Database Systems

Instructor: Bilal Khalid Dar

Agenda

- Purpose and importance of SQL.
- How to retrieve data from database using SELECT

and:

  - Use compound WHERE conditions.
  - Sort query results using ORDER BY.
  - Use aggregate functions.

Objectives of SQL

Ideally, database language should allow user to:
– create the database and relation structures;

– perform insertion, modification, deletion of data from relations;

– perform simple and complex queries.

Must perform these tasks with minimal user effort and command structure/syntax must be easy to learn.

It must be portable.

Objectives of SQL

SQL is a transform-oriented language with 2 major components:

– A DDL for defining database structure.

– A DML for retrieving and updating data.

Until SQL:1999, SQL did not contain flow of control commands. These had to be implemented using a programming or job-control language, or interactively by the decisions of user.

Objectives of SQL

SQL is relatively easy to learn:

– it is non-procedural - you specify *what* information you require, rather than *how* to get it;

– it is essentially free-format

SQL Data Definition and Data Types

- Terminology:
  - **Table**, **row**, and **column** used for relational model terms relation, tuple, and attribute
- `CREATE` statement
  - Main SQL command for data definition

Attribute Data Types and Domains in SQL

- Basic **data types**
  - **Numeric** data types
    - Integer numbers: `INTEGER`, `INT`, and `SMALLINT`
    - Floating-point (real) numbers: `FLOAT` or `REAL`, and `DOUBLE PRECISION`
  - **Character-string** data types
    - Fixed length: `CHAR(`*n*`)`, `CHARACTER(`*n*`)`
    - Varying length : `VARCHAR(`*n*`)`, `CHAR VARYING(`*n*`)`, `CHARACTER VARYING(`*n*`)`

Attribute Data Types and Domains in SQL (cont’d.)

- **Bit-string** data types
  - Fixed length: `BIT(`*n*`)`
  - Varying length: `BIT VARYING(`*n*`)`
- **Boolean** data type
  - Values of `TRUE` or `FALSE` or `NULL`
- **DATE** data type
  - Ten positions
  - Components are `YEAR`, `MONTH`, and `DAY` in the form YYYY-MM-DD

Attribute Data Types and Domains in SQL (cont'd.)

- Additional data types
  - **Timestamp** data type (`TIMESTAMP`)
    - Includes the `DATE` and `TIME` fields
    - Plus a minimum of six positions for decimal fractions of seconds
    - Optional `WITH TIME ZONE` qualifier
  - **`INTERVAL`** data type
    - Specifies a relative value that can be used to increment or decrement an absolute value of a date, time, or timestamp

Attribute Data Types and Domains in SQL (cont’d.)

- Domain
  - Name used with the attribute specification
  - Makes it easier to change the data type for a domain that is used by numerous attributes
  - Improves schema readability
  - Example:
    - `CREATE DOMAIN SSN_TYPE AS CHAR(9);`

Specifying Constraints in SQL

- Basic constraints:
  - Key and referential integrity constraints
  - Restrictions on attribute domains and NULLs
  - Constraints on individual tuples within a relation

Objectives of SQL

Consists of standard English words:

```
1)   CREATE TABLE Staff(staffNo VARCHAR(5),
lName VARCHAR(15),
salary DECIMAL(7,2));

2) INSERT INTO Staff VALUES (‘SG16’, ‘Brown’, 8300);

3) SELECT staffNo, lName, salary
FROM Staff
WHERE salary > 10000;
```

History of SQL

In 1974, D. Chamberlin (IBM San Jose Laboratory) defined language called ‘Structured English Query Language’ (SEQUEL).

A revised version, SEQUEL/2, was defined in 1976 but name was subsequently changed to SQL for legal reasons.

History of SQL

In late 70s, ORACLE appeared and was probably first commercial RDBMS based on SQL.

In 1987, ANSI and ISO published an initial standard for SQL.

In 1989, ISO published an addendum that defined an ‘Integrity Enhancement Feature’.

In 1992, first major revision to ISO standard occurred, referred to as SQL2 or SQL/92.

In 1999, SQL:1999 was released with support for object-oriented data management.

In late 2003, SQL:2003 was released

Importance of SQL

SQL has become part of application architectures such as IBM’s Systems Application Architecture.

It is strategic choice of many large and influential organizations (e.g. X/OPEN).

SQL is Federal Information Processing Standard (FIPS) to which conformance is required for all sales of databases to American Government.

History of SQL

Still pronounced ‘see-quel’, though official pronunciation is ‘S-Q-L’.

IBM subsequently produced a prototype DBMS called *System R*, based on SEQUEL/2.

Roots of SQL, however, are in SQUARE (Specifying Queries as Relational Expressions), which predates System R project.

Importance of SQL

SQL is used in other standards and even influences development of other standards as a definitional tool. Examples include:

– ISO’s Information Resource Directory System (IRDS) Standard
– Remote Data Access (RDA) Standard.

Writing SQL Commands

SQL statement consists of *reserved words* and *userdefined words*.

– **Reserved** words are a fixed part of SQL and must be spelt exactly as required and cannot be split across lines.

– **User-defined** words are made up by user and represent names of various database objects such as relations, columns, views.

Writing SQL Commands

Most components of an SQL statement are *case insensitive*, except for literal character data.

More readable with indentation and lineation:
– Each clause should begin on a new line.
– Start of a clause should line up with start of other clauses.
– If clause has several parts, should each appear on a separate line and be indented under start of clause.

Writing SQL Commands

Use extended form of BNF notation:

- Upper-case letters represent reserved words.
- Lower-case letters represent user-defined words.
- | indicates a *choice* among alternatives.
- Curly braces indicate a *required element*.
- Square brackets indicate an *optional element*.
- … indicates *optional repetition* (0 or more).

Literals

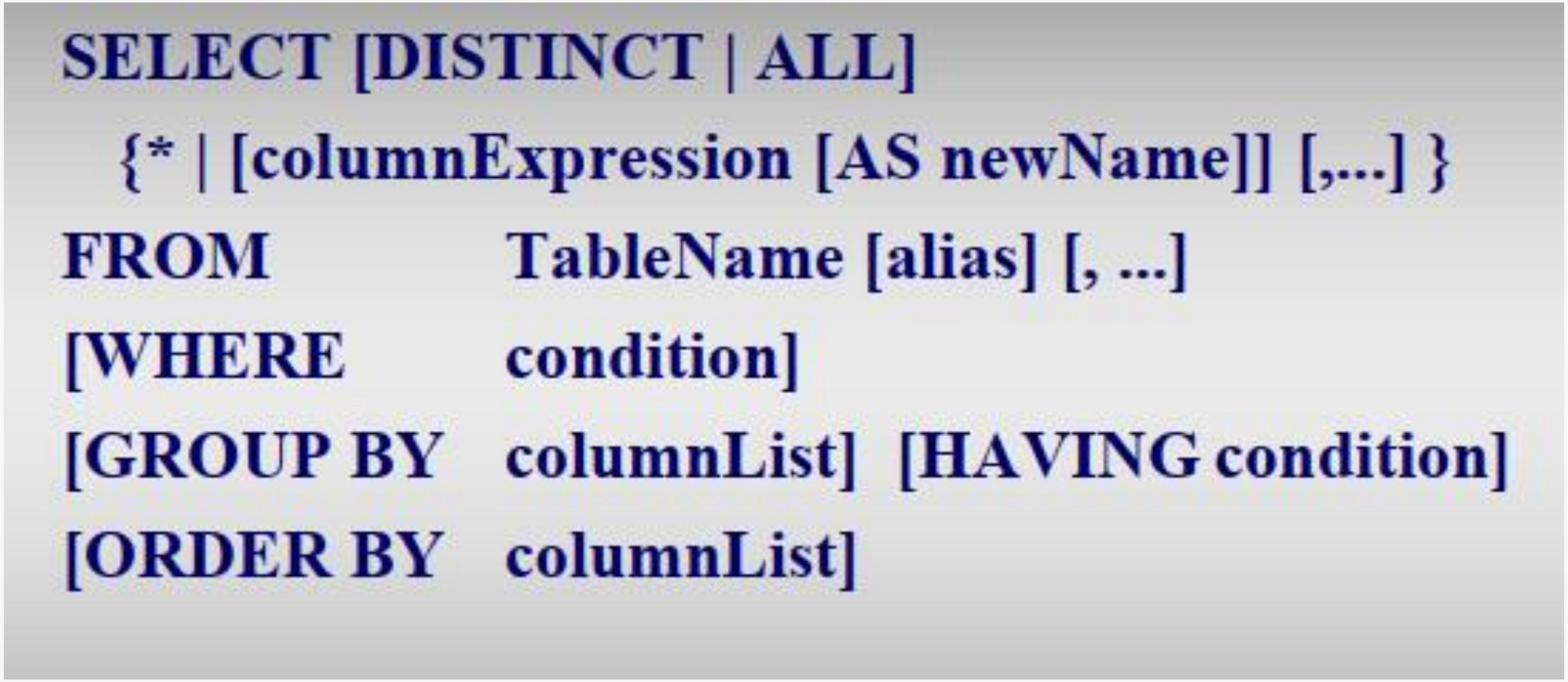
Literals are constants used in SQL statements.

All non-numeric literals must be enclosed in single quotes (e.g. ‘London’).

All numeric literals must not be enclosed in quotes (e.g. 650.00)

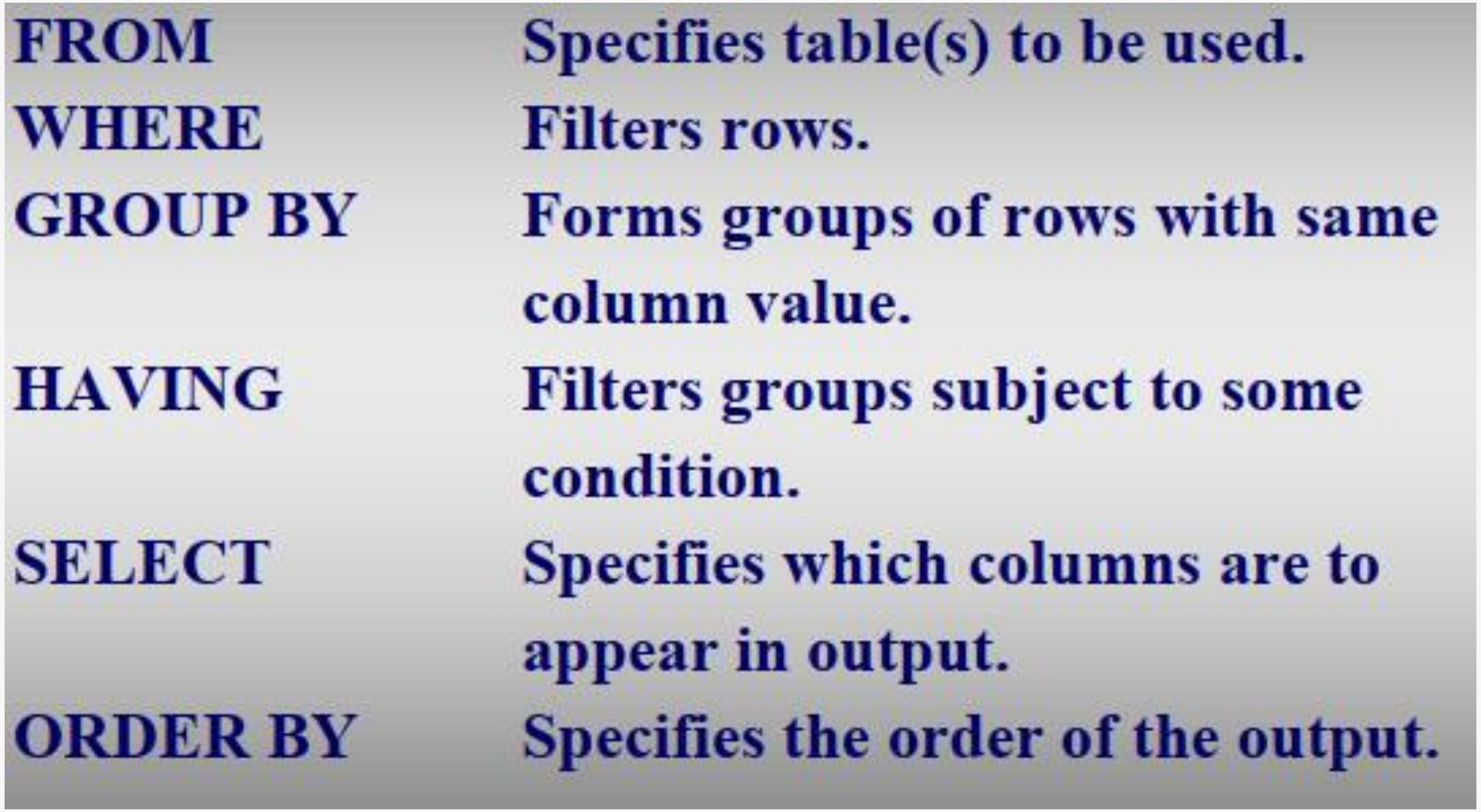
SELECT STATEMENT

SELECT Statement



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

SELECT Statement



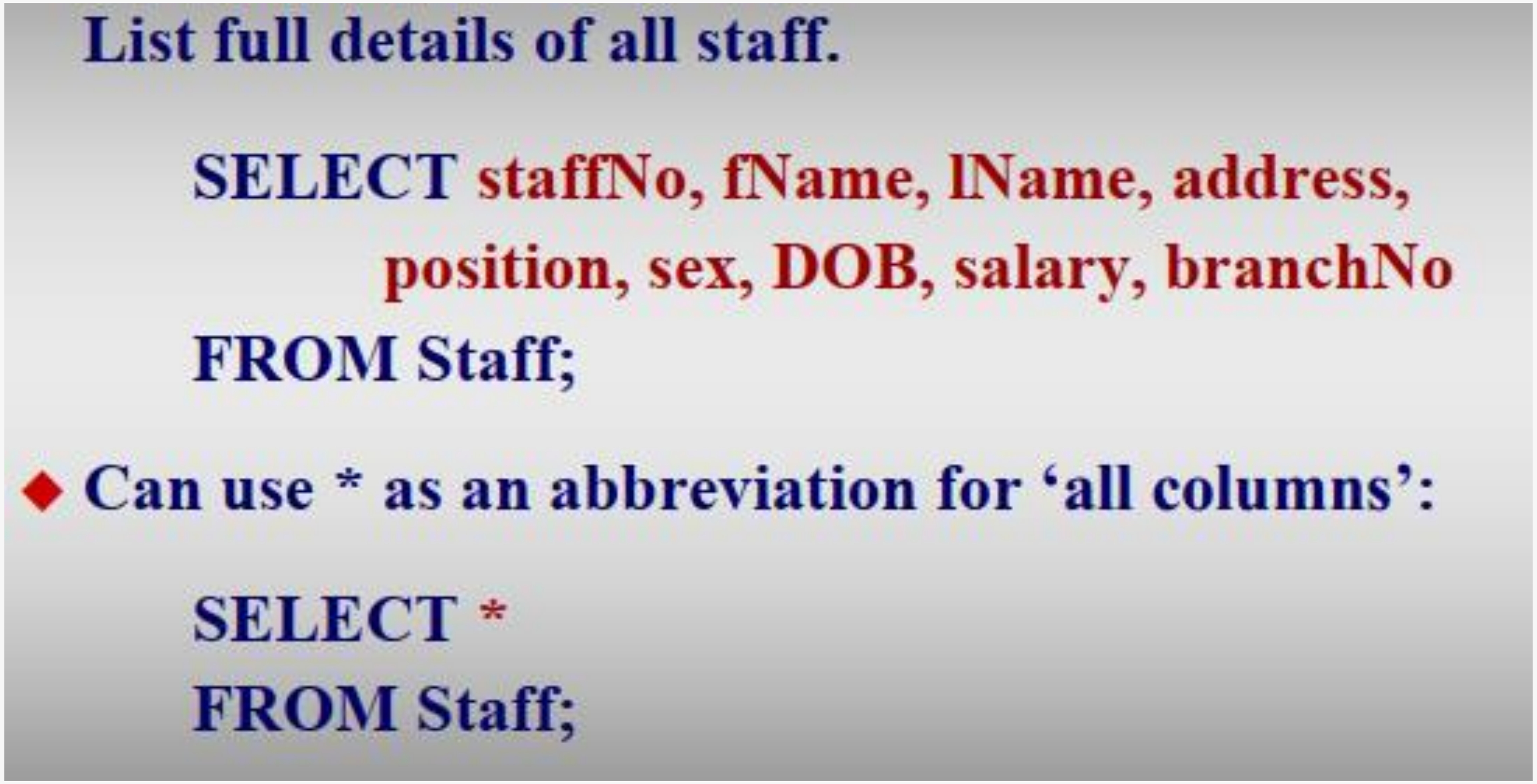
FROM	Specifies table(s) to be used.
WHERE	Filters rows.
GROUP BY	Forms groups of rows with same column value.
HAVING	Filters groups subject to some condition.
SELECT	Specifies which columns are to appear in output.
ORDER BY	Specifies the order of the output.

SELECT Statement

Order of the clauses cannot be changed.

Only SELECT and FROM are mandatory.

Example - All Columns, All Rows



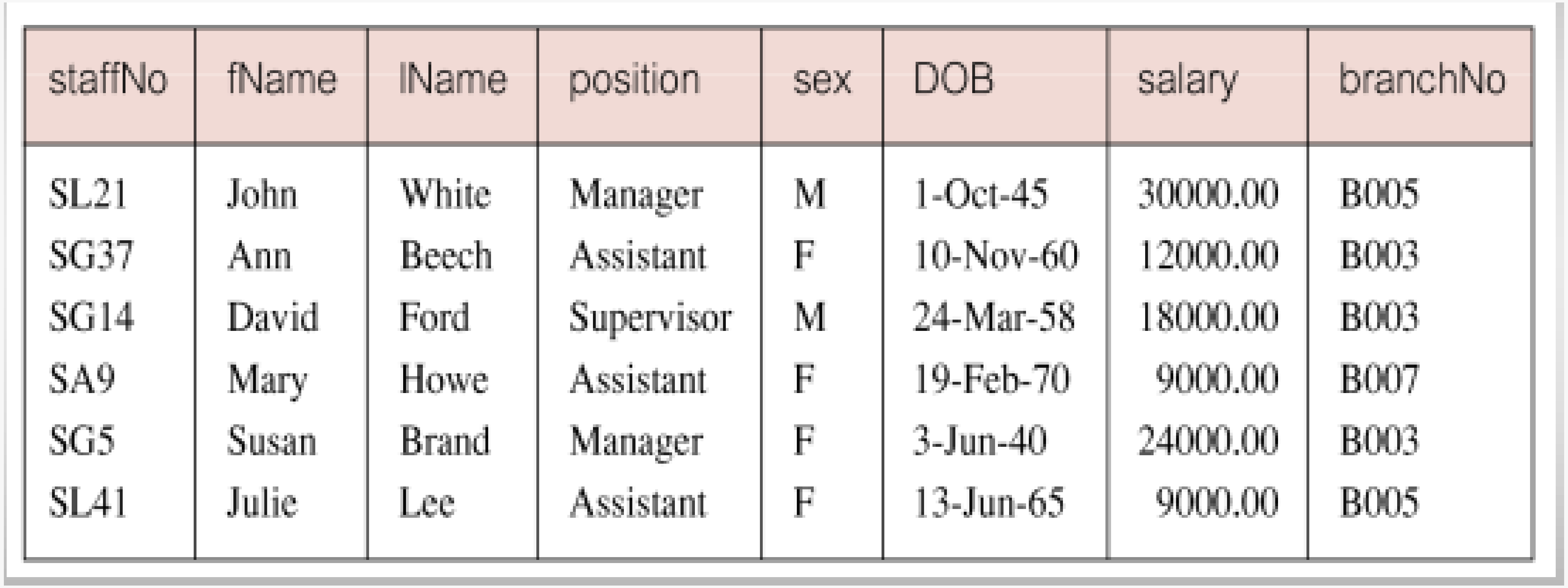
List full details of all staff.

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

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

```
SELECT *
FROM Staff;
```

Example - All Columns, All Rows



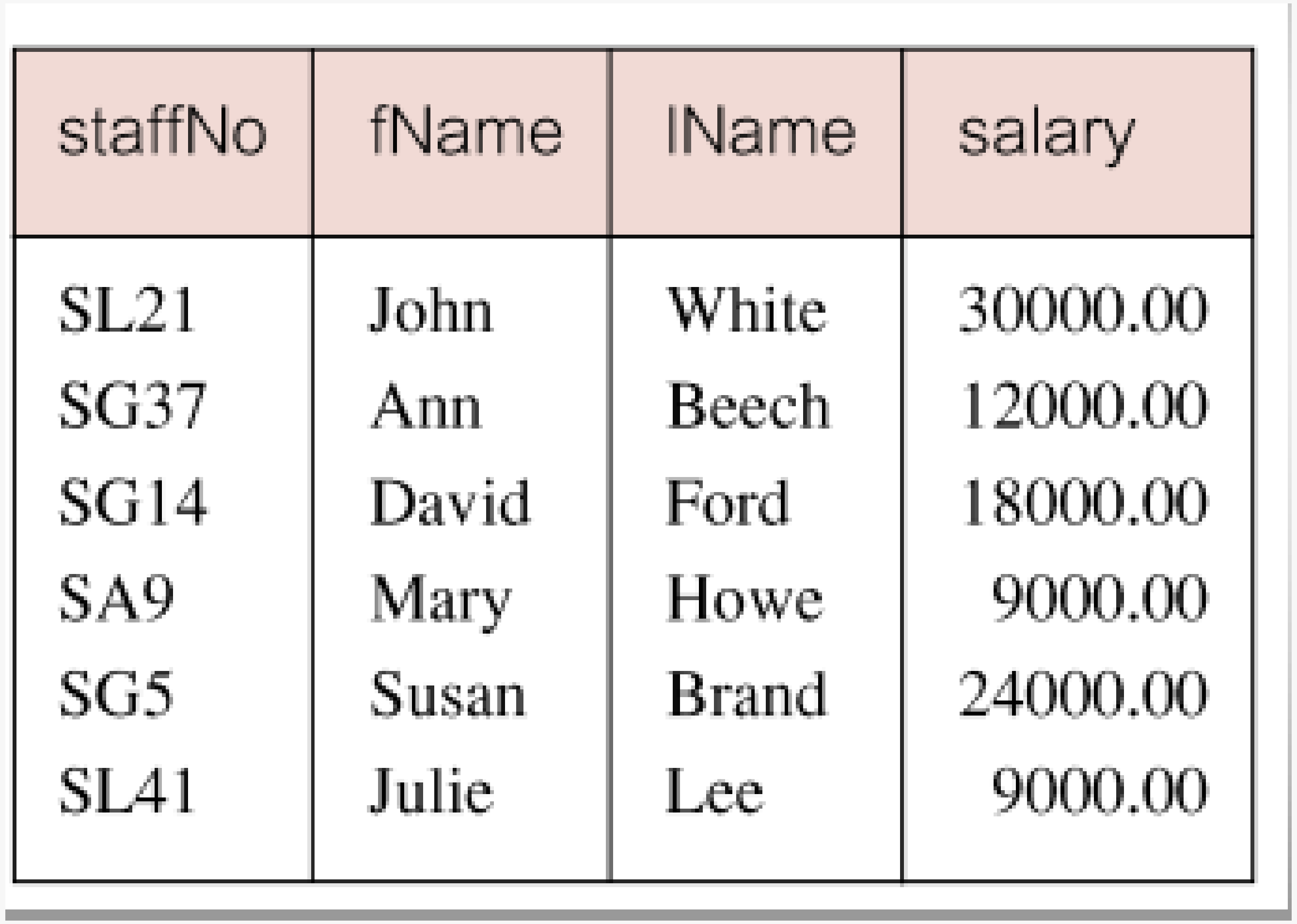
staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003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

Example - All Columns, All Rows

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

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

Example - All Columns, All Rows



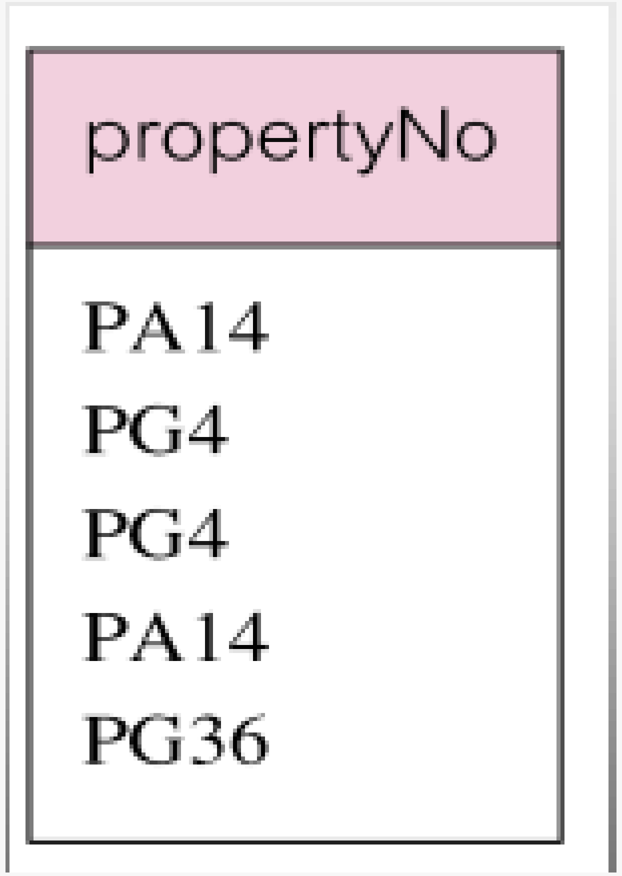
staffNo	fName	lName	salary
SL21	John	White	30000.00
SG37	Ann	Beech	12000.00
SG14	David	Ford	18000.00
SA9	Mary	Howe	9000.00
SG5	Susan	Brand	24000.00
SL41	Julie	Lee	9000.00

Example - Use of DISTINCT

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

- SELECT propertyNo

FROM Viewing;



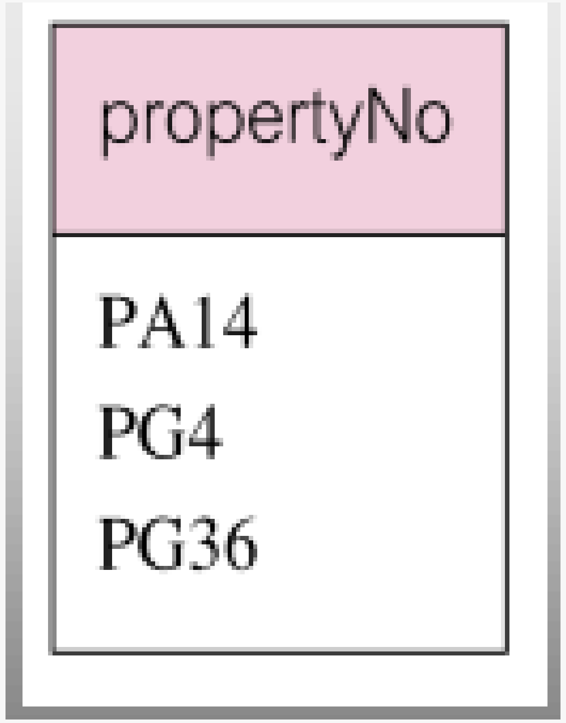
propertyNo
PA14
PG4
PG4
PA14
PG36

Example - Use of DISTINCT

- Use DISTINCT to eliminate duplicates:

- SELECT DISTINCT propertyNo

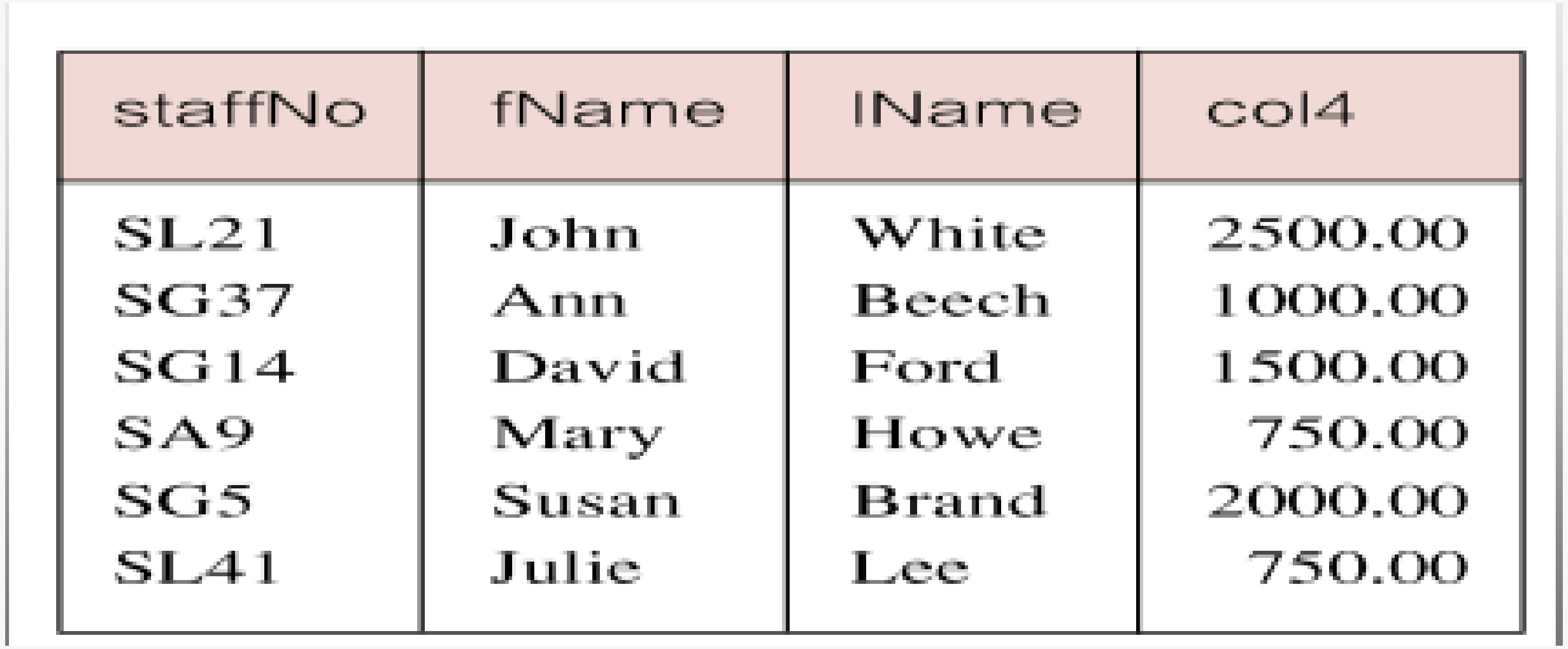
FROM Viewing;



propertyNo
PA14
PG4
PG36

Example Calculated Fields

- Produce list of monthly salaries for all staff, showing staff number, first/last name, and salary.
- SELECT staffNo, fName, lName, salary/12
  FROM Staff



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

Example Calculated Fields

- To name column, use AS clause:
- SELECT staffNo, fName, lName, salary/12

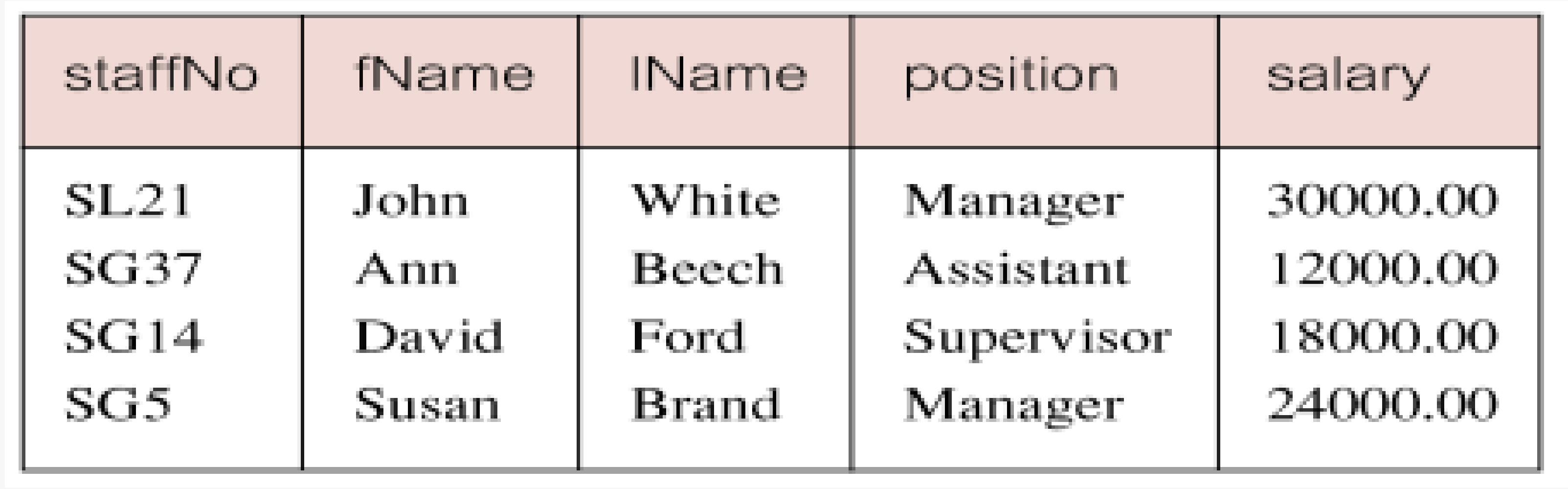
AS monthlySalary

FROM Staff;

Example Comparison Search Condition

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

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



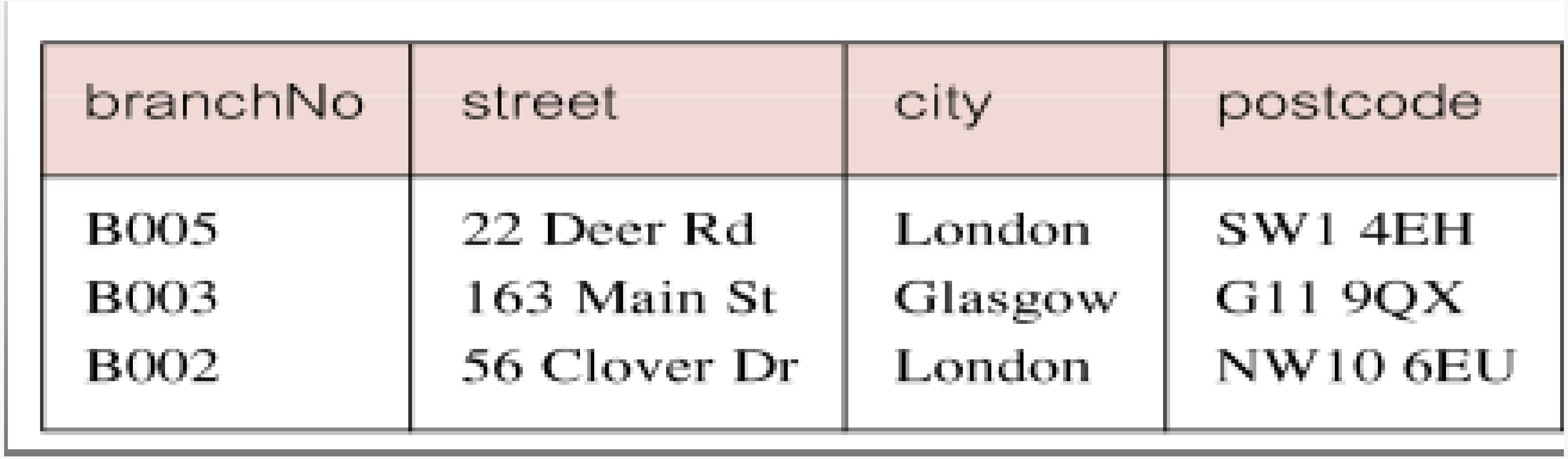
staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG37	Ann	Beech	Assistant	12000.00
SG14	David	Ford	Supervisor	18000.00
SG5	Susan	Brand	Manager	24000.00

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 Range Search Condition

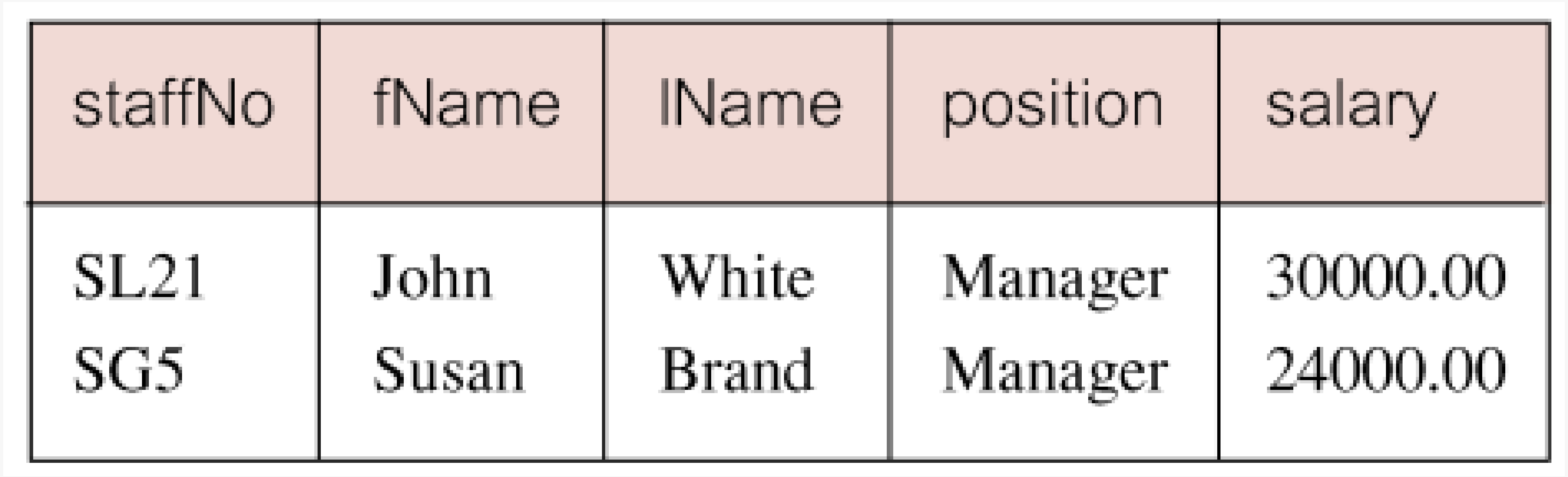
- List all staff with a salary between 20,000 and 30,000.
- SELECT staffNo, fName, lName, position, salary

FROM Staff

WHERE salary BETWEEN 20000 AND 30000;

BETWEEN test includes the endpoints of range

Example Range Search Condition



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

Example Range Search Condition

- Also a negated version NOT BETWEEN.
- 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 Set Membership

List all managers and supervisors.

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

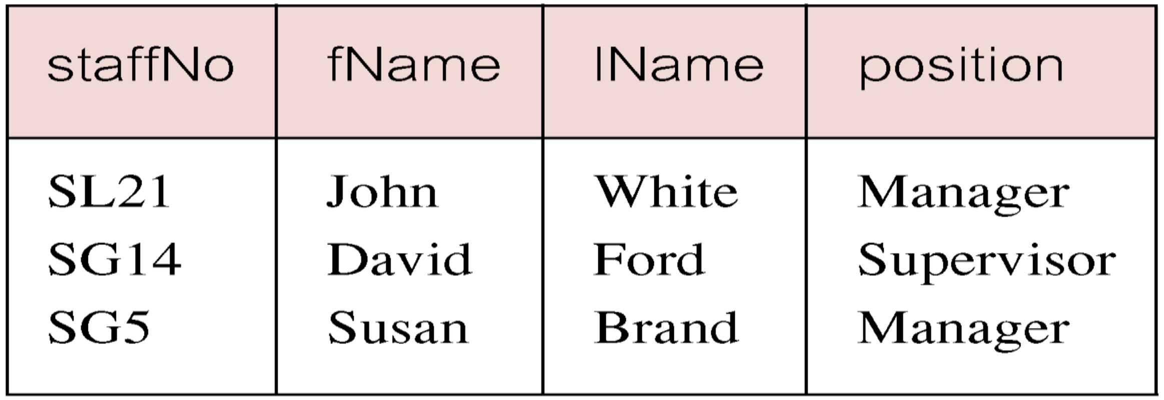


Table 5.8 Result table for Example 5.8.

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

Example Set Membership

- Negated version (NOT IN)
- 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 more efficient when set contains many values

Example Pattern Matching

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

SELECT ownerNo, fName, lName, address, telNo

FROM PrivateOwner

WHERE address LIKE ‘%Glasgow%’;

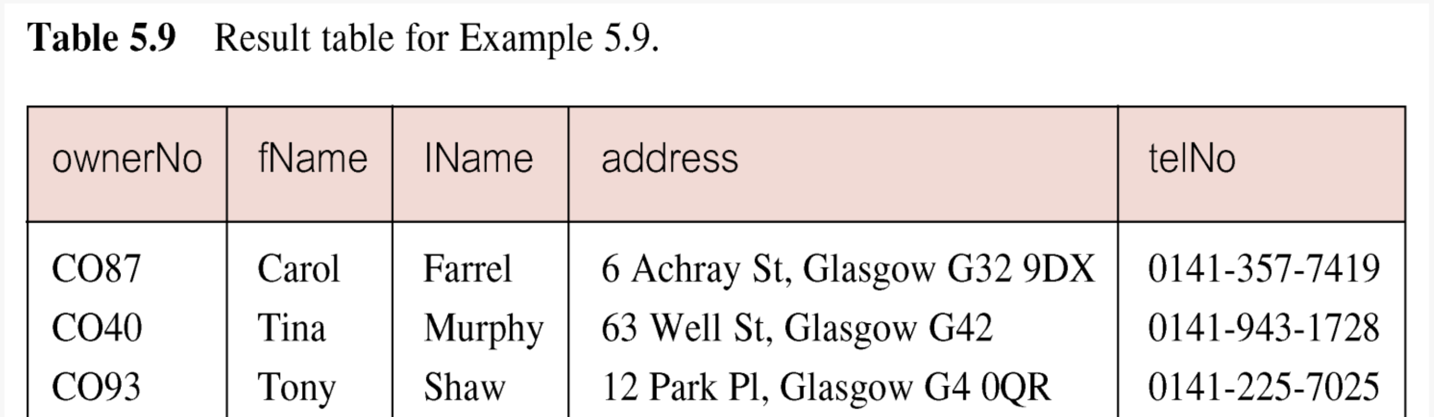


Table 5.9 Result table for Example 5.9.

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 Pattern Matching

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

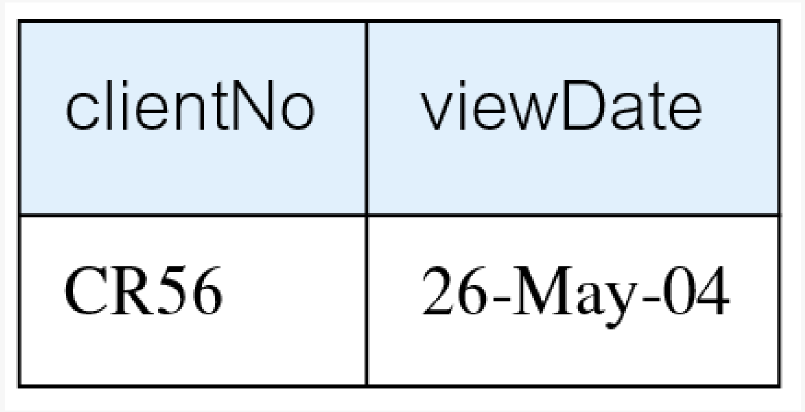
Example NULL Search Condition

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 NULL Search Condition



clientNo	viewDate
CR56	26-May-04

- **Negated version (IS NOT NULL) can test for non-null values**

Thank
You