Chapter 6

SQL: Data Manipulation

Follow Along in The Textbook

- The queries in these slides use the data from Fig 4.3 in the Dream Home case study that you can find on Canvas
- It would help to have this case study available while following the lecture

SQL Query Language Based on Select-Project-Join

- Select: filters rows
- Project: filters columns
- Join: connects information from multiple relations, usually using foreign keys

Key SQL DML Commands

- Select: Retrieves data
- Insert: Adds data
- Update: Modifies data
- Delete: Removes data

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;

Writing SQL Commands

- **Most components of an SQL statement are *case insensitive*, except for literal character data.**
  - **UTK's mysql server is case insensitive for everything but relation names**
    - » **Keywords**
    - » **Field names**
    - » **Literal character data**
  - **Oracle is case-insensitive for keywords, but case sensitive for field names and literal character data**

Writing SQL Commands (Cont)

- **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, each should appear on a separate line and be indented under start of clause.**

Writing SQL Commands

- **These slides use an 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’).
  - In SQL standard, single quotes enclose strings
  - Double quotes enclose things in database, such as column names (e.g., “street address”)
  - Mysql allows double quotes around strings but don’t get into the habit of using them
- All numeric literals must not be enclosed in quotes (e.g. 650.00).

SELECT Statement

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

SELECT Statement

SELECT	**Specifies which columns are to appear in output.**
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.**
ORDER BY	**Specifies the order of the output.**

SELECT Statement

- **Order of the clauses cannot be changed.**
- **Only SELECT and FROM are mandatory.**

Example SQL Queries

- All queries are taken from the books DreamHome case study

Example 6.1 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 6.1 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 6.2 Specific 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 6.2 Specific Columns, All Rows

Table 5.2 Result table for Example 5.2.

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 6.3 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 6.3 Use of DISTINCT

- **Use DISTINCT to eliminate duplicates:**

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

propertyNo
PA14
PG4
PG36

Example 6.4 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;
```

Table 5.4 Result table for Example 5.4.

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 6.4 Calculated Fields

- To name column, use AS clause:

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

Example 6.5 Comparison Search Condition

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

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

Table 5.5 Result table for Example 5.5.

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 6.6 Compound Comparison Search Condition

List addresses of all branch offices in London or Glasgow.

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

Table 5.6 Result table for Example 5.6.

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 6.7 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.**
- **UTK's mysql BETWEEN test is inclusive of the endpoints of the range**

Example 6.7 Range Search Condition

Table 5.7 Result table for Example 5.7.

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

Example 6.7 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 6.8 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 6.8 Set Membership

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

Example 6.9 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 6.9 Pattern Matching

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

Example 6.10 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 6.10 NULL Search Condition

clientNo	viewDate
CR56	26-May-04

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

Example 6.11 Single Column Ordering

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

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

Example 6.11 Single Column Ordering

Table 5.11 Result table for Example 5.11.

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 6.12 Multiple Column Ordering

Produce abbreviated list of properties in order of property type.

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

Example 6.12 Multiple Column Ordering

Table 5.12(a) Result table for Example 5.12 with one sort key.

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

Example 6.12 Multiple Column Ordering

- Four flats in this list - as no minor sort key specified, system arranges these rows in any order it chooses.
- To arrange in order of rent, specify minor order:

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

Example 6.12 Multiple Column Ordering

Table 5.12(b) Result table for Example 5.12 with two sort keys.

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 number of values in specified column.

SUM returns sum of values in specified column.

AVG returns average of values in specified column.

MIN returns smallest value in specified column.

MAX returns largest value in 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.
- Apart from COUNT(*), each function eliminates nulls first and operates only on remaining non-null values.

SELECT Statement - Aggregates

- COUNT(*) counts all rows of a table, regardless of whether nulls or duplicate values occur.
- 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

- Aggregate functions can be used only in SELECT list and in HAVING clause.

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

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

Example 6.13 Use of COUNT(*)

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

```
SELECT COUNT(*) AS myCount
FROM PropertyForRent
WHERE rent > 350;
```

myCount
5

Example 6.14 Use of COUNT(DISTINCT)

How many different properties viewed in May '04?

```
SELECT COUNT(DISTINCT propertyNo) AS myCount
FROM Viewing
WHERE viewDate BETWEEN '1-May-04'
      AND '31-May-04';
```

myCount
2

Example 6.15 Use of COUNT and SUM

Find number of Managers and sum of their salaries.

```
SELECT COUNT(staffNo) AS myCount,
       SUM(salary) AS mySum
FROM Staff
WHERE position = 'Manager';
```

myCount	mySum
2	54000.00

Example 6.16 Use of MIN, MAX, AVG

Find minimum, maximum, and 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 for all rows in a "set" (or equivalently, a group)
  - Example: Use GROUP BY to get the total staff salary for each branch
- SELECT and GROUP BY closely integrated: each item in SELECT list must be:
  - an aggregate function that produces a sub-total
  - a column name used in the GROUP BY clause to group rows to be aggregated
  - constants
  - expression involving combinations of the above.

SELECT Statement - Grouping

- 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 filters groups rather than individual rows
  - HAVING should filter based on aggregate functions that compute a subtotal for each group, such as COUNT(staffNo) > 1
  - HAVING eliminates groups after the aggregate function is computed
- 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

End of Aggregate Functions

Joins

Multi-Table Queries

- If result columns come from more than one table must use a join.

Branch(branchNo, street, city, postcode)

Staff(staffNo, fName, lName, position, sex, birthdate, salary, branchNo)

What address does Brad Vander Zanden work at?

```
SELECT b.street, b.city, b.postcode FROM Branch b, Staff s
        WHERE s.lname = 'Vander Zanden' AND s.fname = 'Brad'
                AND s.branchNo = b.branchNo;
```

- 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 (also called Inner 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.

Table 5.24 Result table for Example 5.24.

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

Some Syntactic Sugar

- If two relations share a common join column, you can write:

```
SELECT c.clientNo, fName, lName,  propertyNo, comment
    FROM Client c INNER JOIN Viewing
    USING (clientNo);
```

- If you want to do an equi-join on all common columns, use NATURAL JOIN and you can omit the join column names

```
SELECT c.clientNo, fName, lName, propertyNo, comment
    FROM Client NATURAL JOIN Viewing;
```

Example 6.26 Three Table Join

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

```
SELECT b.branchNo, b.city, s.staffNo, fName, lName,
       propertyNo
FROM Branch b NATURAL JOIN Staff s
NATURAL JOIN PropertyForRent p
ORDER BY b.branchNo, s.staffNo, propertyNo;
```

Example 6.26 Three Table Join

Table 5.26 Result table for Example 5.26.

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

Example 6.27 Multiple Grouping Columns

Find number of properties handled by each staff member.

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

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. **Eliminate columns that are not in the select list (a projection operation)**

Computing a Join

4. If DISTINCT has been specified, eliminate any duplicate rows from the result table.
5. If there is a GROUP BY clause, perform the aggregate function(s) on those groups
6. If there is an ORDER BY clause, sort result table as required.

Outer Joins

- **Inner join: If a row in one relation of the join is not matched in the other relation of the join, the row is omitted from the result table.**
- **Outer join operations retain rows that do not satisfy the join condition.**

Outer Joins

- Left Join: Includes all rows from the left relation in the result
  - Unmatched rows have NULL values for the columns drawn from the right relation
- Right Join: Includes all rows from the right relation in the result
  - Unmatched rows have NULL values for the columns drawn from the left relation
- Full Join: Includes all unmatched rows in the result relation, both from the left and right relations

Example 6.28 Left Outer Join

List branches and properties that are in same city along with any unmatched branches.

```
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

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.**

Table 5.28 Result table for Example 5.28.

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

Example 6.29 Right Outer Join

List branches and properties in same city and any unmatched properties.

```
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.**

Table 5.29 Result table for Example 5.29.

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

Example 6.30 Full Outer Join

List branches and properties in same city and 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.**

Table 5.30 Result table for Example 5.30.

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

End of Joins

Combining Result Tables

- Set Union(A, B): returns all rows that are in either A or B
- Set Intersection(A, B): returns all rows that are in both A and B
- Set Difference(A, B): returns all rows in A that are not in B

Union-compatibility

- 2 relations are union compatible if they have
  - The same number of columns, and
  - each pair of columns is drawn from the same domain
- The set operations require that their parameter relations be union compatible

Sample Scenario

- University with 4 categories of members
  - Staff
  - Professors
  - Students
  - Administrators

Union

- Find all employees
  - (SELECT name FROM Staff)
    UNION
    (SELECT name FROM Professors)
    UNION
    (SELECT name FROM Administrators)

Intersection

- Find all professors who are also administrators

  (SELECT name FROM Professors)

  INTERSECT

  (SELECT name FROM Administrators)

Intersect in MySQL

- MySQL does not support Intersect keyword. Use a join to implement intersection. For example, R(a,b) ∩ S(a,b):
  a. If Nulls are not an issue: Use Join with distinct, which gets rid of duplicates

     Select Distinct R.a, R.b from R NATURAL JOIN S

  b. Use the <=> operator to pick up Nulls: <=> does equality testing that includes Nulls

     Select Distinct R.a, R.b from R, S

     where R.a <=> S.a and R.b <=> S.b;

Set Difference

- Find all administrators who are not professors
  - (SELECT name FROM Administrators)
    EXCEPT
    (SELECT name FROM Professors)

Set Difference in MySQL

- MySQL does not support EXCEPT. To get P(id) – Q(id):
  a. Select * from P
     where P.id not in (Select id from Q);
  b. Select * from P left join Q
     on P.id = Q.id where Q.id is NULL

P.id	Q.id	P Left Join Q =	P.id	Q.id
3	3		3	3
7	6		7	Null
8	4		8	Null

Modification Statements

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 Staff table supplying data for all mandatory columns.

```
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');
```

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 6.38/39 UPDATE All Rows

Give all staff a 3% pay increase.

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

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 6.41/42 DELETE Specific Rows

Delete all viewings that relate to property PG4.

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

Delete all records from the Viewing table.

```
DELETE FROM Viewing;
```