DATABASE DEsign and development

National 5 Computing Science

Alison Cowie

Portobello High School, Edinburgh

Contents

[Analysis 3](#_Toc498235826)

[Design 4](#_Toc498235827)

[Data Protection Act 1998 4](#_Toc498235828)

[Entities 5](#_Toc498235829)

[Attributes 5](#_Toc498235830)

[Relationships 5](#_Toc498235831)

[Entity relationship diagrams 6](#_Toc498235832)

[Data dictionary 9](#_Toc498235833)

[Designing a solution to a query 13](#_Toc498235834)

[Example designs 16](#_Toc498235835)

[Implementation 17](#_Toc498235836)

[Referential integrity 17](#_Toc498235837)

[SQL. 18](#_Toc498235838)

[Testing 20](#_Toc498235839)

[Evaluation 20](#_Toc498235840)

## Analysis

The development of a database has several phases. These include:

**1. Analysis**

**2. Design**

**3. Implementation**

**4. Testing**

**5. Evaluation**

For example, Mr McLeod is Depute at Craigfar High School. He wants to make a database of school activities (clubs) so that, at any point in time, he can print out a report about any activity and which teacher runs it

Analysis: Mr McLeod meets with Ms Lyall who knows how to make databases. They create a document called the **database specification** that lists the **end user requirements** and the **functional requirements** of the database

**End user requirements**

* the tasks that the end user (Mr McLeod) expects to be able to do using the database

**Functional requirements**

* the processes and activities that the system has to perform
* information that the system has to contain to be able to carry out its functions

Design:

Ms Lyall will consider the **implications of the Data Protection Act** for the database she is about to make

She will draw the **relationship between the entities** that will appear in the database. At the moment the two entities are Activity and Teacher. Then she will draw up a **data dictionary** for the database and **plan the scripts** (programs) for all the operations that Mr McLeod requires. Some of the scripts will be automated within the database software she is going to use so she won’t need to write all the code

Implementation: Once the design (the plan) for the database is complete, Ms Lyall will create the tables for each entity set in the database, create the fields for each table and include their type, size and any appropriate validation. Then she will populate the database with actual information (the relevant details of the activities and teachers). This might mean typing in the information from scratch or being able to import it from other files e.g. teacher details. Then she will include program code to carry out the operations that Mr McLeod has asked for - the end user and functional requirements of the database

Testing: Ms Lyall **will be testing the database continually** while she is working. When she is finished, Mr McLeod will be involved in testing it to make sure that it works as he specified. This may involve going back to previous phases and making changes (iteration). Once it is complete, Ms Lyall will arrange for the database to installed in such a way that Mr McLeod can use it

Evaluation: This is to decide whether the database is **fit for purpose**. That means: Does it meet its **end-user** and **functional requirements**? Is it **robust** e.g. is there **referential integrity?** Is it **reliable** in that the **output is accurate**?

## Design

### Data Protection Act 1998

Companies and organisations like the government hold a lot of personal data about individuals. To protect this data and people’s rights when storing, transmitting and exporting it, the Data Protection Act was passed by Parliament to protect peoples’ privacy

In short the law states that:

* individuals will be asked if data may be held about them (prior consent)
* data should be accurate - if data held about a data subject is incorrect they can demand it is changed
* data should be used only for limited, specifically stated purposes
* the data user is obliged to keep the data safe, secure and up to date, and not to hold the data for longer than is necessary

**Data Subjects** are the individuals who have data kept about them. If a data subject wishes to inspect data held about themselves they can request to see their own personal data

**Exceptions to the Act**

If any data is held by the police, security forces or the Inland Revenue, then you are denied access to the data

**Data Users** are the employees that work for organisations/companies that have access to / make use of the information held about individuals

**Data Controllers** are the companies, organisations or people who store personal information about individuals. Data Controllers must be registered with the **Data Protection Commissioner**. Data Controllers must apply for permission to store data and state what data they want to store and for what purpose

**Data Protection Commissioner** is the person you contact if you wish to raise a complaint about the accuracy of data. They in turn contact the Data Protection Registrar who oversees the administration of the act

**The Data Protection Register** is an information system freely available to the public that holds information about who holds information about members of the public

### Entities

An **entity** is a single person, place, or thing about which data can be stored e.g. a car, Spain, flight 4A07

### Attributes

An entity will have attributes. In the example below, the car entity has four attributes: make, model, colour, registrationNumber

make: Vauxhall

model: Zafira

colour: grey

registrationNumber: S012 GHJ

The attributes have values that can be edited.

When the data is entered into a database, the attributes become the **fields** of a **record**. If we made a database of 1200 cars, we would have 1200 records, each with four fields

### Relationships

We are surrounded by entities that have relationships with each other. The relationships can be described as

* ONE to ONE
* ONE to MANY
* MANY to MANY

For National 5 we only look at ONE to MANY relationships

For example, ONE car has MANY repairs. You might think – but many cars have many repairs – but that is not what the relationship is saying. It means that one specific repair (for example, the brake pads on Vauxhall Zafira SO12 GHJ replaced on 5.3.17) was for ONE car but that ONE car might have MANY different (unique) repairs

has

CAR

REPAIR

For example, the relationship between mother and children is ONE to MANY. You might think – but many mothers have many children – but that is not what the relationship is saying. It means that one specific child (for example, Zoltan Albard born on 12.8.2001) is the child of ONE mother but that ONE mother might have MANY different (unique) children

has

CHILD

MOTHER

### Entity relationship diagrams

A flat file database is shown below – it has 9 fields. It would be better if the database was split into two linked tables with a relationship created between them otherwise modifying duplicated data can lead to errors. For example, if Miss Chess changed her name to Mrs Board, every entry for her would have to be amended which could easily lead to errors (loss of data integrity)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***activityID*** | ***title*** | ***ageRange*** | ***venue*** | ***teacherID*** | ***Name*** | ***teachingRoom*** | ***subject*** | ***fullTime*** |
| *ASC2745* | *Revision* | *11-18* | *Room 101* | *T87* | *Miss Chess* | *214* | *Maths* | *false* |
| *Susp1024* | *Water sports* | *14-18* | *Porty beach* | *T56* | *Mr Splash* | *303* | *PE* | *true* |
| *Susp1013* | *Adventure* | *11-15* | *Crazy Ways* | *T14* | *Ms Yeehah* | *504* | *Geography* | *true* |
| *Susp1067* | *Make a mural* | *11-18* | *PHS* | *T39* | *Mrs Colourist* | *612* | *Art* | *false* |
| *Susp1018* | *Cycling* | *16-18* | *Glen Tress* | *T44* | *Mr Spokes* | *405* | *CDT* | *true* |
| *ASC2543* | *Board games* | *11-14* | *Room 102* | *T87* | *Miss Chess* | *214* | *Maths* | *false* |
| *ASC2716* | *Cycling* | *14-17* | *PHS* | *T44* | *Mr Spokes* | *405* | *CDT* | *true* |
| *ASC3487* | *Adventure* | *15-18* | *PHS* | *T56* | *Mr Splash* | *303* | *PE* | *true* |

If the data shown in the flat file database above was held in two tables as a relational database, the **entity relationship** would be shown as follows:

ONE teacher offers MANY activities

offers

TEACHER

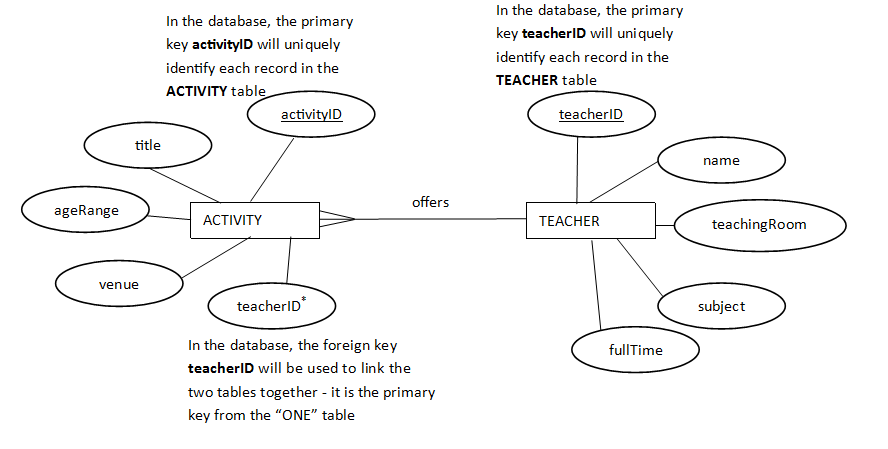
ACTIVITY

N.b.

the word describing the relationship is written into the diagram i.e. “offers”

the “crow’s foot” shows the “many” aspect

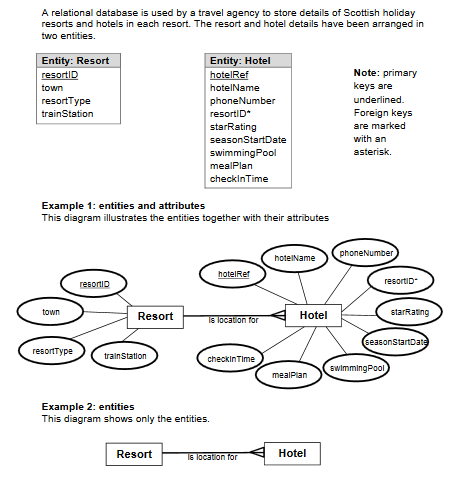
The full entity relationship diagram (ERD) would be:



n.b. It is useful to remember:

“the primary key from the ONE table goes into the MANY table as the foreign key”

Here’s another example:



### Data dictionary

A data dictionary should include the following:

#### Entity names

For the school activities example, the entity names are Activity and Teacher

#### Attribute names

The attribute names for the school activities example are:

*activityID*, *title*, a*geRange*, v*enue*, *teacherID*, *name*, *teachingRoom*, *subject*, *fullTime*

#### Key - Primary or foreign key

***Primary key:*** this is the attribute that **uniquely identifies each entity** e.g. accountNumber, ID, carRegNumber, etc.

In the school activities example, the primary keys are activityID for the Activity table and teacherID for the Teacher table

***Foreign key:*** this is used to link one table to another table in a database i.e. set up relationships between tables. N.b. the foreign key in one of the tables will be the primary key in another table

Helpful hint: “the primary key from the ONE table is used as the foreign key in the MANY table”

#### Attribute (field) types

##### Text

e.g. EH9 1EF, 07652 915845, Jane

(N.b. a telephone number is text – it has a lead zero, may contain a space and would not be used in calculations)

##### Number

e.g. 95, 0.645 Also £32.95 (formatted as currency)

##### Date

e.g. 15/04/2017 (formatted as dd/mm/yyyy)

##### Time

e.g. 23:30, 06:00 (formatted as hh:mm)

##### Boolean

true/false

#### Attribute size

For example, a text attribute might be maximum 20 characters in length

#### Validation

Validation is the process of checking that data entered into a system is of the **correct type and structure**

1. Presence check

checks that data has been entered/data field not left blank

1. Restricted choice

the user presented with a list of options to choose from e.g. from a drop-down menu

1. Field length check

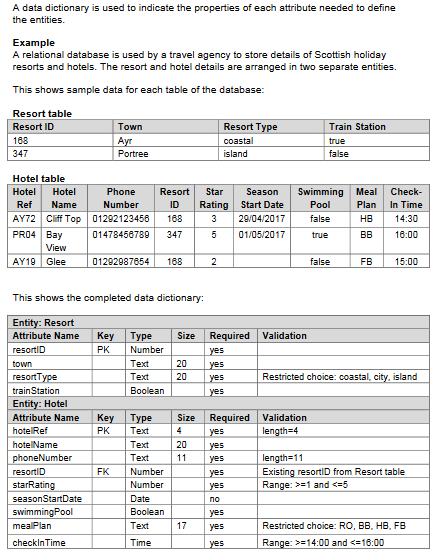
specifies the size of a field e.g. number of characters allowed in a field

1. Range check

specifies the range of values a field can hold e.g. range 0 to 100 inclusive

Altogether, the data dictionary might look like this:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Entity** | **Attribute name** | **Key** | **Type** | **Size** | **Unique** | **Required** | **Validation** | **Sample data** |
| ***Activity*** | ***activityID*** | *PK* | *Text* | *8* | *Yes* | *Yes* | *Presence check* | *ASC2745* |
| ***title*** |  | *Text* | *20* |  | *Yes* |  | *Revision* |
| ***ageRange*** |  | *Text* | *5* |  |  |  | *11-18* |
| ***venue*** |  | *Text* | *15* |  | *Yes* |  | *Room 101* |
| ***teacherID*** | *FK* | *Lookup* |  | *No* | *Yes* | *Lookup from TEACHER* | *T87* |
|  |  |  |  |  |  |  |  |  |
| ***Teacher*** | ***teacherID*** | *PK* | *Text* | *4* | *Yes* | *Yes* | *Presence check* | *T87* |
| ***name*** |  | *Text* | *40* |  |  | *Restricted choice* | *Miss Chess* |
| ***teachingRoom*** |  | *Text* | *3* |  | *Yes* | *Field length check* | *214* |
| ***subject*** |  | *Text* | *20* |  |  |  | *Maths* |
| ***fullTime*** |  | *Boolean* |  |  |  |  | *false* |



Here’s another example:

N.b. The foreign key is not unique in the MANY table. For example, there may be many orders with the same customerID on them

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Entity** | **Attribute name** | **PK/FK** | **Type (size)** | **Unique** | | **Required** | **Validation** | **Format** | **Sample data** |
| ***Customer*** | ***customerID*** | *PK* | *Text (8)* | | *Yes* | *Yes* | *Presence check* |  | *C01456AR* |
| ***title*** |  | *Text (3)* | |  |  | *Restricted choice* |  | *Mrs* |
| ***name*** |  | *Text (30)* | |  | *Yes* |  |  | *Amy Bryce* |
| ***address*** |  | *Text (150)* | |  | *Yes* |  |  | *16 Main St, Craigfar, CK9 1RF* |
| ***age*** |  | *Number* | |  |  | *Range check*  *(16 to 100)* | *000* | *37* |
| ***Order*** | ***orderID*** | *PK* | *Text (4)* | | *Yes* | *Yes* | *Presence check* |  | *T87* |
| ***orderDate*** |  | *Date* | |  |  | *Restricted choice* | *dd/mm/yy* | *12/05/17* |
| ***totalToPay*** |  | *Number* | |  | *Yes* | *Field length check* | *£0000.00* | *£36.95* |
| ***customerID*** | *FK* | *Lookup* | | *No* | *Yes* | *Lookup from CUSTOMER* |  | *C01456AR* |

### Designing a solution to a query

#### Multiple tables

Suppose the flat file database shown here …

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***activityID*** | ***title*** | ***ageRange*** | ***venue*** | ***teacherID*** | ***Name*** | ***teachingRoom*** | ***subject*** | ***fullTime*** |
| *ASC2745* | *Revision* | *11-18* | *Room 101* | *T87* | *Miss Chess* | *214* | *Maths* | *false* |
| *Susp1024* | *Water sports* | *14-18* | *Porty beach* | *T56* | *Mr Splash* | *303* | *PE* | *true* |
| *Susp1013* | *Adventure* | *11-15* | *Crazy Ways* | *T14* | *Ms Yeehah* | *504* | *Geography* | *true* |
| *Susp1067* | *Make a mural* | *11-18* | *PHS* | *T39* | *Mrs Colourist* | *612* | *Art* | *false* |
| *Susp1018* | *Cycling* | *16-18* | *Glen Tress* | *T44* | *Mr Spokes* | *405* | *CDT* | *true* |
| *ASC2543* | *Board games* | *11-14* | *Room 102* | *T87* | *Miss Chess* | *214* | *Maths* | *false* |
| *ASC2716* | *Cycling* | *14-17* | *PHS* | *T44* | *Mr Spokes* | *405* | *CDT* | *true* |
| *ASC3487* | *Adventure* | *15-18* | *PHS* | *T56* | *Mr Splash* | *303* | *PE* | *true* |

… has been implemented as a **relational database** with **two tables** of **records** as shown here…

activityID: ASC2745

title: Revision

age Range: 11-18

venue: Room 101

teacher ID: T87

activityID: Susp1024

title: Water sports

age Range: 14-18

venue: Porty beach

teacher ID: T56

activityID: Susp1013

title: Adventure

age Range: 11-15

venue: Crazy Ways

teacher ID: T14

activityID: Susp1067

title: Make a mural

age Range: 11-18

venue: PHS

teacher ID: T39

activityID: Susp1018

title: Cycling

age Range: 16-18

venue: Glen Tress

teacher ID: T44

activityID: ASC2543

title: Board games

age Range: 11-14

venue: Room 102

teacher ID: T87

activityID: ASC2716

title: Cycling

age Range: 14-17

venue: PHS

teacher ID: T44

activityID: ASC3487

title: Adventure

age Range: 15-18

venue: PHS

teacherID: T56

teacherID: T18

name: Mr Donne

teaching Room: 100

subject: English

full Time: Yes

teacherID: T87

name: Miss Chess

teaching Room: 214

subject: Maths

full Time: No

teacherID: T62

name: Mr Kildare

teaching Room: 101

subject: History

full Time: Yes

teacherID: T56

name: Mr Splash

teaching Room: 303

subject: PE

full Time: Yes

teacherID: T14

name: Ms Yeehah

teaching Room: 504

subject: Geography

full Time: Yes

teacherID: T39

name: Mrs Colourist

teaching Room: 612

subject: Art

full Time: No

teacherID: T75

name: Miss Overby

teaching Room: 518

subject: HE

full Time: No

teacherID: T28

name: Mr Spokes

teaching Room: 405

subject: CDT

full Time: Yes

teacherID: T44

name: Mr Spokes

teachingRoom: 405

subject: CDT

fullTime: true

Mr McLeod wants to create a **report** that shows the names (in **alphabetical order**) of all the **full time** teachers who offer **adventure** activities. Which teachers would the report show?

The report would show *Mr Splash and Ms Yeehah*

There is a programming language called **SQL** that is used to query, insert, update and modify data in a database. SQL stands for **Structured Query Language**.

First step: **design a solution to the SQL query**. Here is an example:

|  |  |
| --- | --- |
| **Field(s)** | name, fullTime, title |
| **Table(s)** | Teacher, Activity |
| **Search criteria** | fullTime = true AND title = “Adventure” |
| **Sort order** | name ASC |

#### Fields and tables

When a database is created using software the entity sets become the tables, the attributes become the fields and each entity becomes a record in a table

|  |  |
| --- | --- |
| **Design stage** | **Implementation stage** |
| Entity set | Table |
| Attribute | Field |
| Entity | Record |

#### Search criteria

Simple search (find)- searching on one field only

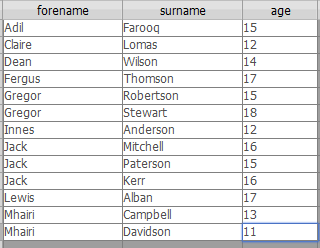
e.g. search for records with field (forename = “James”)

Complex search - searching on two or more fields

e.g. (surname = “McRury”) AND (purchased = true)

#### Sort order

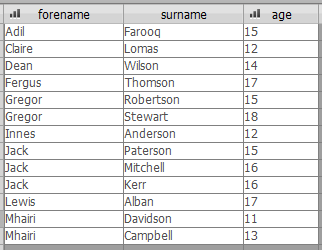
Sort in alphabetical, numerical, chronological (time) order (ascending/descending)



This table has been sorted on **one** field

It has been sorted on the field **forename** in alphabetical ascending order (A to Z). You can see that there are two Gregors, three Jacks and two Mhairis but, within those groups, no other field has been sorted – the Gregor and Mhairi surnames are in order but the Jack surnames aren’t and similarly with their ages

Sorting on more than one field: sorts on a first field, then sorts within the groups on the second field

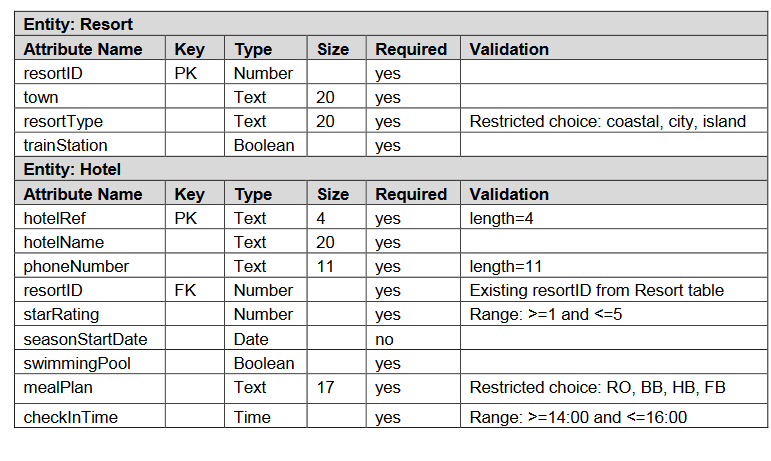


This table has been sorted on two fields (attributes) – first on forename in alphabetical ascending order and then on age in numerical ascending order

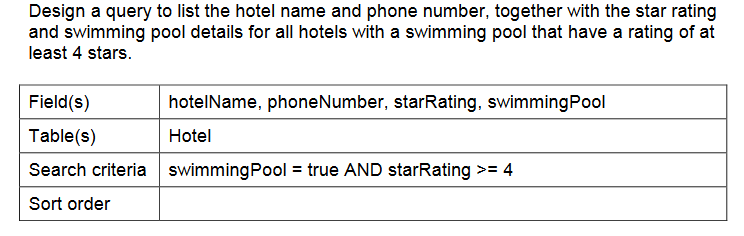
How to tell: if you look at the Gregors, Jacks and Mhairis, their surnames are sometimes ascending, sometimes descending but their ages (in their little groups) are all increasing

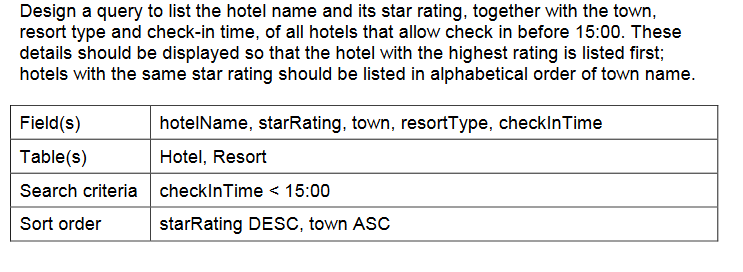
### Example designs

Data dictionary



SQL query design examples

1.

2.

## Implementation

Advantages of a relational database over a flat-file database:

In the **flat file** database shown previously, if Miss Chess changed her name to Mrs Chequer, then someone would have to change every entry for Miss Chess. They are likely to make mistakes and, especially in a large database, this would quickly lead to the data becoming **inaccurate** and **unreliable** (a loss of data integrity)

In the **relational** database, there would be just one record for Miss Chess in the Teacher table. There would be two activities linked to her record by having her teacherID on each activity record and, if her name needed to be amended (e.g. she became Mrs Chequer), then only the record about her would need to be amended and all the activities linked to her would automatically see her new name

So the advantage of a **relational database with linked tables** is that it **avoids modification errors for data that is duplicated** that would occur if the data was held in a flat database. Links can be made from data in one table to data that already exists in another table to provide different pathways through the data. The implementation should ensure that the tables are linked together in the most efficient and useful way so that any necessary reports can be generated

### Referential integrity

Integrity means trustworthiness

A database has **referential** integrity if the table relationships are consistent. In other words, any foreign key field must agree with the primary key that is referenced by the foreign key. If a foreign key has a value that doesn’t exist as a primary key value in the linked table then the database will produce errors resulting from a lack of referential integrity

For example, in the schools activities database, if one of the activity records was trying to reference a teacher record with a teacherID that didn’t actually exist in the Teacher table then there would be a lack of referential integrity

### 

### SQL

There is a programming language called **SQL** that is used to query, insert, update and modify data in a database. It stands for **Structured Query Language**. The following examples have been copied from W3Schools.com

#### SELECT

FROM

For example: SELECT customerName, city

FROM Customers;

customerName, city are **fields** in the **table** called Customers

WHERE

For example: SELECT customerName

FROM Customers

WHERE country = 'Mexico ';

customerName and country are **fields** in the **table** called Customers

AND, OR, <, >, =

For example:

SELECT \* // \* means all the records

FROM Customers  
WHERE country='Germany' AND city='Berlin';

SELECT \*

FROM Customers  
WHERE city='Berlin' OR city='München';

SELECT name

FROM Students

WHERE age >= 14 AND age <= 18;

ORDER BY (max of two fields)

For example:

SELECT \*

FROM Customers  
ORDER BY country DESC;

SELECT \*

FROM Customers  
ORDER BY country ASC, customerName DESC;

#### INSERT

For example:

INSERT INTO Customers (customerName, city, country)  
VALUES ('Cardinal', 'Stavanger', 'Norway');

#### UPDATE

For example:

UPDATE Customers  
SET contactName = 'Alfred Schmidt', city= 'Frankfurt'  
WHERE customerID = 1;

#### DELETE

For example:

DELETE FROM Customers  
WHERE customerName='Alfreds Futterkiste';

#### INNER JOIN (Equi-join between tables)

For example:

SELECT Orders.orderID, Customers.customerName  
FROM Orders  
INNER JOIN Customers ON Orders.customerID = Customers.customerID;

For example:

SELECT Teacher.name

FROM Teacher

INNER JOIN Activity //this makes the link between the two tables

ON Teacher.teacherID = Activity.teacherID

WHERE (Teacher.fullTime = true) AND (Activity.title = “Adventure”);

## Testing

This would involve testing the SQL operations to ensure that they work correctly and produce the correct output

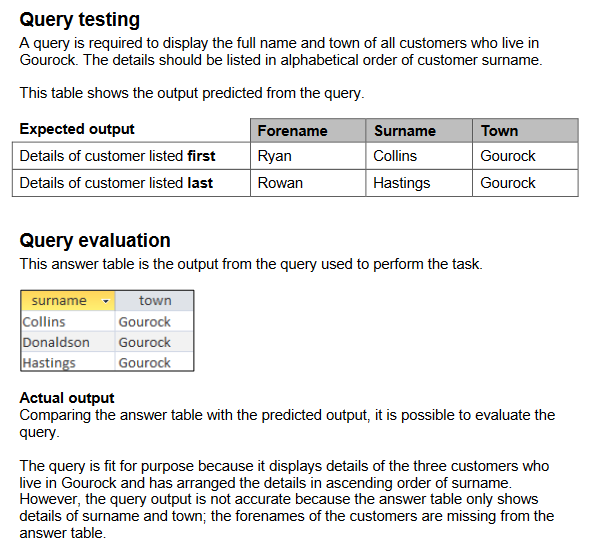
A test table could be drawn up showing the test data, the expected output and the actual output in the form of screenshots. This test table could be referred to in support of your evaluation

## Evaluation

Finally, the complete database is evaluated in terms of whether it is **fit for purpose**. This means making a judgement as to whether it meets its **specification** by delivering its **functional requirements** (see Analysis). You would also judge whether it meets the **end-user requirements**. You would consider your test results and draw conclusions from them.

For example, you might say “The functional requirements Mr McLeod asked for was a button that would let him search for an activity. The search was to result in a report (display) of the activity details. He also wanted a button that let him search for a teacher. The search was to result in a report (display) of all the activities that the teacher is running. My database allowed Mr Mcleod to carry out these operations and the **output was accurate** (see test results) so my database is **fit for purpose** in that sense. However, my user interface was unnecessarily complicated for Mr McLeod (a novice user) to use so I would say that it did not fully meet the end-user requirements”

Here is another example:



### 

### 