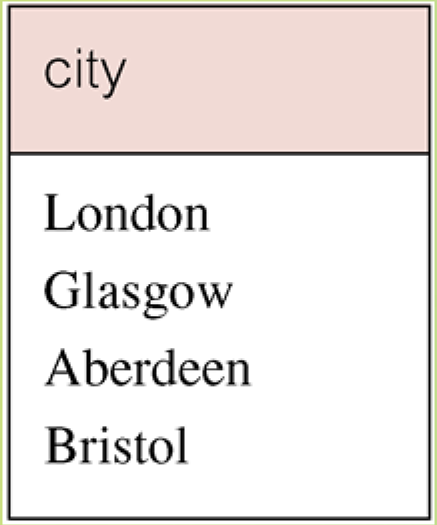
Example 6.32 Use of UNION

- Or

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

Example 6.32 Use of UNION

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



city
London
Glasgow
Aberdeen
Bristol

Example 6.33 Use of INTERSECT

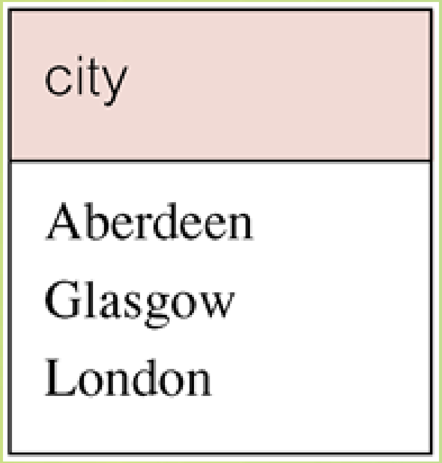
List all cities where there is both a branch office and a property.

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

Example 6.33 Use of INTERSECT

- Or

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



city
Aberdeen
Glasgow
London

Example 6.33 Use of INTERSECT

- Could rewrite this query without INTERSECT operator:

```
SELECT b.city
FROM Branch b PropertyForRent p
WHERE b.city = p.city;
```

- Or:

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

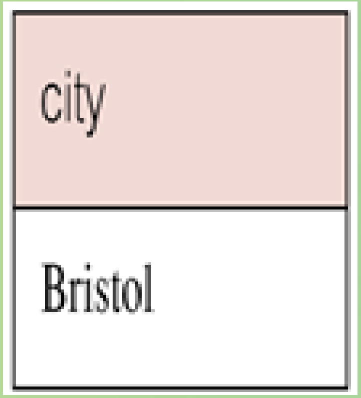
Example 6.34 Use of EXCEPT

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

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

- Or

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



city
Bristol

Example 6.34 Use of EXCEPT

- Could rewrite this query without EXCEPT:

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

- Or

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

Database Updates

- INSERT – adds new rows of data to a table
- UPDATE – modifies existing data in a table
- DELETE – removes rows of data from a table

INSERT

```
INSERT INTO TableName [ (columnList) ]
 VALUES (dataValueList)
```

- *columnList* is optional; if omitted, SQL assumes a list of all columns in their original CREATE TABLE order.
- Any columns omitted must have been declared as NULL when table was created, unless DEFAULT was specified when creating column.

INSERT

- ***dataValueList*** **must match *columnList* as follows:**
  - **number of items in each list must be same;**
  - **must be direct correspondence in position of items in two lists;**
  - **data type of each item in *dataValueList* must be compatible with data type of corresponding column.**

Example 6.35 INSERT … VALUES

Insert a new row into Staff table supplying data for all columns.

```
INSERT INTO Staff
VALUES ('SG16', 'Alan', 'Brown', 'Assistant', 'M',
    Date'1957-05-25', 8300, 'B003');
```

Example 6.36 INSERT using Defaults

Insert a new row into the Staff table supplying data for all mandatory columns: staffNo, fName, lName, position, salary, and branchNo.

```
INSERT INTO Staff (staffNo, fName, lName,
                                    position, salary, branchNo)
VALUES ('SG44', 'Anne', 'Jones',
                 'Assistant', 8100, 'B003');
```

Or

```
INSERT INTO Staff
VALUES ('SG44', 'Anne', 'Jones', 'Assistant', NULL,
                  NULL, 8100, 'B003');
```

INSERT … SELECT

- **Second form of INSERT allows multiple rows to be copied from one or more tables to another:**

```
INSERT INTO TableName [ (columnList) ]
    SELECT ...
```

Example 6.37 INSERT ... SELECT

Assume there is a table StaffPropCount that contains names of staff and number of properties they manage:

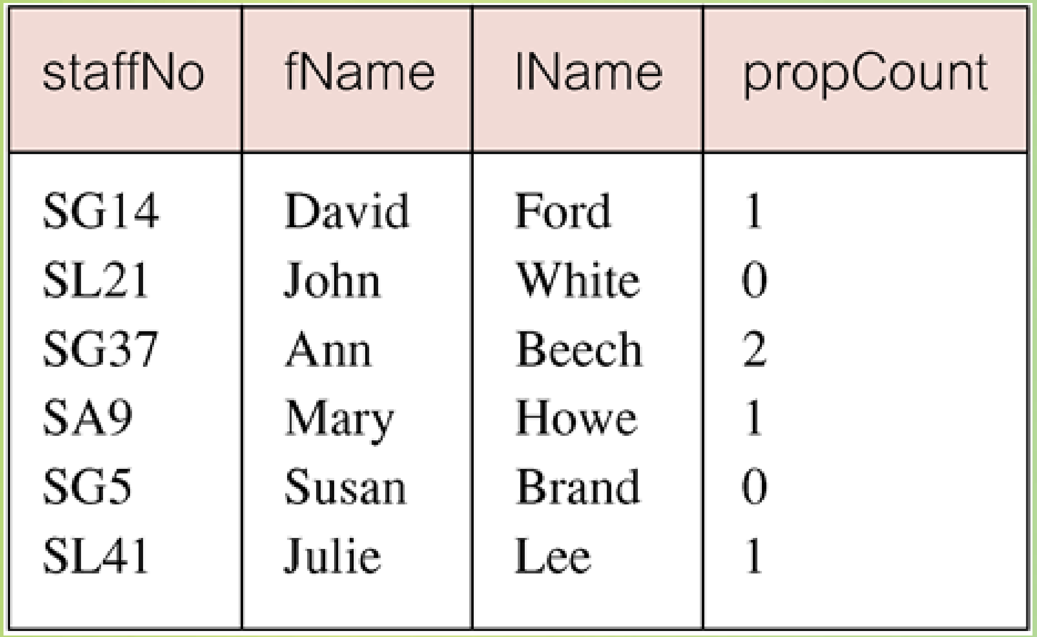
StaffPropCount(<u>staffNo</u>, fName, lName, propCnt)

Populate StaffPropCount using Staff and PropertyForRent tables.

Example 6.37 INSERT … SELECT

```
INSERT INTO StaffPropCount
    (SELECT s.staffNo, fName, lName, COUNT(*)
    FROM Staff s, PropertyForRent p
    WHERE s.staffNo = p.staffNo
    GROUP BY s.staffNo, fName, lName)
    UNION
    (SELECT staffNo, fName, lName, 0
    FROM Staff
    WHERE staffNo NOT IN
            (SELECT DISTINCT staffNo
            FROM PropertyForRent));
```

Example 6.37 INSERT … SELECT



staffNo	fName	lName	propCount
SG14	David	Ford	1
SL21	John	White	0
SG37	Ann	Beech	2
SA9	Mary	Howe	1
SG5	Susan	Brand	0
SL41	Julie	Lee	1

- **If second part of UNION is omitted, excludes those staff who currently do not manage any properties.**

UPDATE

```
UPDATE TableName
SET columnName1 = dataValue1
                [, columnName2 = dataValue2...]
[WHERE searchCondition]
```

- *TableName* can be name of a base table or an updatable view.
- SET clause specifies names of one or more columns that are to be updated.

UPDATE

- WHERE clause is optional:
  - if omitted, named columns are updated for all rows in table;
  - if specified, only those rows that satisfy *searchCondition* are updated.
- New *dataValue(s)* must be compatible with data type for corresponding column.

Example 6.38 UPDATE All Rows

Give all staff a 3% pay increase.

```
UPDATE Staff
SET salary = salary*1.03;
```

Example 6.39 UPDATE All Rows

Give all Managers a 5% pay increase.

```
UPDATE Staff
SET salary = salary*1.05
WHERE position = 'Manager';
```

Example 6.40 UPDATE Multiple Columns

Promote David Ford (staffNo='SG14') to Manager and change his salary to £18,000.

```
UPDATE Staff
SET position = 'Manager', salary = 18000
WHERE staffNo = 'SG14';
```

DELETE

```
DELETE FROM TableName
[WHERE searchCondition]
```

- ***TableName*** **can be name of a base table or an updatable view.**
- ***searchCondition*** **is optional; if omitted, all rows are deleted from table. This does not delete table. If** ***search_condition*** **is specified, only those rows that satisfy condition are deleted.**

Example 6.41 DELETE Specific Rows

Delete all viewings that relate to property PG4.

```
DELETE FROM Viewing
WHERE propertyNo = 'PG4';
```

Example 6.42 DELETE Specific Rows

Delete all records from the Viewing table.

```
DELETE FROM Viewing;
```