Databases

iGCSE Computer Science

What is a database?

An organised, structured collection of related information.

Similar to a spreadsheet, information is organised into rows and columns. In fact, spreadsheets are commonly used as simple databases.

Some terminology

- Table
- Record
- Field
- Primary key
- Relational databases
- Foriegn key
- Composite key
- Data type

Table

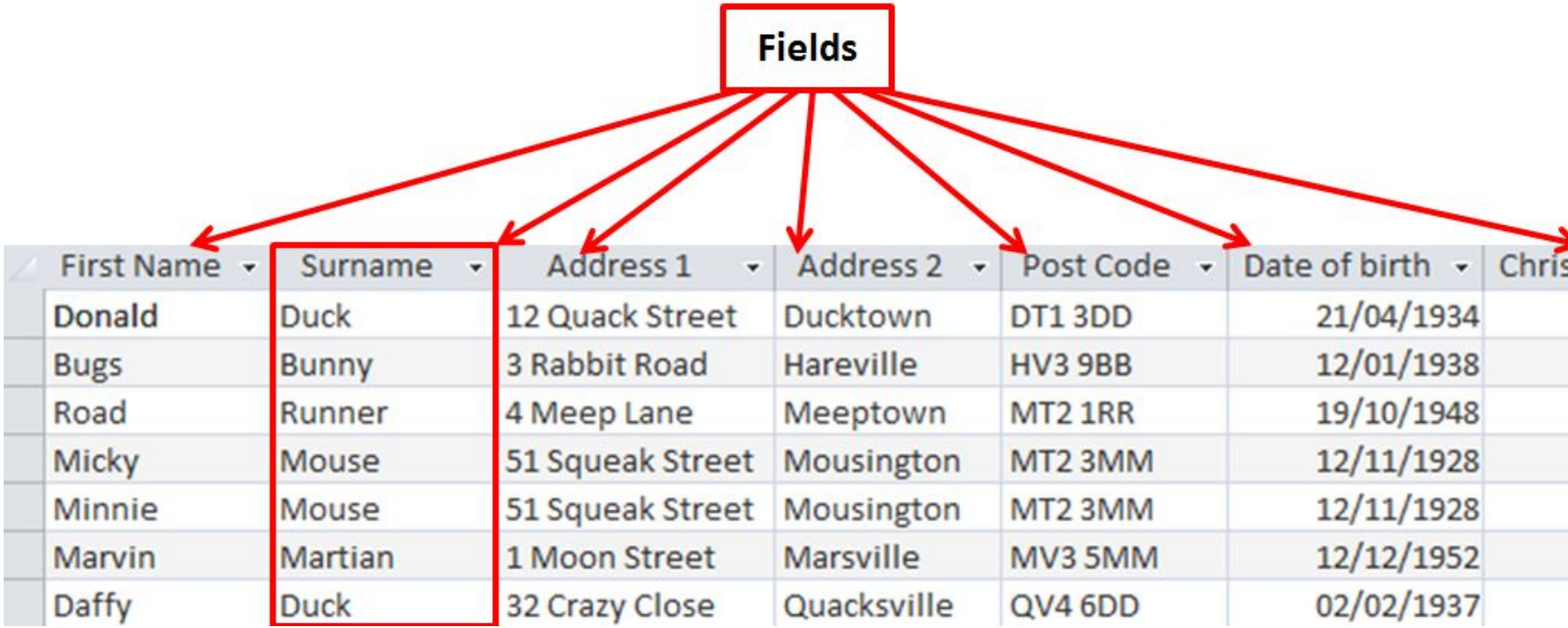
A collection of data is known as a table.

First Name	Surname	Address 1	Address 2	Post Code	Date of birth	Christmas Card
Donald	Duck	12 Quack Street	Ducktown	DT1 3DD	21/04/1934	☐
Bugs	Bunny	3 Rabbit Road	Hareville	HV3 9BB	12/01/1938	☑
Road	Runner	4 Meep Lane	Meeptown	MT2 1RR	19/10/1948	☑
Micky	Mouse	51 Squeak Street	Mousington	MT2 3MM	12/11/1928	☐
Minnie	Mouse	51 Squeak Street	Mousington	MT2 3MM	12/11/1928	☐
Marvin	Martian	1 Moon Street	Marsville	MV3 5MM	12/12/1952	☑
Daffy	Duck	32 Crazy Close	Quacksville	QV4 6DD	02/02/1937	☑

Record

First Name	Surname	Address 1	Address 2	Post Code	Date of birth	Christmas Card
Donald	Duck	12 Quack Street	Ducktown	DT1 3DD	21/04/1934	☐
Bugs	Bunny	3 Rabbit Road	Hareville	HV3 9BB	12/01/1938	☑
Road	Runner	4 Meep Lane	Meeptown	MT2 1RR	19/10/1948	☑
Micky	Mouse	51 Squeak Street	Mousington	MT2 3MM	12/11/1928	☐
Minnie	Mouse	51 Squeak Street	Mousington	MT2 3MM	12/11/1928	☐
Marvin	Martian	1 Moon Street	Marsville	MV3 5MM	12/12/1952	☑
Daffy	Duck	32 Crazy Close	Quacksville	QV4 6DD	02/02/1937	☑

Field



First Name	Surname	Address 1	Address 2	Post Code	Date of birth	Christmas Card
Donald	Duck	12 Quack Street	Ducktown	DT1 3DD	21/04/1934	☐
Bugs	Bunny	3 Rabbit Road	Hareville	HV3 9BB	12/01/1938	☑
Road	Runner	4 Meep Lane	Meeptown	MT2 1RR	19/10/1948	☑
Micky	Mouse	51 Squeak Street	Mousington	MT2 3MM	12/11/1928	☐
Minnie	Mouse	51 Squeak Street	Mousington	MT2 3MM	12/11/1928	☐
Marvin	Martian	1 Moon Street	Marsville	MV3 5MM	12/12/1952	☑
Daffy	Duck	32 Crazy Close	Quacksville	QV4 6DD	02/02/1937	☑

Primary key

A field that uniquely identifies one record and only one record.

Note in the example why names are not appropriate.

Database primary keys are to blame for all the numbers assigned to us.

Primary Key

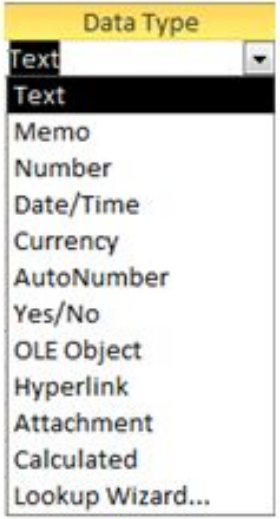
Employee ID	SURNAME	GIVEN NAME	MIDDLE NAME
8001000000	Smith	Jennifer	Abad
8001000001	Smith	John Nhiel	Galvez
8001000002	Dela Cruz	RJ	Prachaya
8001000003	Reyes	Gab	Ugalino
8001000004	Doe	RJ	Mendoza
8001000005	Licauco	David	Galvez

Data types

The datatypes available vary by database system.

For the igcse, you only need to consider:

- TEXT
- NUMBER
- DATE/TIME
- CURRENCY
- BOOLEAN



Field Name	Data Type
First Name	Text
Surname	Text
Address 1	Text
Address 2	Text
Post Code	Text
Date of birth	Date/Time
Christmas Card	Yes/No

Exercise

Scenario:

Imagine you have been tasked to create a database for a **School assessment/reports system**.

You need to record information about:

- Students
- Classes
- Their assessments
- Their grades

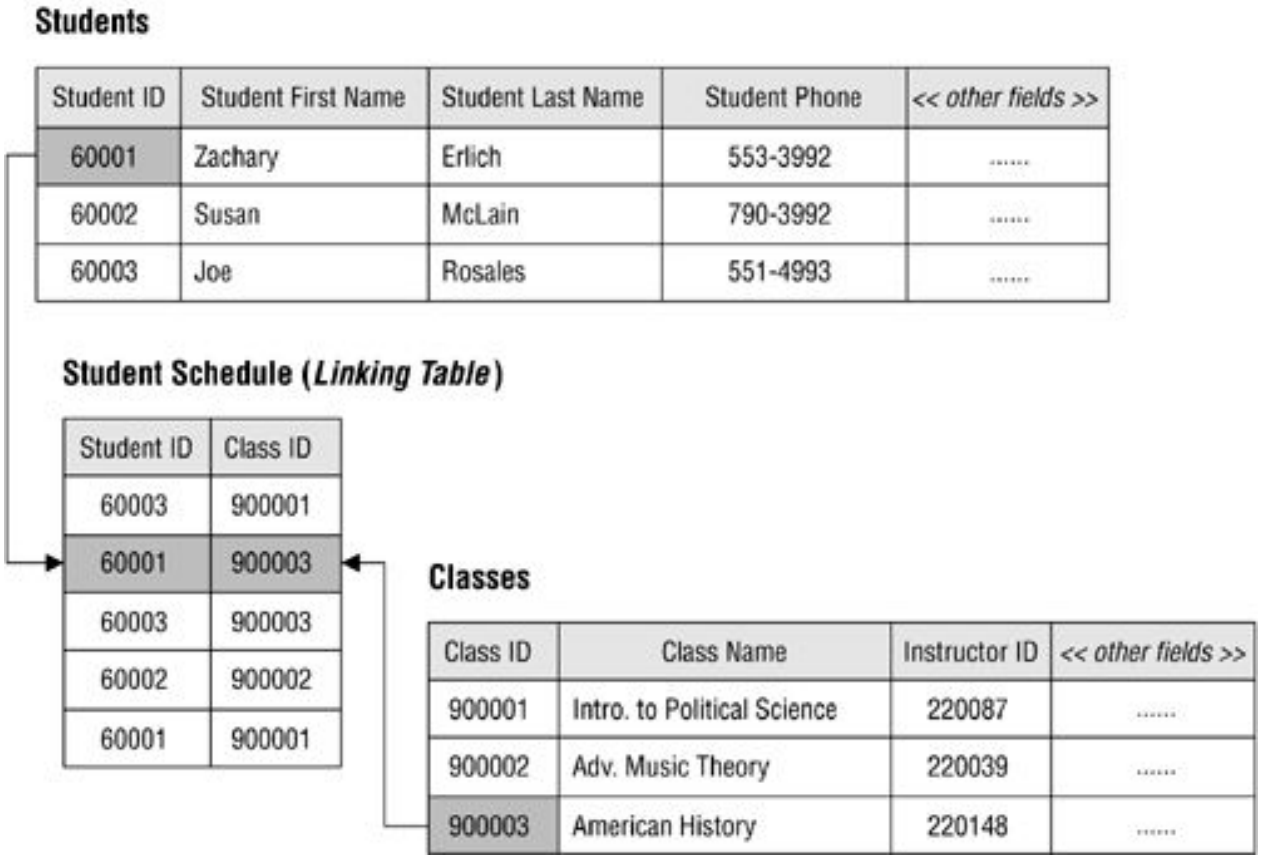
...to be able to produce the end of term/semester reports.

Identify the fields and datatypes you anticipate needing for this scenario. It may help to imagine example outputs/printouts the system may require.

Relational databases

Relational databases

The power of databases comes from being able to **link** information from one table to **related** information in other tables.



Students

Student ID	Student First Name	Student Last Name	Student Phone	*<< other fields >>*
60001	Zachary	Erlich	553-3992	.....
60002	Susan	McLain	790-3992	.....
60003	Joe	Rosales	551-4993	.....

Student Schedule (*Linking Table*)

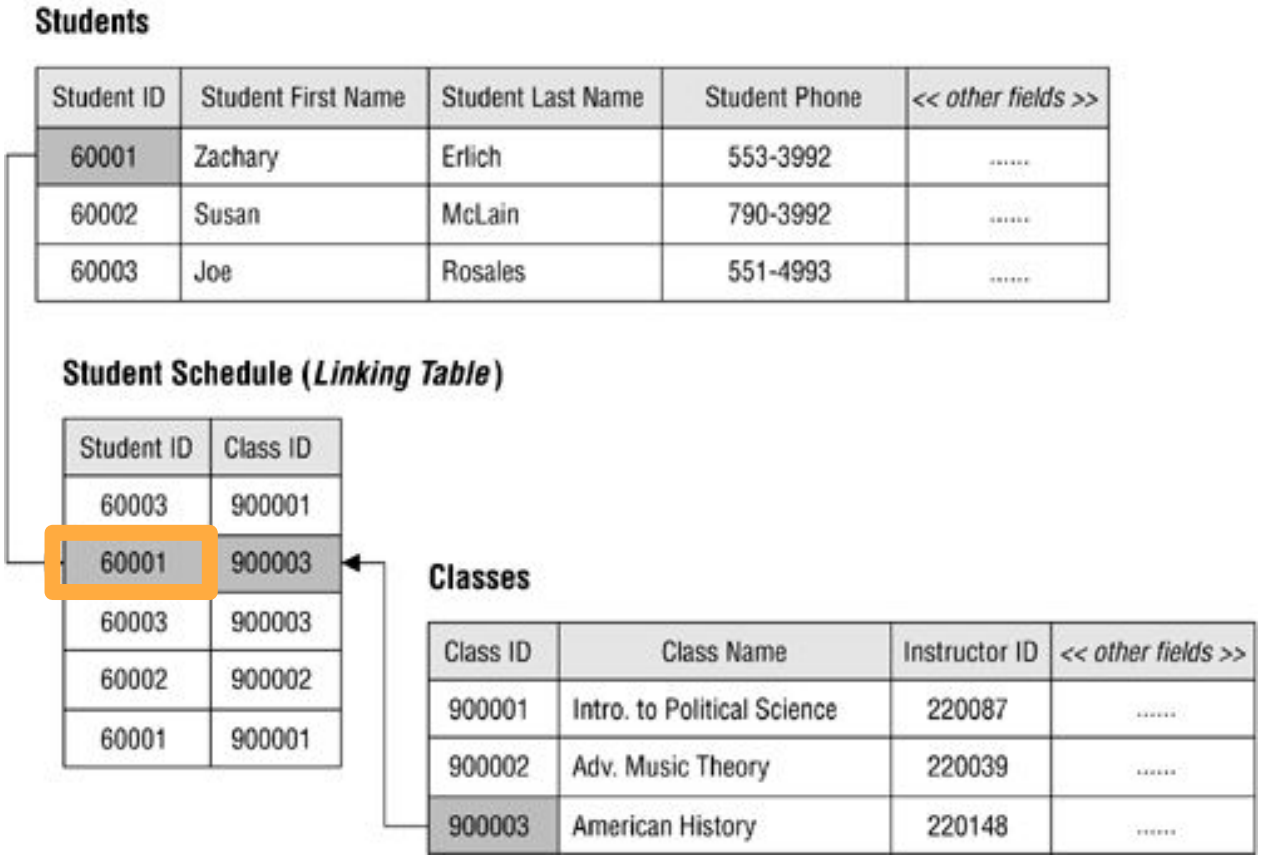
Student ID	Class ID
60003	900001
60001	900003
60003	900003
60002	900002
60001	900001

Classes

Class ID	Class Name	Instructor ID	*<< other fields >>*
900001	Intro. to Political Science	220087	.....
900002	Adv. Music Theory	220039	.....
900003	American History	220148	.....

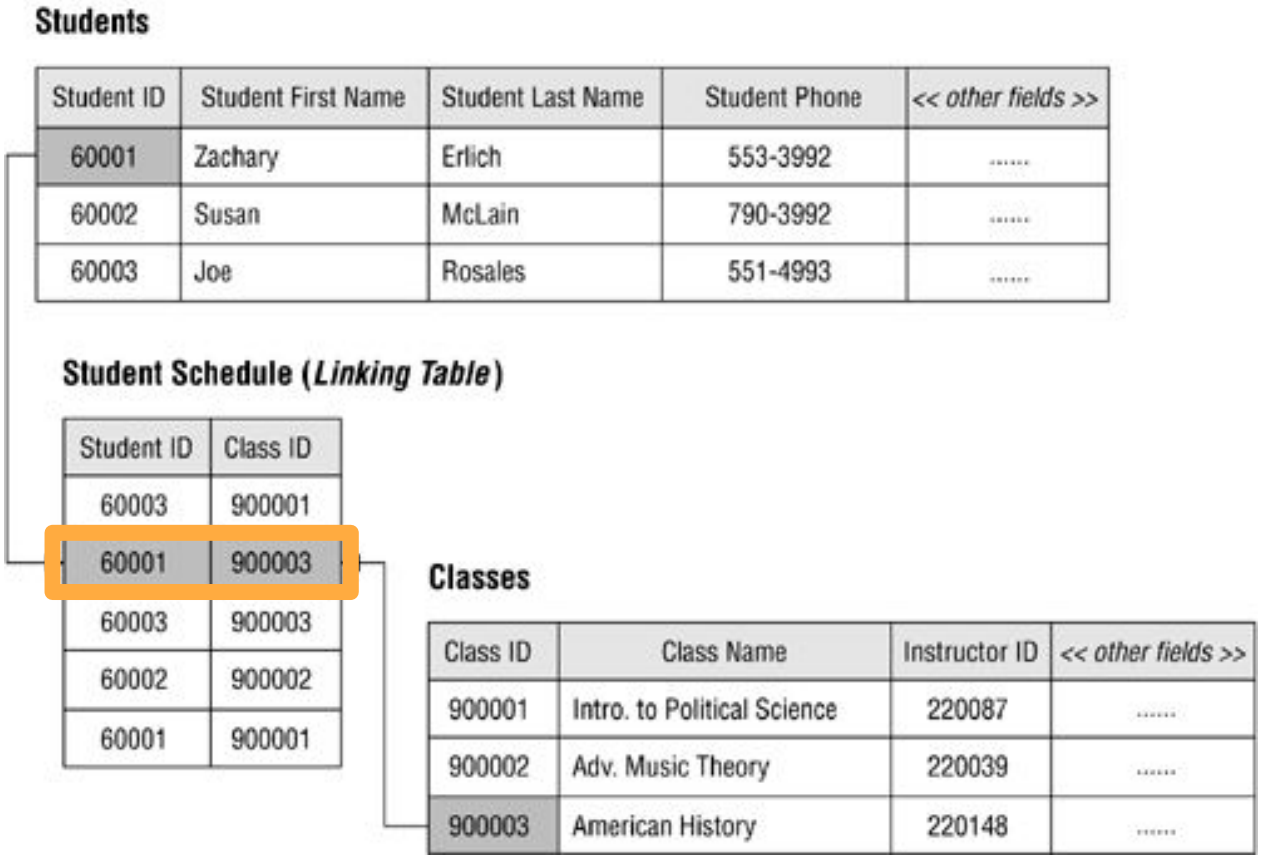
Foreign key

Links to the primary key of an alternate table.

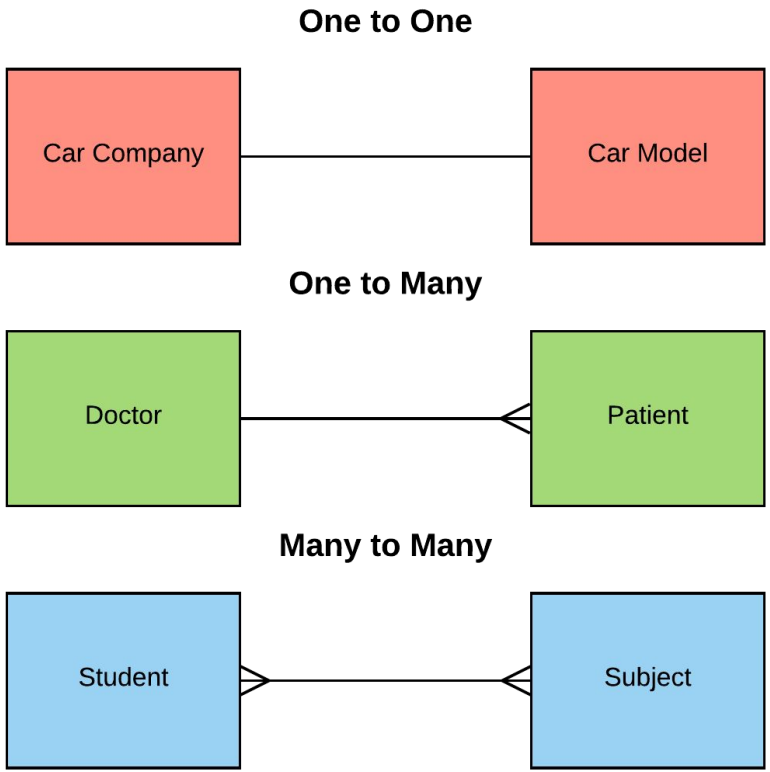


Composite key

Multiple fields being combined to act as the primary key.



Relationship types



Normalising data

Normalising data

Consider the following data

StudentID	Name	Subjects	Teachers	Grades
101	Alex	Physics, Chemistry	Murdoch, Lewis	5, 6
103	Elizabeth	Java, Math	Baumgarten, Wilson	7, 6
102	Brian	Math, English	Wilson, Pegg	3, 4

1st normal form

Each column must contain only one value.

BEFORE

StudentID	Name	Subjects	Teachers	Grades
101	Alex	Physics, Chemistry	Murdoch, Lewis	5, 6
103	Elizabeth	Java, Math	Baumgarten, Wilson	7, 6
102	Brian	Math, English	Wilson, Pegg	3, 4

AFTER

StudentID	Name	Subject	Teacher	Grade
101	Alex	Physics	Murdoch	5
101	Alex	Chemistry	Lewis	6
103	Elizabeth	Java	Baumgarten	7
103	Elizabeth	Math	Wilson	6
102	Brian	Math	Wilson	3
102	Brian	English	Pegg	4

2nd normal form

Every field is dependent on the primary key.

- Split fields into different tables as required until this is met.

StudentID	Name	Subject	Teacher	Grade
101	Alex	Physics	Murdoch	5
101	Alex	Chemistry	Lewis	6
103	Elizabeth	Java	Baumgarten	7
103	Elizabeth	Math	Wilson	6
102	Brian	Math	Wilson	3
102	Brian	English	Pegg	4

StudentID	Name
101	Alex
103	Elizabeth
102	Brian

Subject	Teacher
Physics	Murdoch
Chemistry	Lewis
Java	Baumgarten
Math	Wilson
English	Pegg

StudentID	Subject	Grade
101	Physics	5
101	Chemistry	6
103	Java	7
103	Math	6
102	Math	3
102	English	4

3rd normal form

No non key attribute is transitively dependent on the primary key ...*What?*

A is dependent on B, and B is dependent on C, then C is “transitively dependent” on A (via B).

Address records are a good example. Is everything below really dependent on the primary key?

Member ID	Name	Street	District	State	ZIP
432	Philippa	9929 Saxon Road	Maspeth	NY	11378
345	Willem	8094 Brickyard Street	Wheaton	NE	68506
456	Kajetan	2 W. Gartner St	Palmetto	IL	60187
457	Sharna	431 Bear Hill Rd.	New Albany	FL	34221
256	Rosie	775 Walt Whitman St	Naugatuck	IN	47150
374	Kiyan	8906 Old York Dr	Lincoln	CT	16770

3rd normal form

Member ID	Name	Street	District	State	ZIP
432	Philippa	9929 Saxon Road	Maspeth	NY	11378
345	Willem	8094 Brickyard Street	Wheaton	NE	68506
456	Kajetan	2 W. Gartner St	Palmetto	IL	60187
457	Sharna	431 Bear Hill Rd.	New Albany	FL	34221
256	Rosie	775 Walt Whitman St	Naugatuck	IN	47150
374	Kiyan	8906 Old York Dr	Lincoln	CT	16770

Storing the District and State is redundant because that can be determined via the ZIP code. Hence it is transitively dependent.

Member ID	Name	Street	ZIP (foriegn key)
432	Philippa	9929 Saxon Road	11378
345	Willem	8094 Brickyard Street	68506
456	Kajetan	2 W. Gartner St	60187
457	Sharna	431 Bear Hill Rd.	34221
256	Rosie	775 Walt Whitman St	47150
374	Kiyan	8906 Old York Dr	16770

ZIP	District	State
11378	Maspeth	NY
68506	Wheaton	NE
60187	Palmetto	IL
34221	New Albany	FL
47150	Naugatuck	IN
16770	Lincoln	CT

Other considerations

Try to anticipate possible uses for the data.

- You might need a formal name for use on certificates, and a casual name for use in emails or correspondance.

Design as inclusive as possible. Consider names:

- First name, given name, preferred name
- Last name, surname, full name

https://shinesolutions.com/2018/01/08/falsehoods-programmers-believe-about-names-with-examples/

Software should be built to meet your requirements while trying to anticipate strange situations. Context remains important. Address edge cases in some reasonable way.

Exercise

Previously you created a list of fields & data types for a School assessment/reporting system.

We just saw a similar example in the most recent slides. Using the principles of normalisation, how would you extend this example to include:

- Information on each individual assessment and
- Information on each student's achievement in each individual assessment

SQL: Structured query language

SQL databases

SQLite3

Download and install the DB Browser for SQLite

https://sqlitebrowser.org/

SQL: Creating a table

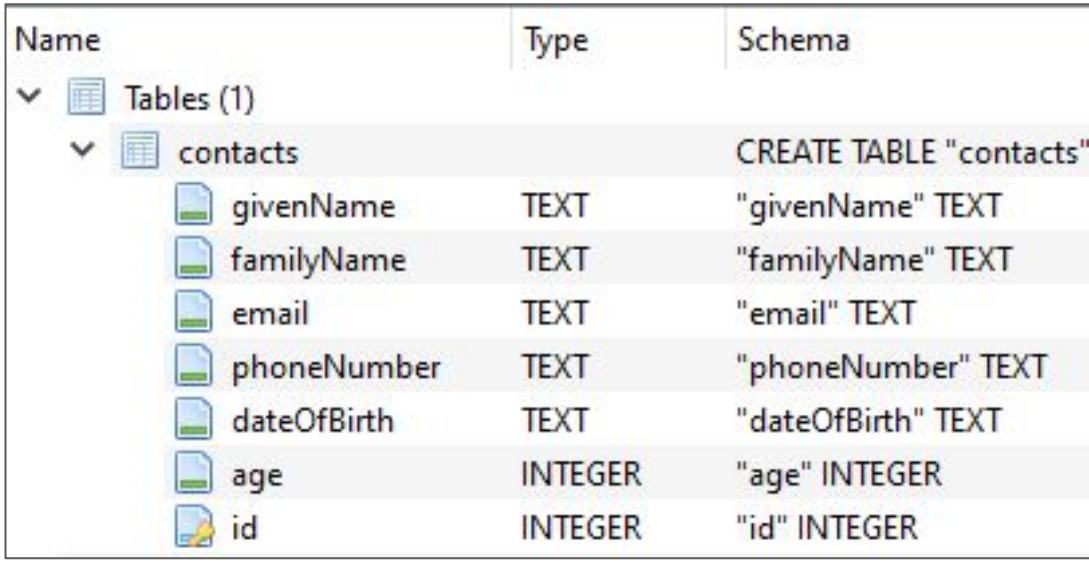
The rule:

```
CREATE table (
    field datatype,
    field datatype,
    ...
    PRIMARY KEY (field, ...)
);
```

SQL: Creating a table

Examples

```
CREATE TABLE 'contacts' (
    'givenName'    TEXT,
    "familyName'   TEXT,
    'email'        TEXT,
    'phoneNumber'  TEXT,
    'dateOfBirth'  TEXT,
    'age'          INTEGER,
    'id'           INTEGER,
    PRIMARY KEY('id')
);
```



Note: Strings are enclosed with single quotes.

SQL: Inserting records

The rule:

```
INSERT INTO table (field, field, ...) VALUES (value, value, ...);
```

Examples:

```
INSERT INTO contacts (
   givenName, familyName, email,
   phoneNumber, dateOfBirth, age, id
) VALUES (
   'Sheldon', 'Cooper', 'sheldon@gmail.com',
   '555 0001', '26/02/1980', 41, 1
);
```

Note: Strings are enclosed with single quotes.

SQL: Querying our database

Get every field of every record from a table.

```
SELECT * FROM table;
```

Get some fields for every record from a table.

```
SELECT field, field, field FROM table;
```

Get records that match where field is set to value.

```
SELECT ... FROM table WHERE field = value;
```

Get records that match where two fields have set values.

```
SELECT ... FROM table WHERE field = value AND field = value;
```

Get records that match where two possible field/values exist.

```
SELECT ... FROM table WHERE field = value OR field = value;
```

SQL: Querying our database

Examples.

```
SELECT * FROM contacts;

SELECT * FROM contacts WHERE id=5;

SELECT * FROM contacts WHERE age >= 40;

SELECT * FROM contacts WHERE familyName >= 'M';

SELECT * FROM contacts WHERE dateOfBirth LIKE '%1981';
```

SQL: Updating records

The rule:

```
UPDATE table SET field=value, field=value WHERE field=value;
```

Example:

```
UPDATE contacts SET age=41 WHERE id=1;
```

Be careful of your WHERE clause. Leave it out and you will change EVERY RECORD. Test it with a SELECT before using it to UPDATE.

SQL: Deleting records

The rule:

```
DELETE FROM table WHERE field=value;
```

Example:

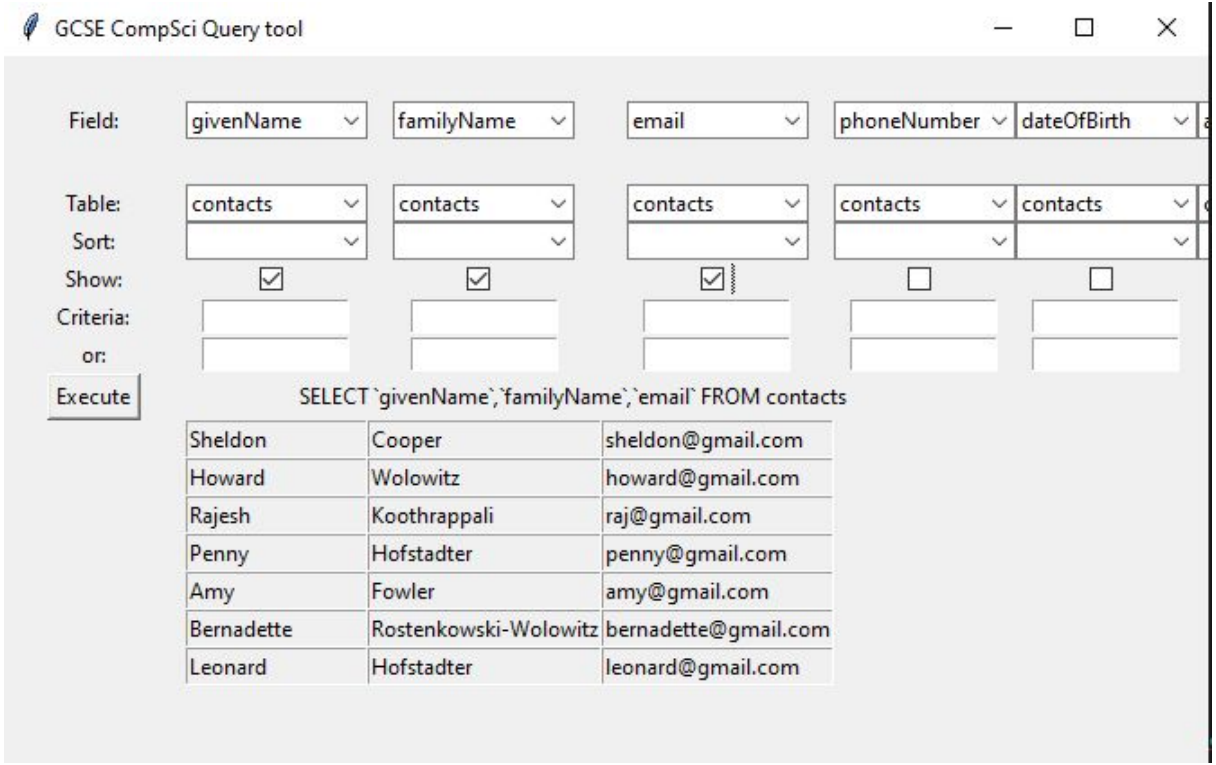
```
DELETE FROM contacts
WHERE familyName='Cooper' AND givenName='Sheldon';
```

Be careful of your WHERE clause. Leave it out and you will change EVERY RECORD. Test it with a SELECT before using it to UPDATE.

igcse "query-by-example"

Field:				
Table:				
Sort:				
Show:	☐	☐	☐	☐
Criteria:				
or:				

igcse "query-by-example"



https://github.com/paulbaumgarten/gcse-query-tool

igcse "query-by-example"

Challenge

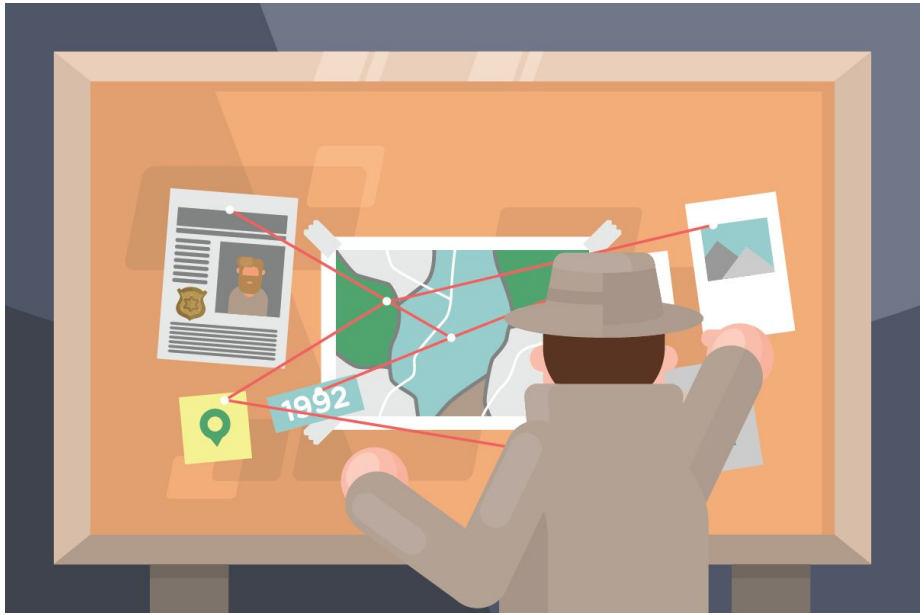
Solve the murder!

There's been a Murder in SQL City!

A crime has taken place and the detective needs your help. The detective gave you the crime scene report, but you somehow lost it. You vaguely remember that the crime was a murder that occurred sometime on Jan.15, 2018 and that it took place in SQL City.

Search the database, solve the murder!

https://mystery.knightlab.com/



Tasker

tasker.db

accounts	folders	tasks
userid (text, PK) password (text) salt (text) email (text) name (text)	userid (text, PK) id (text, PK) name	userid (text) folderid (text) id (text, PK) title (text) due (text) reminder (text) created (text) category (text) priority (integer) status (text) notes (text) link (text)

tasker.db: Add test data

server.py: Database functionality

server.db: Read database

server.db: Render HTML

server.py: Receive HTML forms

server.py: Save to database

Review

Past paper questions review