# Section A: Database concepts

1. What is an **entity** in the context of databases?

Something that has information stored about it in some context, for example products or people

2. The relationship between two entities may be one of three types, or degrees.

What are the three degrees of relationship between entities?

one to many

one to one

many to many

3. Draw entity relationship diagrams for each of the following pairs of related entities:

(a) Dentist and patient

https://app.diagrams.net/#HJachymT%2Fa-level-cs-blog%2Fmain%2FComputer%20Systems%2F1.3%2F1.3.2%2FER%20diagrams.drawio

or

https://github.com/JachymT/a-level-cs-blog/blob/main/Computer%20Systems/1.3/1.3.2/ER%20diagrams.drawio

(b) Student and teacher

(c) UK citizen and UK Passport

(d) Product and component

4. A company makes a range of kitchen utensils which they sell online. They record details of their customers, products and orders received in a database. An order may be for several products.

Complete the E-R diagram to show all the relationships between all the entities.

Customer
Product
Order
OrderLine

5. A cinema club shows up to three screenings of films on a given day. They need a database to keep track of which films have been shown on which dates. It has been suggested that they use one table (relation) about the films they show, which will hold the following data:

FilmID, Title, Duration, Male Lead, Female Lead, Date shown, Time shown, Tickets sold.

(a) Give a reason why this is not a satisfactory solution.

A flat file database is inefficient, actor information will be duplicated every time a new record is added. DATA REDUNDANCY

There may be more than one male/ female lead - creates a many to many relationship which causes issues like not linking to any film in particular.

(b) Show how the data could be reorganised into **four** relations, using the notation

Entity name (attribute1, attribute2, attribute3 …)

An underscore indicates the primary key. The entities that have been identified are Film, Actor, ActorInFilm and Showing.

Film (FilmID, Title, Duration)
Showing(Date, time, tickets sold, *film ID*) - a showID could be created here as a primary key = Showing(ShowID, Date, time, tickets sold, *film ID*)
Actor (ActorID, name, gender, role) - works better than female and male lead data
ActorInFilm (FilmID, ActorID) -- link table solves the many to many relationship

(c) Identify **one** foreign key and **one** composite key in any of the relations.

In the *Showing* entity the date shown and time shown are the composite key and the foreign key is the filmID

(d) Explain how a primary key is established from a set of candidate keys and how a secondary key can be identified in relations to this.

The primary key must be unique in that table. Secondary keys (candidate keys that are not selected as the primary key) are all other keys that are not defined in other tables. For example in the Film entity the title is the secondary key.

## Section B: SQL

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 MS 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 titles beginning with “S” (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”) | |

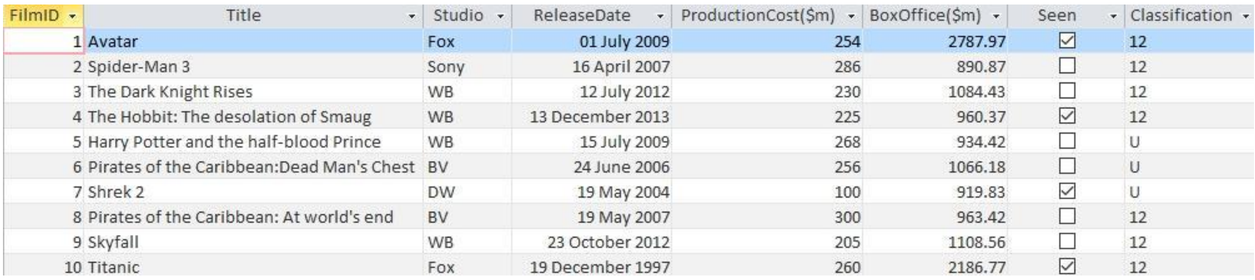
note :
* symbol and % symbol are used to note any amount of characters when querying

eg. ‘*a’ ‘%a’

_ symbol and ? symbol are used to note 1 character (any) when querying eg. ‘b_’ ‘b?

## Task 1

The questions in this task all relate to **tblFilm**, shown below.



| FilmID | Title | Studio | ReleaseDate | ProductionCost($m) | BoxOffice($m) | Seen | Classification |
|---|---|---|---|---|---|---|---|
| 1 | Avatar | Fox | 01 July 2009 | 254 | 2787.97 | ☑ | 12 |
| 2 | Spider-Man 3 | Sony | 16 April 2007 | 286 | 890.87 | ☐ | 12 |
| 3 | The Dark Knight Rises | WB | 12 July 2012 | 230 | 1084.43 | ☐ | 12 |
| 4 | The Hobbit: The desolation of Smaug | WB | 13 December 2013 | 225 | 960.37 | ☑ | 12 |
| 5 | Harry Potter and the half-blood Prince | WB | 15 July 2009 | 268 | 934.42 | ☐ | U |
| 6 | Pirates of the Caribbean:Dead Man's Chest | BV | 24 June 2006 | 256 | 1066.18 | ☐ | U |
| 7 | Shrek 2 | DW | 19 May 2004 | 100 | 919.83 | ☑ | U |
| 8 | Pirates of the Caribbean: At world's end | BV | 19 May 2007 | 300 | 963.42 | ☐ | 12 |
| 9 | Skyfall | WB | 23 October 2012 | 205 | 1108.56 | ☐ | 12 |
| 10 | Titanic | Fox | 19 December 1997 | 260 | 2186.77 | ☑ | 12 |

Write SQL statements to:

(a) select the Film ID, Title and Classification of all films with classification U or 12, which have been marked as “Seen”.

The results should be ordered in Ascending order of Title.

```
SELECT FilmID, Title, Classification
FROM tblFilm
WHERE Classification in (‘12’,’U’)
AND Seen = True
ORDER BY Title
```

*note : ASC (ascending) order is the default order*

Which Film IDs will be selected, in what order?

1,7,4,10

(b) Select the Title and Studio of all films released in 2012 or 2013 which took more than £220m at the box office.

```
SELECT Title, Studio FROM tblFilm
WHERE releaseDate (between #01/01/2012 AND #31/12/2013)
AND (BoxOffice >220)
```

*note: remember to format dates correctly with #xx/xx/xxxx*

(c) Select all columns for films from Fox, Sony or WB and display in descending order of release date.

SELECT * FROM tblFilm

WHERE Studio IN (“Fox”, “Sony”, “WB”)

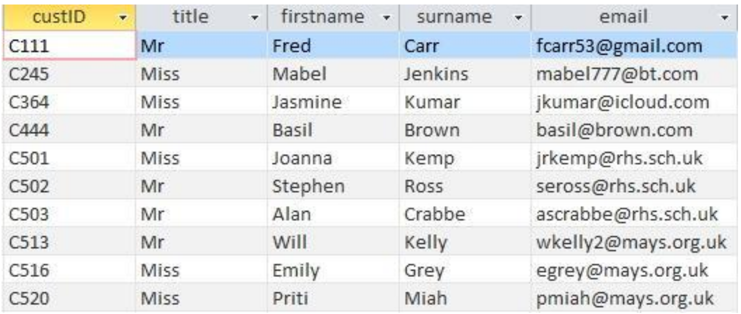
ORDER BY ReleaseDate DESC

*note: remember functions in capital letters*

## Task 2

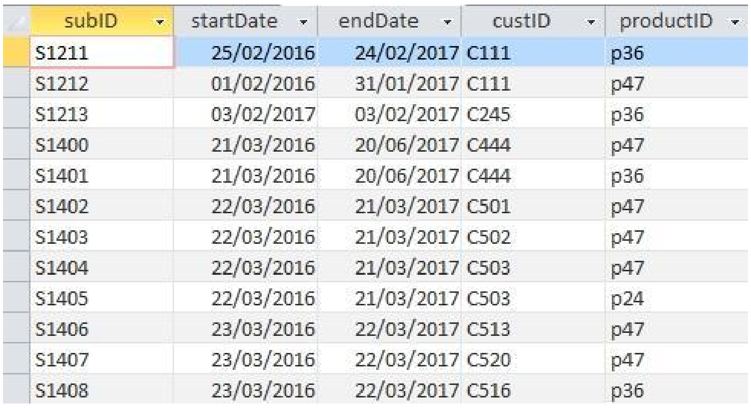
The database **RevisionSubs.accdb** has three tables:

**tblCustomer**



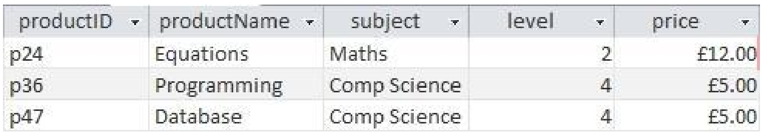
| custID | title | firstname | surname | email |
|---|---|---|---|---|
| C111 | Mr | Fred | Carr | fcarr53@gmail.com |
| C245 | Miss | Mabel | Jenkins | mabel777@bt.com |
| C364 | Miss | Jasmine | Kumar | jkumar@icloud.com |
| C444 | Mr | Basil | Brown | basil@brown.com |
| C501 | Miss | Joanna | Kemp | jrkemp@rhs.sch.uk |
| C502 | Mr | Stephen | Ross | seross@rhs.sch.uk |
| C503 | Mr | Alan | Crabbe | ascrabbe@rhs.sch.uk |
| C513 | Mr | Will | Kelly | wkelly2@mays.org.uk |
| C516 | Miss | Emily | Grey | egrey@mays.org.uk |
| C520 | Miss | Priti | Miah | pmiah@mays.org.uk |

**tblSubscription**



| subID | startDate | endDate | custID | productID |
|---|---|---|---|---|
| S1211 | 25/02/2016 | 24/02/2017 | C111 | p36 |
| S1212 | 01/02/2016 | 31/01/2017 | C111 | p47 |
| S1213 | 03/02/2017 | 03/02/2017 | C245 | p36 |
| S1400 | 21/03/2016 | 20/06/2017 | C444 | p47 |
| S1401 | 21/03/2016 | 20/06/2017 | C444 | p36 |
| S1402 | 22/03/2016 | 21/03/2017 | C501 | p47 |
| S1403 | 22/03/2016 | 21/03/2017 | C502 | p47 |
| S1404 | 22/03/2016 | 21/03/2017 | C503 | p47 |
| S1405 | 22/03/2016 | 21/03/2017 | C503 | p24 |
| S1406 | 23/03/2016 | 22/03/2017 | C513 | p47 |
| S1407 | 23/03/2016 | 22/03/2017 | C520 | p47 |
| S1408 | 23/03/2016 | 22/03/2017 | C516 | p36 |

**tblProduct**



| productID | productName | subject | level | price |
|---|---|---|---|---|
| p24 | Equations | Maths | 2 | £12.00 |
| p36 | Programming | Comp Science | 4 | £5.00 |
| p47 | Database | Comp Science | 4 | £5.00 |

(a) List the IDs and surnames of all the customers who will be displayed by the following query:

SELECT tblCustomer.custID, firstname, surname, ProductName, tblProduct.productID

FROM tblCustomer, tblProduct, tblSubscription

```
WHERE  tblCustomer.custID = tblSubscription.custID
    AND tblProduct.productID = tblSubscription.productID
    AND (productID = “p36” OR productID = “p24”)
```

*note: here they haven't joined the tables, but you don't technically have to, it just reduces steps in complex queries to join them together.*

C111, Carr

C245, Jenkins

C444, Brown

C503, Crabbe

C516, Grey

(b) Write an SQL statement to display IDs and surnames all the customers at Mays School (identified by their email address) who have subscriptions for product p47.

```
SELECT tblCustomer.custID, tbl.Customer.surname
FROM tblCustomer JOIN tblProduct ON (tblCustomer.custID = tblProduct.custID)
WHERE email LIKE ‘%@mays.org.uk’
AND productID = ‘p47’
```

**another solution using all three table and with referential integrity from checking the primary and foreign keys instead of a JOIN doing it for us (derived from question 2a)**

```
SELECT tblCustomer.custID, tbl.Customer.surname
        FROM tblCustomer, tblProduct, tblSubscription
        WHERE  tblCustomer.custID = tblSubscription.custID
            AND tblProduct.productID = tblSubscription.productID
            AND tblCustomer.email LIKE ‘%@mays.org.uk’
            AND productID = ‘p47’
```

## Task 3

The table below shows common data types:

| 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 CURRENCY |

(a) Write an SQL statement to create a table for a table called Member, which has the following fields:

| | |
|---|---|
| MemberID | 4 characters (Primary key, compulsory field) |
| Firstname | max 12 characters (compulsory field) |
| Surname | max 20 characters (compulsory field) |
| DateJoined | Date dd/mm/yy (compulsory field) |
| SubPaid | Yes/No (optional field) |

```
CREATE TABLE Member
(
MemberID CHAR(4) NOT NULL PRIMARY KEY,
Firstname VARCHAR(12) NOT NULL,
Surname VARCHAR(20) NOT NULL.
DateJoined DATE NOT NULL,
SubPaid BOOLEAN
)
```

(b) Write an SQL statement to amend the table to add a new column for Category, a Boolean data type

```
ALTER TABLE Member
    ADD COLUMN Category BOOLEAN;
```

(c) Write an SQL statement to delete the column SubPaid

```
ALTER TABLE Member
    DROP COLUMN SubPaid;
```

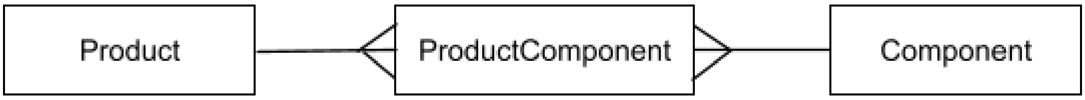
(d) Write an SQL statement to change the maximum length of the Firstname field to 15 characters

```
ALTER TABLE Member
    MODIFY COLUMN Firstname VARCHAR(15) NOT NULL;
```

***note COLUMN keyword is optional**

## Task 4

Three linked tables are defined as follows:



Product (ProductID, Description, Price)

ProductComponent (*ProductID*, *CompID*, Quantity)

Component (CompID, CompDesc, Cost)

When there are three linked tables, the linking table is defined as follows:

```
CREATE TABLE ProductComponent
(
ProductID       CHAR(4) NOT NULL,
CompID          CHAR(6) NOT NULL,
Quantity        INTEGER,
FOREIGN KEY     ProductID REFERENCES Product(ProductID),
FOREIGN KEY     CompID REFERENCES Component(CompID),
PRIMARY KEY (ProductID, CompID)
)
```

Write the SQL statements to create the table **Component**. CompDesc is to be a maximum of 25 characters, and Cost is a currency field. All fields are compulsory.

```
CREATE TABLE Component
(
CompID CHAR(6) NOT NULL PRIMARY KEY,
CompDesc VARCHAR(25) NOT NULL,
Cost CURRENCY NOT NULL,
)
```

## Task 5

(a) Write an SQL statement to insert a new record into the Member table described in Task 3(a). The new record is to have the following data values:

| | |
|---|---|
| MemberID | M046 |
| Firstname | William |
| Surname | Oldfield |
| DateJoined | 23/06/2016 |
| SubPaid | No |

```
INSERT INTO Member
VALUE (“M046”, “William”, “Oldfield”, #23/06/2016#, False)
```

(b) Write an SQL statement to update this record, the first name is to be changed to “Bill” and the subscription has now been paid.

```
UPDATE Member
SET Firstname = “Bill”, SubPaid = True
WHERE MemberID = “M046”
```

(c) Write an SQL statement to delete the record for member M025.

```
DELETE FROM Member

WHERE MemberID = “M025”
```