Database – SQL

A Level Computer Science: Exchanging Data

Learning Aims

- Be able to use SQL to retrieve data from multiple tables of a relational database
- Be able to interpret and modify SQL
- Be able to use SQL to define a database table
- Be able to use SQL to update, insert and delete data from multiple tables of a relational database

CRUD

All relational databases must have certain basic functionality to be useful. This is often summarised by the acronym CRUD. This stands for:

- Create
- Read
- Update
- Delete.

Each of these functions can be actioned by an equivalent SQL statement:

- INSERT/CREATE
- SELECT
- UPDATE
- DELETE.

SQL

SQL, or Structured Query Language (pronounced either as S-Q-L or Sequel) is a declarative language used for querying and updating tables in a relational database.

It can also be used to create tables.

We will look at SQL statements used in querying a database.

The tables shown will be used to demonstrate some SQL statements.

The tables are part of a database used by a retailer to store details of CDs in a database that will allow information about the CDs to be extracted. The four entities CD, CDSong, Song and Artist are connected by the following relationships:

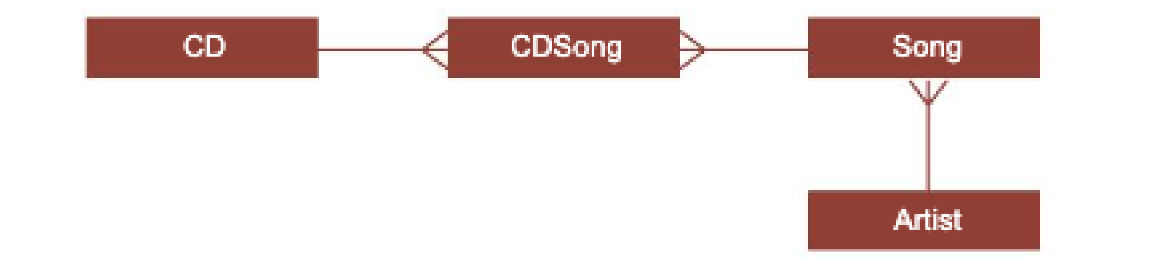
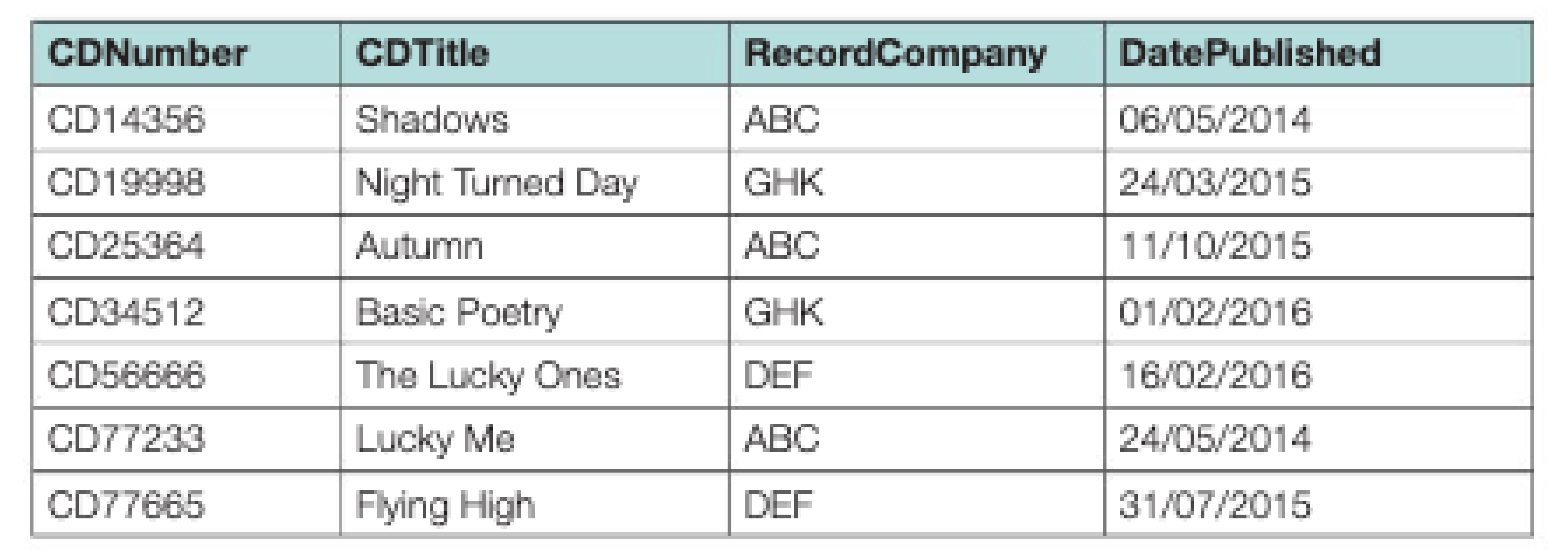


Figure 18.1

CD Table

CD table



CDNumber	CDTitle	RecordCompany	DatePublished
CD14356	Shadows	ABC	06/05/2014
CD19998	Night Turned Day	GHK	24/03/2015
CD25364	Autumn	ABC	11/10/2015
CD34512	Basic Poetry	GHK	01/02/2016
CD56666	The Lucky Ones	DEF	16/02/2016
CD77233	Lucky Me	ABC	24/05/2014
CD77665	Flying High	DEF	31/07/2015

SELECT .. FROM .. WHERE

The SELECT statement is used to extract a collection of fields from a given table. The basic syntax of this statement is

SELECT list the fields to be displayed

FROM list the table or tables the data will come from

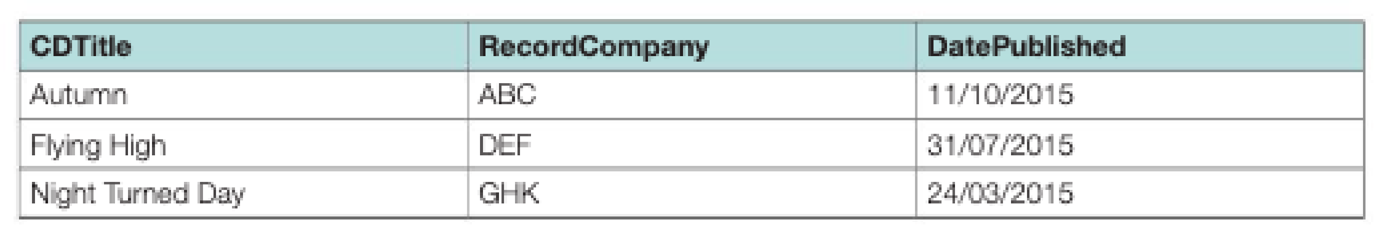
WHERE list the search criteria

ORDER BY list the fields that the results are to be sorted on (default is Ascending order)

Example 1

```
SELECT      CDTitle, RecordCompany, DatePublished
FROM        CD
WHERE       DatePublished BETWEEN #01/01/2015# AND #31/12/2015#
ORDER BY    CDTitle
```

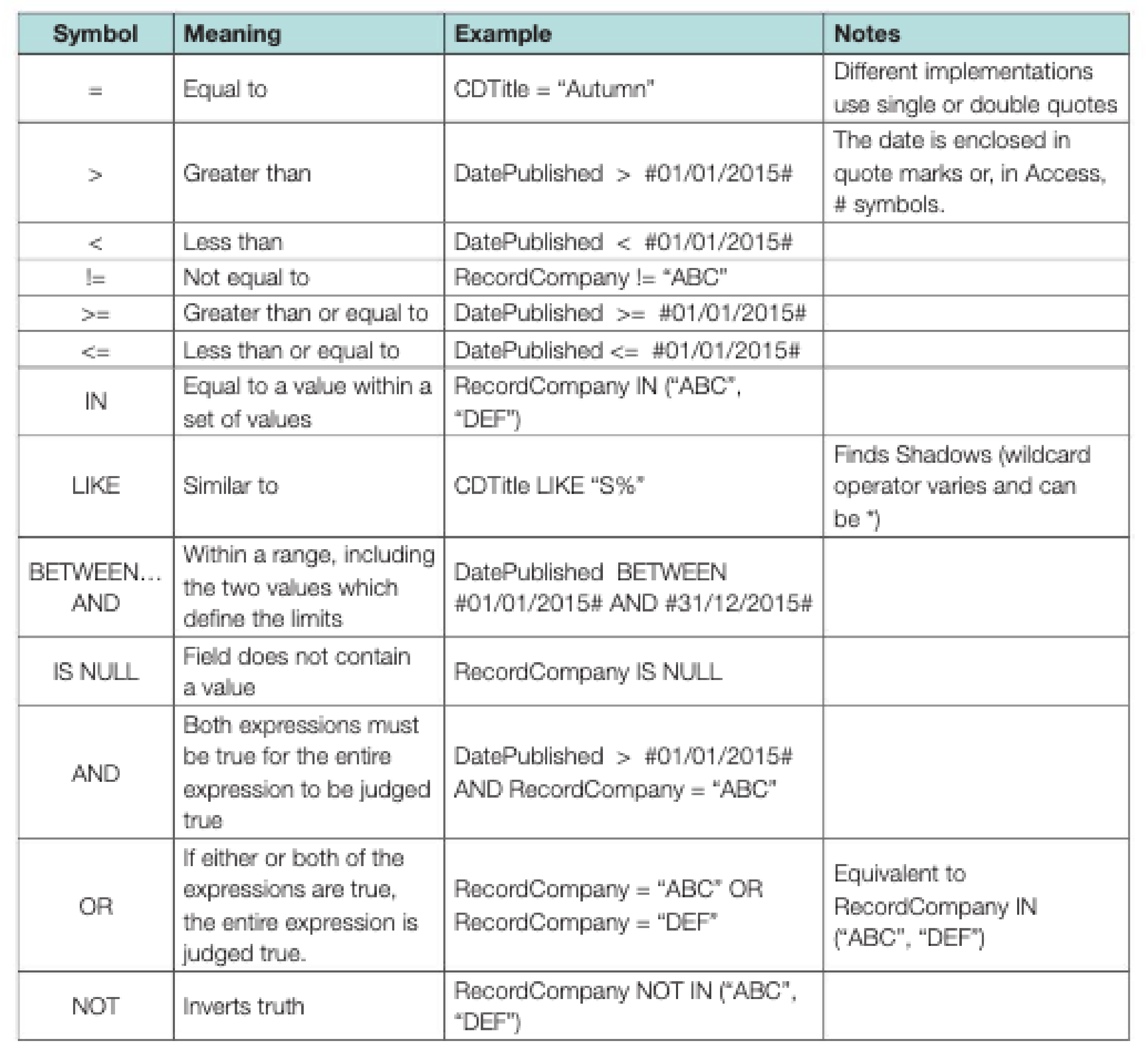
This will return the following records:



CDTitle	RecordCompany	DatePublished
Autumn	ABC	11/10/2015
Flying High	DEF	31/07/2015
Night Turned Day	GHK	24/03/2015

Conditions

Conditions in SQL are constructed from the following operators:



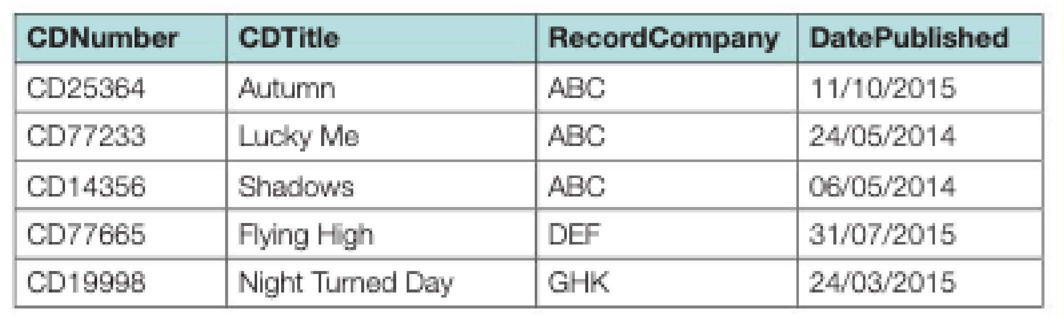
Symbol	Meaning	Example	Notes
=	Equal to	CDTitle = "Autumn"	Different implementations use single or double quotes
>	Greater than	DatePublished > #01/01/2015#	The date is enclosed in quote marks or, in Access, # symbols.
<	Less than	DatePublished < #01/01/2015#	
!=	Not equal to	RecordCompany != "ABC"	
>=	Greater than or equal to	DatePublished >= #01/01/2015#	
<=	Less than or equal to	DatePublished <= #01/01/2015#	
IN	Equal to a value within a set of values	RecordCompany IN ("ABC", "DEF")	
LIKE	Similar to	CDTitle LIKE "S%"	Finds Shadows (wildcard operator varies and can be *)
BETWEEN… AND	Within a range, including the two values which define the limits	DatePublished BETWEEN #01/01/2015# AND #31/12/2015#	
IS NULL	Field does not contain a value	RecordCompany IS NULL	
AND	Both expressions must be true for the entire expression to be judged true	DatePublished > #01/01/2015# AND RecordCompany = "ABC"	
OR	If either or both of the expressions are true, the entire expression is judged true.	RecordCompany = "ABC" OR RecordCompany = "DEF"	Equivalent to RecordCompany IN ("ABC", "DEF")
NOT	Inverts truth	RecordCompany NOT IN ("ABC", "DEF")	

Specifying a sort order

ORDER BY gives you control over the order in which records appear in the Answer table. If for example you want the records to be displayed in ascending order of RecordCompany and within that, descending order of DatePublished, you would write, for example:

```
SELECT      *
FROM        CD
WHERE       DatePublished < #31/12/2015#
ORDER BY    RecordCompany, DatePublished Desc
```

This would produce the following results:

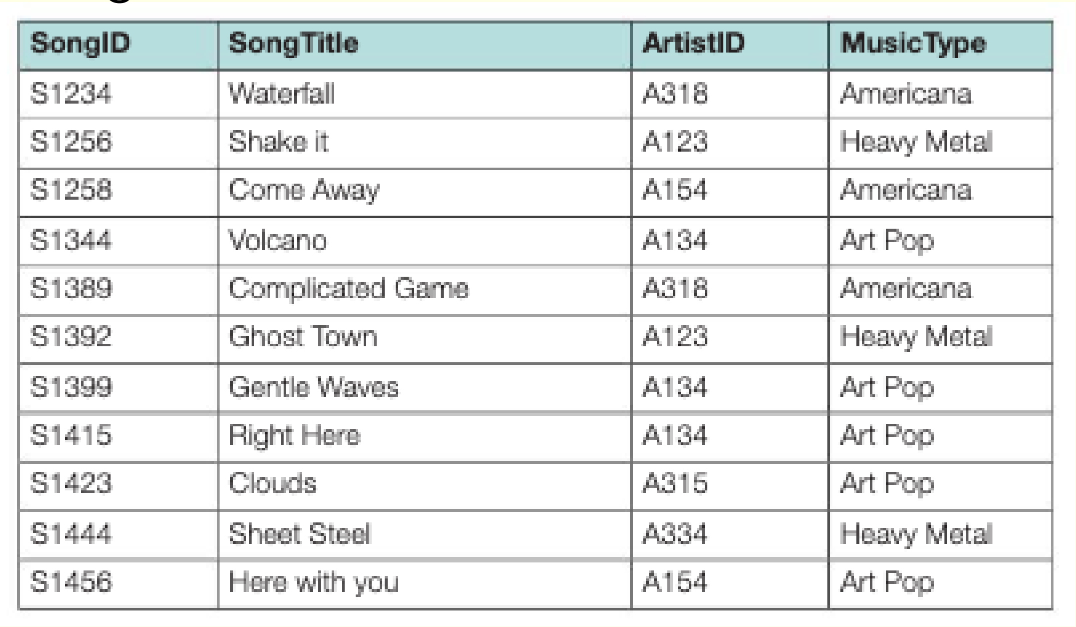


CDNumber	CDTitle	RecordCompany	DatePublished
CD25364	Autumn	ABC	11/10/2015
CD77233	Lucky Me	ABC	24/05/2014
CD14356	Shadows	ABC	06/05/2014
CD77665	Flying High	DEF	31/07/2015
CD19998	Night Turned Day	GHK	24/03/2015

Extracting data from several tables

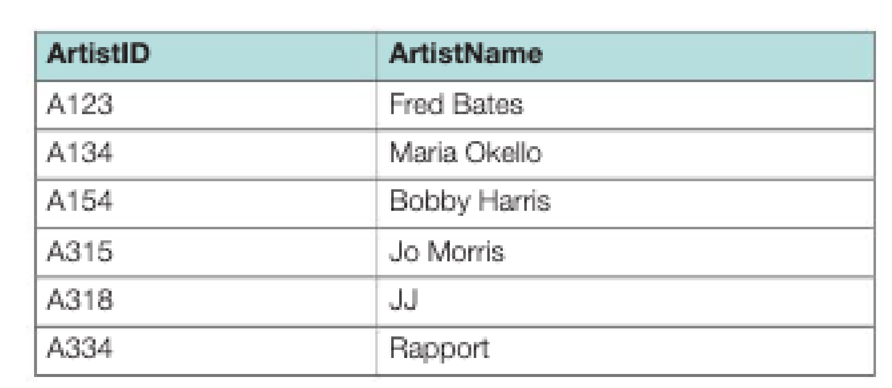
So far we have only taken data from one table. The **Song** and **Artist** tables have the following contents:

Song table



SongID	SongTitle	ArtistID	MusicType
S1234	Waterfall	A318	Americana
S1256	Shake it	A123	Heavy Metal
S1258	Come Away	A154	Americana
S1344	Volcano	A134	Art Pop
S1389	Complicated Game	A318	Americana
S1392	Ghost Town	A123	Heavy Metal
S1399	Gentle Waves	A134	Art Pop
S1415	Right Here	A134	Art Pop
S1423	Clouds	A315	Art Pop
S1444	Sheet Steel	A334	Heavy Metal
S1456	Here with you	A154	Art Pop

Artist table



ArtistID	ArtistName
A123	Fred Bates
A134	Maria Okello
A154	Bobby Harris
A315	Jo Morris
A318	JJ
A334	Rapport

Extracting data from several tables

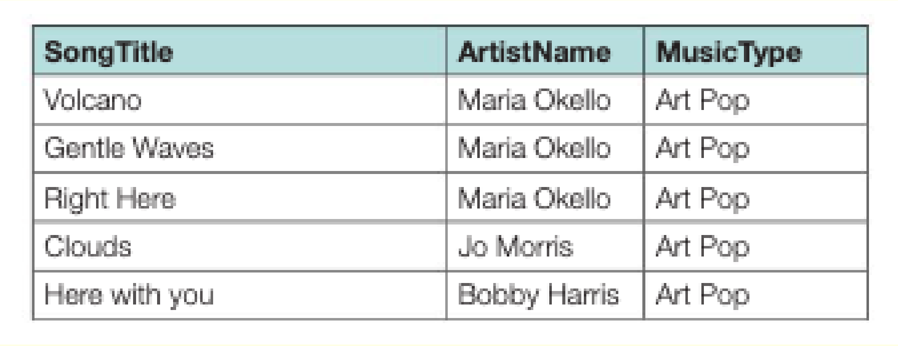
Using SQL you can combine data from two or more tables, by specifying which table the data is held in.

For example, suppose you wanted SongTitle, ArtistName and MusicType for all Art Pop music.

When more than one table is involved, SQL uses the syntax tablename.fieldname. (The table name is optional unless the field name appears in more than one table.)

```
SELECT          Song.SongTitle, Artist.ArtistName, Song.MusicType
FROM            Song, Artist
WHERE           (Song.ArtistID = Artist.ArtistID) AND (Song.MusicType = "Art Pop")
```

The condition Song.ArtistID = Artist.ArtistID provides the link between the Song and Artist tables so that the artist's name corresponding to the ArtistID in the Song table can be found in the Artist table. This will produce the following results:



SongTitle	ArtistName	MusicType
Volcano	Maria Okello	Art Pop
Gentle Waves	Maria Okello	Art Pop
Right Here	Maria Okello	Art Pop
Clouds	Jo Morris	Art Pop
Here with you	Bobby Harris	Art Pop

SQL JOIN

JOIN provides an alternative method of combining rows from two or more tables, based on a common field between them.

The query above could be written as follows:

```
SELECT      Song.SongTitle, Artist.ArtistName, Song.MusicType
FROM        Song
JOIN        Artist
ON          Song.ArtistID = Artist.ArtistID
WHERE       Song.MusicType = "Art Pop"
```

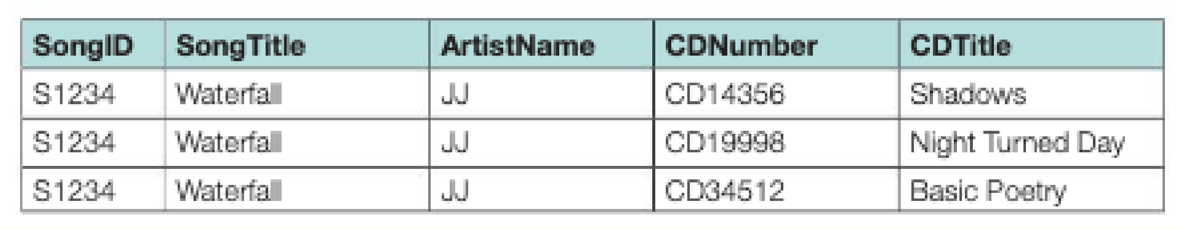
SQL JOIN

The fourth table in the database is the table CDSong which links the songs to one or more of the CDs.

We can make a search to find the CDNumbers and titles all the CDs containing the song Waterfall, sung by JJ.

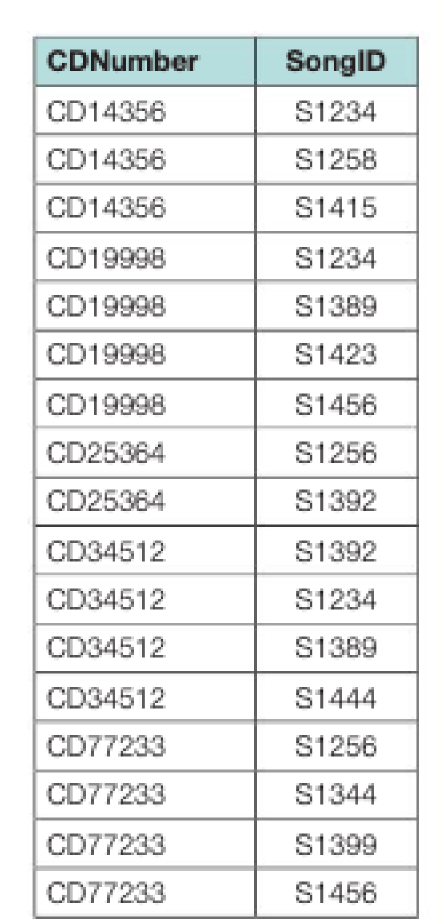
```
SELECT     Song.SongID, Song.SongTitle, Artist.ArtistName, CDSong.CDNumber, CD.CDTitle
FROM       Song, Artist, CDSong, CD
WHERE      CDSong.CDNumber = CD.CDNumber
           AND CDSong.SongID = Song.SongID
           AND Artist.ArtistID = Song.ArtistID
           AND Song.SongTitle = "Waterfall"
```

This will produce the following results:



SongID	SongTitle	ArtistName	CDNumber	CDTitle
S1234	Waterfall	JJ	CD14356	Shadows
S1234	Waterfall	JJ	CD19998	Night Turned Day
S1234	Waterfall	JJ	CD34512	Basic Poetry

CDSong table



CDNumber	SongID
CD14356	S1234
CD14356	S1258
CD14356	S1415
CD19998	S1234
CD19998	S1389
CD19998	S1423
CD19998	S1456
CD25364	S1256
CD25364	S1392
CD34512	S1392
CD34512	S1234
CD34512	S1389
CD34512	S1444
CD77233	S1256
CD77233	S1344
CD77233	S1399
CD77233	S1456

Note

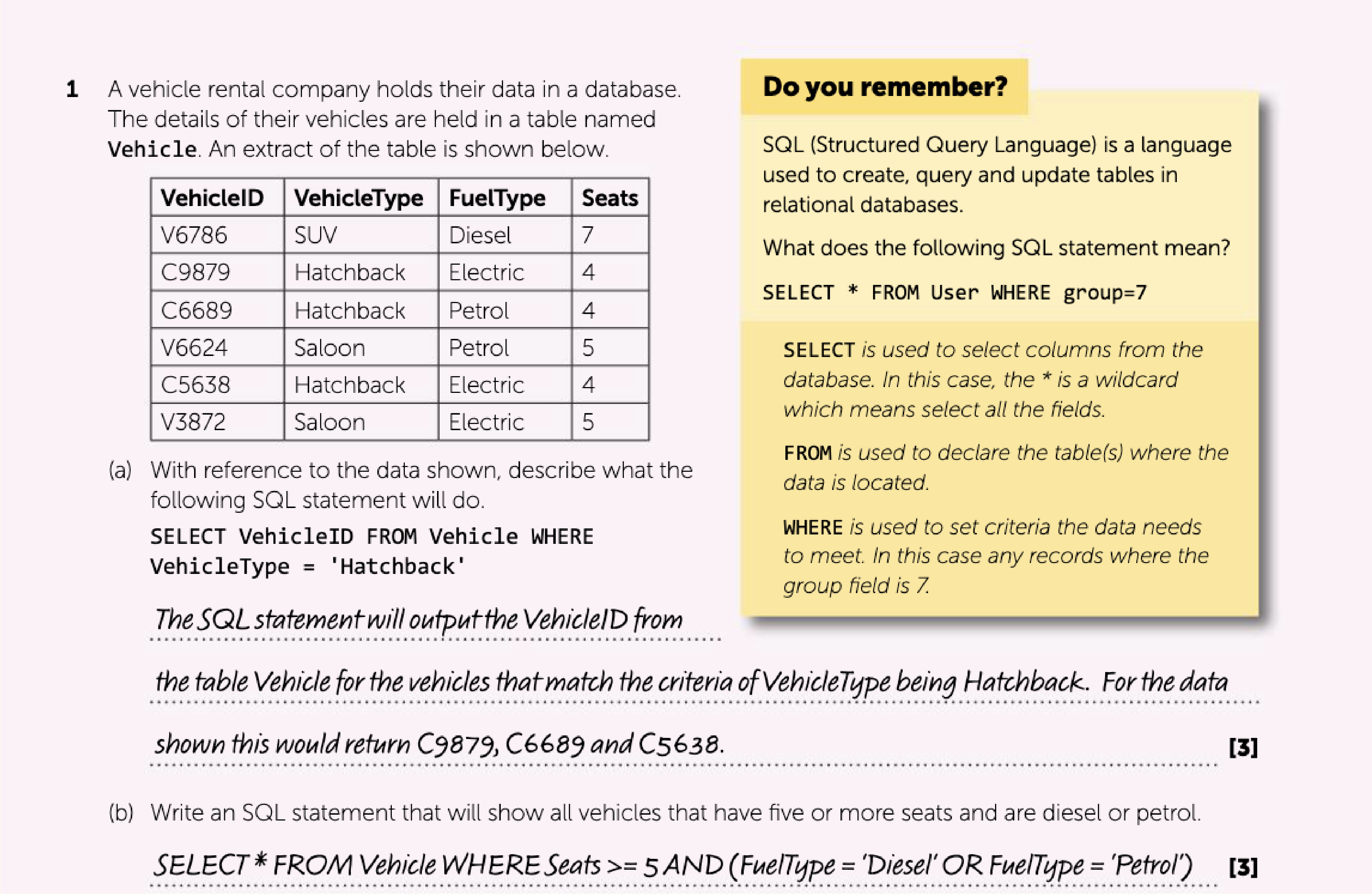
Note that in the SELECT statement, it does not matter whether you specify Song.SongID or CDSong.SongID since they are connected.

The same is true of CDSong.CDNumber and CD.CDNumber.

The Boolean conditions

CDSong.SongID = Song.SongID and Artist.ArtistID = Song.ArtistID are required to specify the relationships between the data tables.

Model Answer



1 A vehicle rental company holds their data in a database. The details of their vehicles are held in a table named `Vehicle`. An extract of the table is shown below.

VehicleID	VehicleType	FuelType	Seats
V6786	SUV	Diesel	7
C9879	Hatchback	Electric	4
C6689	Hatchback	Petrol	4
V6624	Saloon	Petrol	5
C5638	Hatchback	Electric	4
V3872	Saloon	Electric	5

(a) With reference to the data shown, describe what the following SQL statement will do.

```
SELECT VehicleID FROM Vehicle WHERE
VehicleType = 'Hatchback'
```

The SQL statement will output the VehicleID from the table Vehicle for the vehicles that match the criteria of VehicleType being Hatchback. For the data shown this would return C9879, C6689 and C5638. **[3]**

(b) Write an SQL statement that will show all vehicles that have five or more seats and are diesel or petrol.

*SELECT * FROM Vehicle WHERE Seats >= 5 AND (FuelType = 'Diesel' OR FuelType = 'Petrol')* **[3]**

Do you remember?

SQL (Structured Query Language) is a language used to create, query and update tables in relational databases.

What does the following SQL statement mean?

```
SELECT * FROM User WHERE group=7
```

`SELECT` *is used to select columns from the database. In this case, the * is a wildcard which means select all the fields.*

`FROM` *is used to declare the table(s) where the data is located.*

`WHERE` *is used to set criteria the data needs to meet. In this case any records where the group field is 7.*

Defining a database table

The following example shows how to create a new database table.

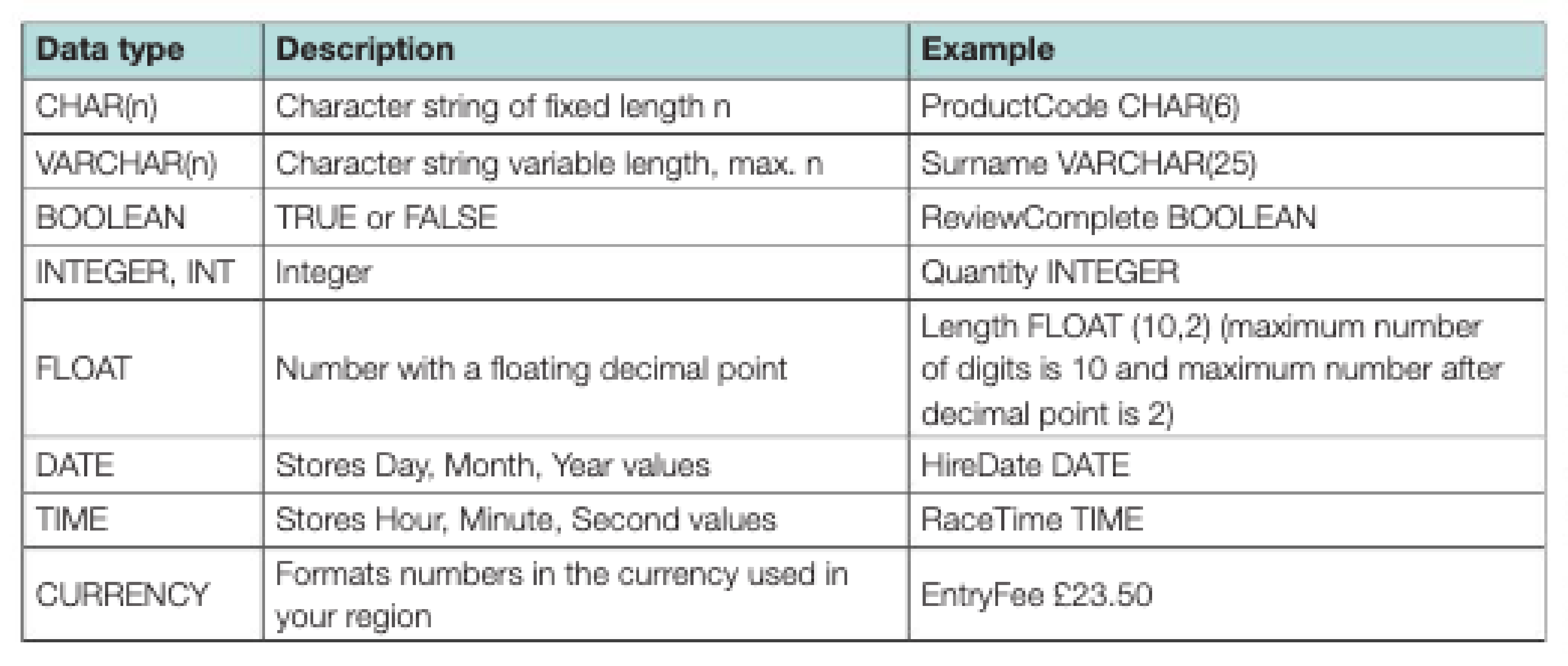
Use SQL to create a table named Employee, which has four columns:

EmpID (a compulsory int field which is the primary key), EmpName (a compulsory character field of length 10), HireDate (an optional date field) and Salary (an optional real number field).

```
CREATE TABLE Employee
(
EmpID       INTEGER NOT NULL, PRIMARY KEY,
EmpName     VARCHAR(20) NOT NULL,
HireDate    DATE,
Salary      CURRENCY
)
```

Data types

Some of the most commonly used data types are described in the table below. (The data types vary depending on the specific implementation.)



Data type	Description	Example
CHAR(n)	Character string of fixed length n	ProductCode CHAR(6)
VARCHAR(n)	Character string variable length, max. n	Surname VARCHAR(25)
BOOLEAN	TRUE or FALSE	ReviewComplete BOOLEAN
INTEGER, INT	Integer	Quantity INTEGER
FLOAT	Number with a floating decimal point	Length FLOAT (10,2) (maximum number of digits is 10 and maximum number after decimal point is 2)
DATE	Stores Day, Month, Year values	HireDate DATE
TIME	Stores Hour, Minute, Second values	RaceTime TIME
CURRENCY	Formats numbers in the currency used in your region	EntryFee £23.50

Altering a table structure

The ALTER TABLE statement is used to add, delete or modify columns (i.e. fields) in an existing table:

To add a column (field):

```
ALTER TABLE Employee
ADD Department VARCHAR(10)
```

To delete a column:

```
ALTER TABLE Employee
DROP COLUMN HireDate
```

To change the data type of a column:

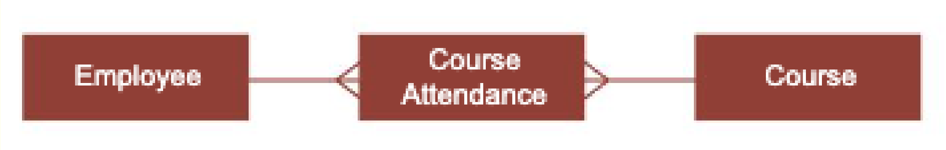
```
ALTER TABLE Employee
MODIFY COLUMN EmpName VARCHAR(30)NOT NULL
```

Defining linked tables

If you set up several tables, you can link tables by creating foreign keys.

Suppose that an extra table is to be added to the Employee database which lists the training courses offered by the company.

A third table shows which date an employee attended a particular course.



The structure of the Employee table is:

EmpID	Integer (Primary key)
Name	30 characters maximum
HireDate	Date
Salary	Currency
Department	30 characters maximum

Course Table and CourseAttenance

The structure of the Course table is:

CourseID 6 characters, fixed length (Primary key)

CourseTitle 30 characters maximum (must be entered)

OnSite Boolean

The structure of the CourseAttendance table is:

CourseID   6 characters, fixed length (foreign key)

EmpID   Integer (foreign key) Course ID and EmpID form a composite primary key

CourseDate   Date (note that the same course may be run several times on different dates)

The CourseAttendance table is created using the SQL statements:

```
CREATE TABLE CourseAttendance
(
CourseID    CHARACTER(6)NOT NULL,
EmpIDI           NTEGER NOT NULL,
CourseDate       DATE,
FOREIGN KEY      CourseID REFERENCES Course(CourseID),
FOREIGN KEY      EmpID REFERENCES Employee(EmpID)
PRIMARY KEY      (CourseID, EmpID)
)
```

Inserting, updating, and deleting data using SQL

The SQL INSERT INTO statement
This statement is used to insert a new record in a database table. The syntax is:

```
INSERT INTO tableName (column1, column2, ...)
VALUES (value1, value2, ...)
```

Example: add a record for employee number 1122, Bloggs, who was hired on 1/1/2001 for the technical department at a salary of £18000.

```
INSERT INTO Employee (EmpID, Name, HireDate, Salary, Department)
VALUES ("1122", "Bloggs", #1/1/2001#, 18000, "Technical")
```

Note that if all the fields are being added in the correct order you would not need the field names in the brackets above to be specified. INSERT INTO Employee would be sufficient

Example: add a record for employee number 1125, Cully, who was hired on 1/1/2001. Salary and Department are not known.

```
INSERT INTO Employee (EmpID, Name, HireDate)
VALUES ("1125", "Cully", #1/1/2001#)
```

The SQL UPDATE statement

This statement is used to update a record in a database table. The syntax is:

```
UPDATE tableName
SET column1 = value1, column2 = value2, …
WHERE columnX = value
```

Example: increase all salaries of members of the Technical department by 10%

```
UPDATE Employee
SET Salary = Salary*1.1
WHERE Department = "Technical"
```

Example: Update the record for Bloggs, who has moved to Administration.

```
UPDATE Employee
SET Department = "Administration"
WHERE EmpID = "1122"
```

The SQL DELETE statement

This statement is used to delete a record from a database table.

The syntax is:

```
DELETE FROM tableName
WHERE columnX = value
```

Example: Delete the record for Bloggs.

```
DELETE FROM Employee
WHERE EmpID = "1122"
```