Syllabus Content

9- Databases

Candidates should be able to:

- Define a single-table database from given data storage requirements

  Notes and guidance

  - fields
  - records
  - validation

- Suggest suitable basic data types

  Notes and guidance
  Including:

  - text/alphanumeric
  - character
  - Boolean
  - Integer
  - Real
  - Date/time

- Understand the purpose of a primary key and identify a suitable primary key for a given database table

- Read, understand and complete structured query language (SQL) scripts to query data stored in a single database table

  Notes and guidance
  Limited to:

  - ❖ SELECT
  - ❖ FROM
  - ❖ WHERE
  - ❖ ORDER BY
  - ❖ SUM
  - ❖ COUNT
  - ❖ Identifying the output given by an SQL statement that will query the given contents of a database table

What is a Database?

A **database**, also called **electronic database** is organized collection of data or information typically stored electronically in a computer system for storage, rapid search and retrieval of data.

A **database management system (DBMS)** extracts information from the database in response to queries.

Database Entity:

Database entity is a thing, person, place, unit, object or any item about which the data should be captured and stored in the form of properties, and tables.

Table:
In Relational database model, a **table** is a collection of data elements organized in terms of rows and columns.

Record:
Record is the storage representation of a **row** of data.

Field:
A database field is a set of data values, of the same data type, in a table. It is also referred to as a **column** or an **attribute**.

Table

	Field 1	Field 2	Field 3	Field 4
Record 1	Field 1	Field 2	Field 3	Field 4
Record 2	Field 1	Field 2	Field 3	Field 4
Record 3	Field 1	Field 2	Field 3	Field 4
Record 4	Field 1	Field 2	Field 3	Field 4
Record 5	Field 1	Field 2	Field 3	Field 4
Record 6	Field 1	Field 2	Field 3	Field 4

Row → Record 2

Column

▲ **Figure 9.1** Structure of a database table

Flat- File database (Single table database):

A flat file database is a type of database that stores data in a single table. This is unlike a relational database, which makes use of multiple tables and relations.

🔖 Flat-File databases hold all of their data in **one table only**.

🔖 They are only suitable for **very simple databases**.

The patient database is an example of a flat-file as all of the information is stored in one single table:

Flat-File (one table)

Patient Id	Name	D.o.B	Gender	Phone	Doctor Id	Doctor	Room
134	Jeff	4-Jul-1993	Male	7876453	01	Dr Hyde	03
178	David	8-Feb-1987	Male	8635467	02	Dr Jekyll	06
198	Lisa	18-Dec-1979	Female	7498735	01	Dr Hyde	03
210	Frank	29-Apr-1983	Male	7943521	01	Dr Hyde	03
258	Rachel	8-Feb-1987	Female	8367242	02	Dr Jekyll	06

Limitations of a Flat-File Database

The problems with using a flat-file databases are as follows:
- **Duplicated Data** is often **unnecessarily** entered.
- **Database space** is wasted with this duplicated data.
- **Duplicated Data** takes a long time to enter and update (unnecessarily).

What is Data Redundancy?

Data Redundancy is where you store the **same data many times** (duplicate data) in your table.
This repeated data needs to be **typed in over and over again** which takes a long time.
For example:-
The patients database contains several entries of duplicate data:
- **Doctor Id**
- **Dr. Hyde**
- **Room 03**

Patient Id	Name	D.o.B	Gender	Phone	Doctor Id	Doctor	Room	
134	Jeff	4-Jul-1993	Male	7876453	01	Dr Hyde	03	Duplicate
178	David	8-Feb-1987	Male	8635467	02	Dr Jekyll	06	
198	Lisa	18-Dec-1979	Female	7498735	01	Dr Hyde	03	Duplicate
210	Frank	29-Apr-1983	Male	7943521	01	Dr Hyde	03	
258	Rachel	8-Feb-1987	Female	8367242	02	Dr Jekyll	06	

REMEMBER!

Data that is duplicated unnecessarily within a database is bad practice. If we had 100 patients who were all assigned Dr Hyde. His **Doctor Id, Name** and **Room Number** would have to be **entered 100 separate times**.

Also if Dr Hyde left the doctors surgery, we would have to update the new doctors details for every patient in the database.

What is the solution to Data Redundancy?

To avoid the data redundancy with flat-file databases is to create
a **relational database**.

Primary Key:

The **primary key** is a field which uniquely identifies each record in the table. It cannot be
duplicated or contain a null value e.g. **Candidate number** for a school student in Table. Every
student must have a **Candidate number.** It cannot a blank value or duplicated.

Relational Databases

- Relational Databases use **two or more tables** linked together
  (to form a relationship).
- Relational Databases do not store all the data in the same table.
- Repeated data is moved into it's **own table** as shown in the image below:

Key Words:
Relationships,
Primary Key,
Foreign Key,
Common Field

NOTE!
Here I have moved all
the **repeating data** into
a table of its own.

Now I have a **patient
table** (for patient
details) and a **doctor
table** (for doctor details)

Patient Table

Patient Id	Name	D.o.B	Gender	Phone	Doctor Id
134	Jeff	4-Jul-1993	Male	7876453	01
178	David	8-Feb-1987	Male	8635467	02
198	Lisa	18-Dec-1979	Female	7498735	01
210	Frank	29-Apr-1983	Male	7943521	01
258	Rachel	8-Feb-1987	Female	8367242	02

Doctor Table

Doctor Id	Doctor	Room
01	Dr Hyde	03
02	Dr Jekyll	06

What is a relationship?

- A relationship is formed when our **two tables are joined together**.
- Relationships make use of **key fields** and **primary keys** to allow the **two tables
  to communicate** with each other and **share their data**.
- Key fields are identified using a primary key as shown in the image below:

Key Field

Patient Table

Key Field

Doctor Table

Patient Id	Name	D.o.B	Gender	Phone	Doctor Id
134	Jeff	4-Jul-1993	Male	7876453	01
178	David	8-Feb-1987	Male	8635467	02
198	Lisa	18-Dec-1979	Female	7498735	01
210	Frank	29-Apr-1983	Male	7943521	01
258	Rachel	8-Feb-1987	Female	8367242	02

Doctor Id	Doctor	Room
01	Dr Hyde	03
02	Dr Jekyll	06

- Once the tables are linked together each one can read data from the other.
- This means that we only need to enter the details of each doctor once instead of
  many separate entries.

How do you form the relationship? (Link the tables)

- In order to link the tables we need to use a **common field**.

- A common field is **data that appears in BOTH tables**.
- If you look at the image below you will see that the common field in the patient database is **Doctor Id**:

Patient Table

Patient Id	Name	D.o.B	Gender	Phone	Doctor Id
134	Jeff	4-Jul-1993	Male	7876453	01
178	David	8-Feb-1987	Male	8635467	02
198	Lisa	18-Dec-1979	Female	7498735	01
210	Frank	29-Apr-1983	Male	7943521	01
258	Rachel	8-Feb-1987	Female	8367242	02

Doctor Table

Doctor Id	Doctor	Room
01	Dr Hyde	03
02	Dr Jekyll	06

What is a foreign key?

- A foreign key is a **regular field in one table** which is being used as the **key field (primary key) in another table**.
- Foreign keys are used to **provide the link** (relationship) between the tables.

For example:-

In our patient database, **Doctor Id** is a **key field in the Doctor Table** but is also being used in the **Patient Table as a foreign key**:

Patient Table

Patient Id	Name	D.o.B	Gender	Phone	Doctor Id (Foreign Key)
134	Jeff	4-Jul-1993	Male	7876453	01
178	David	8-Feb-1987	Male	8635467	02
198	Lisa	18-Dec-1979	Female	7498735	01
210	Frank	29-Apr-1983	Male	7943521	01
258	Rachel	8-Feb-1987	Female	8367242	02

Doctor Table

Doctor Id (Primary Key)	Doctor	Room
01	Dr Hyde	03
02	Dr Jekyll	06

- The foreign key (Doctor Id in the patient table) can then be used to match to the primary key (Doctor Id in the doctor table) and share the correct data.

For example:-
A patient with a Doctor Id **01** will be automatically assigned to **Doctor Hyde** & **Room 03**.

REMEMBER!
Now that we have linked our two tables we can update our doctor information very quickly.

For example:-
If Dr Hyde changed his **room number to 02**, we only need to **change this information once** in the doctor table. The new room would **automatically be assigned to every patient** who was under the care of Dr Hyde.

What are Data Types used in Databases?

Each field will require a **data type** to be selected.

A data type classifies how the data is stored, displayed and the operations that can be performed on the stored value.

These database data types are specified in the syllabus.

They are available to use as **Access data types**, but the names Access uses may be different.

Data Types in syllabus:

text/alphanumeric (stores text)

character (stores a single character)

Boolean (Yes/ No, or True/False)

Integer (stores Whole number)

Real (stores decimal number)

Date/time (stores Date and Time)

Validation:

The role of validation was discussed in earlier chapters. Some validation checks will be automatically provided by the database management software that is used to construct and maintain the database.

Other validation checks need to be set up by the database developer during the construction of the database.

There are many different types of validation checks including:

range checks

length checks

type checks

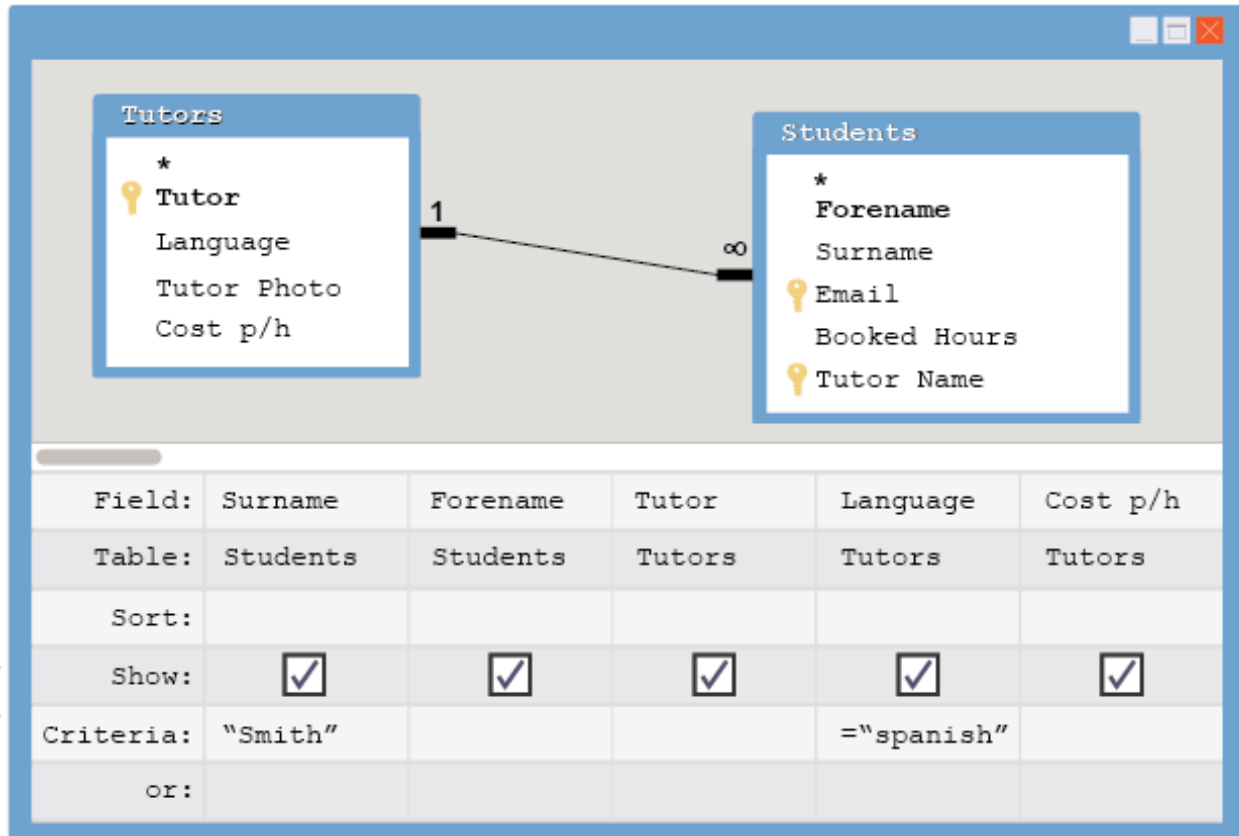presence checks

format checks

Database Queries:

Queries allow users to search and sort data held in a database. There are two different ways to create queries.

The first is to use the built in query generator that comes with most database software.

The example below shows a complex query searching two tables to find any student named Smith who is taught Spanish.

The query below has been created using an in built query generator.

Tutors				
*			**Students**	
🔑 Tutor	1		*	
Language		∞	Forename	
Tutor Photo			Surname	
Cost p/h			🔑 Email	
			Booked Hours	
			🔑 Tutor Name	

Field:	Surname	Forename	Tutor	Language	Cost p/h
Table:	Students	Students	Tutors	Tutors	Tutors
Sort:					
Show:	☑	☑	☑	☑	☑
Criteria:	"Smith"			="spanish"	
or:					

More advanced users may want to create some code to **perform queries.**

SQL is an example of a language used to create queries within database applications.

The same query in SQL could be written as:

```
SELECT surname, forename, tutor, language, cost p/h
FROM Students, Tutors
WHERE Students.Surname ='Smith' AND Tutors.Language = 'spanish';
```

Past paper (sample exam style questions):

A database table, FLIGHT, is used to keep a record of flights from a small airfield. Planes can carry passengers, freight or both. Some flights are marked as private and only carry passengers.

Flight number	Plane	Notes	Departure time	Passengers
FN101	Caravan 1	Private passenger flight	08:00	Y
CN101	Caravan 2	Freight only	08:30	N
CN102	Piper 1	Freight only	09:00	N
FN104	Piper 2	Passengers only	09:20	Y
FN105	Piper 1	Freight and passengers	10:00	Y
FN106	Caravan 1	Passengers only	10:30	Y
CN108	Caravan 2	Freight only	08:00	N
CN110	Lear	Private passenger flight	08:00	Y

(b) A query-by-example has been written to display just the flight numbers of all planes leaving after 10:00 that only carry passengers.

Field:	Flight number	Passengers	Departure time	
Table:	FLIGHT	FLIGHT	FLIGHT	
Sort:				
Show:	☐	☑	☐	☐
Criteria:		= Y	= 10:00	
or:				

Explain why the query-by-example is incorrect, and write a correct query-by-example. Explanation

....

Re-Write correct query:

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

[7]

Answers:

Field:	Flight number	Passengers	Departure time	Notes
Table:	FLIGHT	FLIGHT	FLIGHT	FLIGHT
Sort:				
Show:	☑	☐	☐	☐
Criteria:		=Y	>10:00	<> "Freight and passengers"
or:				

OR

Field:	Flight number	Departure time	Notes
Table:	FLIGHT	FLIGHT	FLIGHT
Sort:			
Show:	☑	☐	☐
Criteria:		>10:00	= "Passengers only"
or:		>10:00	="Private passenger flight"

Queries in SQL:

Databases allow us to store and filter data to find specific information. A database can be queried using a variety of methods, although this depends on the software you are using.

Databases can use query languages or graphical methods to interrogate the data.

Query Language:

Query language is a **written language used only to write specific queries**. This is a powerful tool as the user can define precisely what is required in a database. SQL is a popular query language used with many databases.
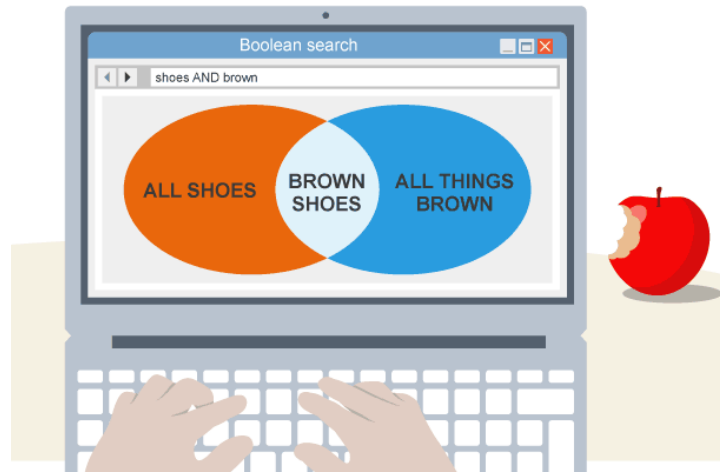
Boolean Operators:

Boolean **operators** are used to filter databases using **AND**, **OR** or **NOT**. They can search multiple fields at the same time to help us retrieve the data that we need.

AND

AND is used to search records that contain one piece of information AND another.

A query for the words *brown AND shoes* would return results that contain the words brown and shoes.

In general, search engines treat the query *brown shoes* as *brown AND shoes*, which means that all results will contain both words, eg *brown trousers and red shoes for sale*.

OR

OR is used to search for records that contain EITHER one piece of information OR another.

for example *black shoes OR white shoes*. This would present results for **any shoes that were black or white**.

Most search engines use the OR function best if the search statements are defined by speech marks, eg *"brown shoes" OR "black jeans"* would show pages which either contain brown shoes or black jeans.

NOT

NOT is used to exclude results.

The query *shoes NOT brown* will return results that **contain the word shoes but NOT the word brown**.

Arithmetic operators

A query can also be performed using **arithmetic operators**. These help to make specific searches related to numerical data.

Functions of arithmetic operators

This table shows some arithmetic operators and their functions:

Operator	Meaning
=	Equals
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
<>	Not equal to

Using arithmetic operators

The table below shows some BBC TV programme listings:

ID	Title	Genre	Duration (mins)	Channel
01	EastEnders	Drama	30	BBC1
02	Dragons' Den	Entertainment	60	BBC2
03	The Voice	Entertainment	75	BBC3
04	Blue Peter	Children's	25	CBBC
05	Wild Brazil	Nature	60	BBC4
06	Match of the Day	Sport	80	BBC1
07	Tom and Dom	Comedy	10	CBBC

Queries are useful for searching for specific conditions. You might want to find entertainment programmes on BBC3. A query for these conditions would look like this:

```
SELECT * FROM Programmes
WHERE Genre = "Entertainment"
AND Channel = "BBC3";
```

This would return the programme **'The Voice'**.

03	The Voice	Entertainment	75	BBC3

You may want a programme that is less than 20 minutes long or is a nature programme. A query for these conditions would look like this:

```
SELECT * FROM Programme
WHERE Duration < 20
OR Genre ='Nature';
```

This would return the programmes "Tom and Dom" and "Wild Brazil".

05	Wild Brazil	Nature	60	BBC4
07	Tom and Dom	Comedy	10	CBBC

Working with SQL: Step 2

You can query this table using SQL code. You may decide that you wish to sort the programmes by duration.

The SQL code needed would look like the example below:

```
SELECT Duration, Title
FROM Programmes
ORDER BY Duration;
```

- the **SELECT** statement states which fields to look at - the Title and Duration fields
- the **FROM** statement states which table to look at - Programmes
- the **ORDER BY** statement sorts the Duration field in ascending order by default

This table shows the results from this query:

Duration	Title

Duration	Title
25	Blue Peter
30	EastEnders
50	Newsnight
60	Wild Brazil
75	The Voice
90	Sherlock

Working with SQL: Step 3

The SQL **WHERE** statement is used to isolate one record or several records with similar attributes.

ID	Title	Genre	Duration
01	EastEnders	Drama	30
02	Newsnight	Current affairs	50
03	The Voice	Entertainment	75
04	Blue Peter	Children's	25
05	Wild Brazil	Nature	60
06	Sherlock	Drama	90

The following code searches the Title field of the table to find the words 'The Voice'.

```
SELECT ID, Title, Genre, Duration
FROM Programmes
WHERE Title = "The Voice";
```

- the **WHERE** statement specifies which text to look for

This table shows the results from this query:

ID	Title	Genre	Duration
03	The Voice	Entertainment	75

Alternatively, you could find all of the programmes which are less than 30 minutes long using this code:

```
SELECT ID, Title, Genre, Duration
FROM Programmes
WHERE Duration <30;                    or


SELECT * FROM Programmes
WHERE Duration < 30;
```

This table shows the results from this query:

ID	Title	Genre	Duration
04	Blue Peter	Children's	25

Wildcards

The wildcard uses the * symbol, and is used in place of any number of unknown characters. For example, the following code searches for all programmes with the letter **i** in the title:

```
SELECT * FROM Programmes
WHERE Title LIKE "*i*" ;
```

This table below shows the results from this query:

ID	Title	Genre	Duration
02	Newsnight	Current Affairs	50
03	The Voice	Entertainment	75
05	Wild Brazil	Nature	60

COUNT() Syntax

The COUNT() function returns the number of rows that matches a specified criterion.

SELECT COUNT(*column_name*)
FROM *table_name*
WHERE *condition*;

DEMO DATABASE
Table Products

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

SELECT COUNT ProductID
FROM Products;

RESULT COUNT	
COUNT(Price)	5

The COUNT() function returns the number of rows that matches a specified criteria.

SUM() Syntax

The SUM() function returns the total sum of a numeric column.

SELECT SUM(*column_name*)

FROM *table_name*

WHERE *condition*;

Demo Database
Below is a selection from the "**Order**" **table** in the sample database:

OrderDetailID	OrderID	ProductID	Quantity
1	10248	11	12
2	10248	42	10
3	10248	72	5
4	10249	14	9
5	10249	51	40

```
SELECT SUM Quantity
FROM Order ;
```

RESULT	
SUM (Quantity)	76

References:

- https://www.bbc.com/education/guides/z37tb9q/revision/8
- https://www.w3schools.com/sql/sql_count_avg_sum.asp
- Cambridge (Hodder) O level Book by David Watson & Helen Williams