Structured Query Language (SQL)

Welcome to a Computer Science Taster session.

Ideally these lessons would have been carried out in a classroom environment with much more interaction. Owing to the continuing Covid constraints the following worksheet and practical on-line activities are presented as a taster for Computer Science lessons.

Work through the theory explained in the first part of the document and ten try and apply what you have learnt in the practical on-line activity on pages 11, 12 and 13.

In normal lessons we would be using Microsoft Teams to host the workbook and would be inserting screenshots and submitting assignments for our work. The lesson would revolve around a presentation from the teacher to introduce, explain and discuss the concepts.

I look forward to welcoming you to Computer Science. Feel free to direct any questions for additional information and help via email or Teams chat to:

p.macaree@oaklandscatholicschool.org

Introduction

A database is a table, or a collection of tables that store data. The software that operates on this data is called a database management system (DBMS) and all database management systems will need to be able to carry out the following actions:

- Create a database
- Add data to the database
- Delete data from the database
- Edit the data
- Search the database for specific for data

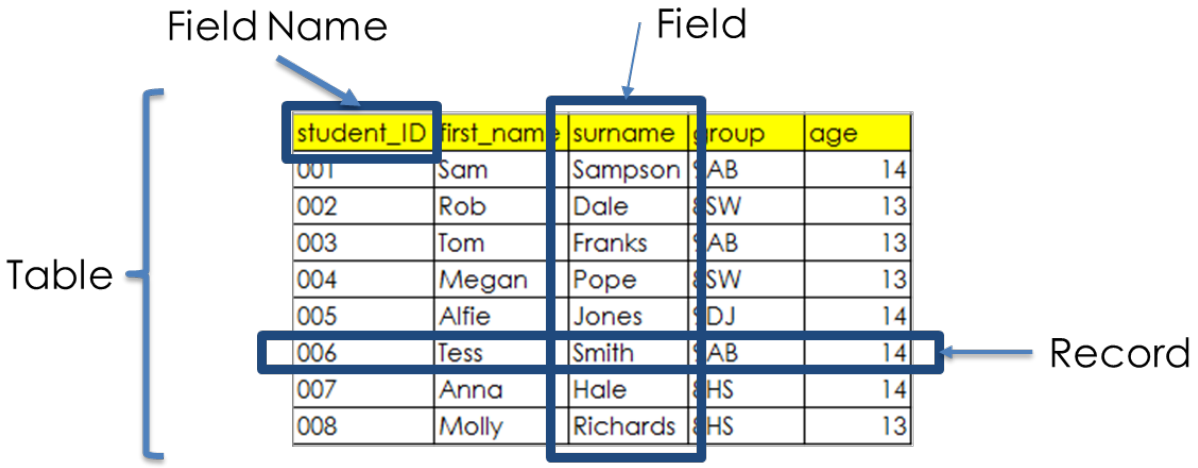
Database Terminology

TABLE: A table is simply a collection of data that relates to a person or object (often referred to as an entity (e.g. students))

RECORD: A collection of data about a single entity (e.g. a student)

FIELD: A unique piece of data about an entity (student surnames)

FIELD NAME: An identifier for the single piece of data (e.g. 'surnames').



Queries

A very common job carried out on a database is a query. A query is where we search for data in our database. We ask the database questions and it responds with answers. For example, we might ask our database for all students who are 14 and the database will respond with a list of all students that match the 'age=14' criteria.

When we write programs that have databases, SQL (structured query language) is the language we use to create, update, delete and query our program's database.

SQL

The SQL language is made up of a selection of statements to carry out jobs on databases.

There are quite a few but the basic / common statements are:

CREATE TABLE – create a database table

SELECT – selects data

UPDATE – edits data

DELETE – removes data

INSERT INTO – inserts new data

CREATE-ing a Table

If we wanted to create the following empty table…

student_ID	first_name	surname	group	age

...we would use the following syntax:

```
CREATE TABLE Students
(student_ID text,
first_name text,
surname text,
group text,
age number)
```

The first line of SQL creates a table called 'Students'. The following lines, separated by commas specify the name of each column and the type of data it is to hold.

SELECT-ing Data

The SELECT statement is used to select particular data that we want to work on. We could select the entire table or specific data based on certain criteria.

Using this example table…

student_ID	first_name	surname	group	age
001	Sam	Sampson	9AB	14
002	Rob	Dale	8SW	13
003	Tom	Franks	9AB	13
004	Megan	Pope	8SW	13
005	Alfie	Jones	9DJ	14
006	Tess	Smith	9AB	14
007	Anna	Hale	8HS	14
008	Molly	Richards	8HS	13

...to select data we could use the following syntax:

```
SELECT first_name FROM students
```

...which would produce the following:

first_name
Sam
Rob
Tom
Megan
Alfie
Tess
Anna
Molly

This particular example will select all data in the column called 'first_name'.

As well as selecting a single column, you can also select multiple columns from a table…

```
SELECT first_name, surname FROM students
```

Would produce...

first_name	surname
Sam	Sampson
Rob	Dale
Tom	Franks
Megan	Pope
Alfie	Jones
Tess	Smith
Anna	Hale
Molly	Richards

...and if you want to select everything we can use the wildcard (*).

```
SELECT * FROM students
```

Would select the entire table…

student_ID	first_name	surname	group	age
001	Sam	Sampson	9AB	14
002	Rob	Dale	8SW	13
003	Tom	Franks	9AB	13
004	Megan	Pope	8SW	13
005	Alfie	Jones	9DJ	14
006	Tess	Smith	9AB	14
007	Anna	Hale	8HS	14
008	Molly	Richards	8HS	13

The WHERE Clause

The "Where" clause is used with the 'Select' statement and is great if we want to drill down and retrieve specific data which match a given criteria.

The SQL in this example selects all rows in the table where the value in the field column matches '9AB'.

```
SELECT * FROM students WHERE group='9AB'
```

Would produce…

student_ID	first_name	surname	group	age
001	Sam	Sampson	9AB	14
003	Tom	Franks	9AB	13
006	Tess	Smith	9AB	14

The UPDATE Statement

The 'update' statement allows us to update / alter data that is already stored in the database. The WHERE statement sets the particular record to be updated. If there is no value in the WHERE statement, then all the records will be updated.

The SQL in this example will update the table called students, changing the surname to 'Goodie', but only in the row(s) where the first name matches the value 'Tess'.

student_ID	first_name	surname	group	age
001	Sam	Sampson	9AB	14
002	Rob	Dale	8SW	13
003	Tom	Franks	9AB	13
004	Megan	Pope	8SW	13
005	Alfie	Jones	9DJ	14
006	Tess	Smith	9AB	14
007	Anna	Hale	8HS	14
008	Molly	Richards	8HS	13

UPDATE students SET surname = 'Goodie' WHERE first_name='Tess'

Would produce

student_ID	first_name	surname	group	age
001	Sam	Sampson	9AB	14
002	Rob	Dale	8SW	13
003	Tom	Franks	9AB	13
004	Megan	Pope	8SW	13
005	Alfie	Jones	9DJ	14
006	Tess	Goodie	9AB	14
007	Anna	Hale	8HS	14
008	Molly	Richards	8HS	13

...notice that Tess now has the surname 'Goodie'.

The INSERT Statement

If we want to insert new data into our database we can use the INSERT INTO statement.

student_ID	first_name	surname	group	age
001	Sam	Sampson	9AB	14
002	Rob	Dale	8SW	13
003	Tom	Franks	9AB	13
004	Megan	Pope	8SW	13
005	Alfie	Jones	9DJ	14
006	Tess	Smith	9AB	14
007	Anna	Hale	8HS	14
008	Molly	Richards	8HS	13

INSERT INTO students VALUES ('009', 'Jim', 'Smith', '9AB', '14')

Would produce

student_ID	first_name	surname	group	age
001	Sam	Sampson	9AB	14
002	Rob	Dale	8SW	13
003	Tom	Franks	9AB	13
004	Megan	Pope	8SW	13
005	Alfie	Jones	9DJ	14
006	Tess	Smith	9AB	14
007	Anna	Hale	8HS	14
008	Molly	Richards	8HS	13
009	Jim	Smith	9AB	14

+

The SQL in this example will insert a new row into the table called students, with the values specified in the brackets. They will be entered in the order in which they are provided.

The DELETE Statement

Unsurprisingly this statement is used to delete a record / records from a database.

student_ID	first_name	surname	group	age
001	Sam	Sampson	9AB	14
002	Rob	Dale	8SW	13
003	Tom	Franks	9AB	13
004	Megan	Pope	8SW	13
005	Alfie	Jones	9DJ	14
006	Tess	Smith	9AB	14
007	Anna	Hale	8HS	14
008	Molly	Richards	8HS	13
009	Jim	Smith	9AB	14

DELETE FROM students WHERE first_name='Jim'

Would produce

student_ID	first_name	surname	group	age
001	Sam	Sampson	9AB	14
002	Rob	Dale	8SW	13
003	Tom	Franks	9AB	13
004	Megan	Pope	8SW	13
005	Alfie	Jones	9DJ	14
006	Tess	Smith	9AB	14
007	Anna	Hale	8HS	14
008	Molly	Richards	8HS	13

-

The SQL in this example will delete the row(s) in the table called students, where the value in the field called 'first_name' matches 'Jim'.

The LIKE Statement

The LIKE statement is used with a WHERE statement in order to find patterns within the data in a column. For example it can be used along with a wildcard '%' to find all names that start with a particular letter.

student_ID	first_name	surname	group	age
001	Sam	Sampson	9AB	14
002	Rob	Dale	8SW	13
003	Tom	Franks	9AB	13
004	Megan	Pope	8SW	13
005	Alfie	Jones	9DJ	14
006	Tess	Smith	9AB	14
007	Anna	Hale	8HS	14
008	Molly	Richards	8HS	13
009	Jim	Smith	9AB	14

student_ID	first_name	surname	group	age
003	Tom	Franks	9AB	13
006	Tess	Smith	9AB	14

Would produce

Example:

```
SELECT * FROM students
WHERE first_name LIKE 'T%'
```

The SQL in this example will select all row(s) in the table called students, where the value in the field called 'first_name' starts with the letter 't'.

The JOIN Statement

The JOIN statement is used to select records in two separate tables that both have the same value.

Example:

```
SELECT Orders.OrderNo, Customers.Surname
FROM Orders
JOIN Customers ON Orders.CustomerID = Customers.CustomerID
```

Customers Table

Customer ID	First Name	Surname	House	Street	Town	Post Code
1	Dave	Jones	12	Jubilee Rd	Exeter	EX2 5FD
2	Mandy	Williams	63	Fore Street	Taunton	TA8 9GH
3	Sam	Smith	233	Grange Rd	Torquay	TQ1 9DR
4	Holly	Moon	15	House Lane	Exmouth	EX9 7HG
5	Alfred	Richie	7	Good Street	Exeter	EX1 6GF

Products Table

Item ID	Item	Price
1	Monitor	100
2	Keyboard	22
3	PC	500
4	Gamepad	12

link

link

Orders Table

Order No.	Date	Customer ID	Item ID
1	12-03-2015	1	1
2	12-03-2015	2	2
3	13-03-2015	3	3
4	14-03-2015	4	1
5	14-03-2015	5	4
6	15-03-2015	1	4

Order No.	Surname
1	Jones
2	Williams
3	Smith
4	Moon
5	Richie
6	Jones

Would produce

The SQL in this example will join the order numbers with the corresponding customer surnames.

The DROP Statement

We have already seen how the DELETE statement can be used to delete data in a table / database. What is important to recognise is that when data is deleted, the actual structure of the table/database will remain.

The DROP statement however, is used to delete either a table or a database's data along with the table/database structures.

student_ID	first_name	surname	group	age
001	Sam	Sampson	9AB	14
002	Rob	Dale	8SW	13
003	Tom	Franks	9AB	13
004	Megan	Pope	8SW	13
005	Alfie	Jones	9DJ	14
006	Tess	Smith	9AB	14
007	Anna	Hale	8HS	14
008	Molly	Richards	8HS	13
009	Jim	Smith	9AB	14

No more table!

Example:

```
DROP TABLE students
```

Would produce

The SQL in this example will remove the table 'students' from existence.

ORDERING QUERY RESULTS

After selecting a set of records during a query, sometimes it is useful to order the results by a particular field. This can be done using the SQL statement **ORDER BY**.

For example, for the given table of data, if we wished to select all students in a particular group and order the result of the query by surname we could use the following SQL statement:

student_ID	first_name	surname	group	age
001	Sam	Sampson	9AB	14
002	Rob	Dale	8SW	13
003	Tom	Franks	9AB	13
004	Megan	Pope	8SW	13
005	Alfie	Jones	9DJ	14
006	Tess	Smith	9AB	14
007	Anna	Hale	8HS	14
008	Molly	Richards	8HS	13

```
SELECT * FROM students
WHERE group = 8HS
ORDER BY surname ASC
```

DESC can be used to place the results in descending order.

TRANSACTION PROCESSING

A single operation that is performed on a database is known as a transaction. For example, a single operation carried out on a database booking system could be the booking of a hotel room, which cannot complete until payment has been made. This transaction therefore could involve more than one process (1. booking 2. after payment confirmed).

It is therefore vital that transactions fully complete and cannot partially complete (e.g. booking without payment).

To ensure that a database processes transactions reliably, it will follow the ACID principles.

ACID (Atomicity, Consistency, Isolation, Durability)

Atomicity

Atomicity is the requirement that a transaction must be fully processed or not processed at all, removing the possibility of a partially completed transaction.

Consistency

Consistency is the requirement that a transaction cannot compromise referential integrity.

For example, consistency will ensure that an order cannot be placed by a customer (who exists in the customer table) for a product that does not exist in the products table. In the same way, consistency would ensure that a product cannot be deleted from the products table if it already exists in a previous order.

Isolation

Isolation is the requirement that a transaction is performed in isolation so that no other process can interfere until the transaction is completed. In practical terms, to facilitate this, whilst a transaction is operating on a record, that record is locked, stopping others writing to the record until the transaction has been completed and committed to memory.

Durability

Durability is the requirement that once a transaction has completed, the database change is immediately written to secondary storage so that the transaction cannot be lost in the event of a system failure or power cut.

Record Locking

There could be a situation where a record is accessed and edited by two (or more) different users, simultaneously, which can cause issues.

Example:

For a given customer record, if a database user altered the customer's surname and another altered the customer's address and both saved their changes at the same time, what changes would persist?

The likelihood is that one of the changes would not be made. Inconsistencies would therefore be likely.

Record locking ensures that the issues outlined above do not happen.

Record locking is the process by which a record is locked the moment a user retrieves a record, so that no other user can access the record whilst it is being edited/updated. The moment that transaction has completed, the record is then unlocked for others to access.

Questions

'General Understanding' Questions

1. Write the SQL command that would create a table called 'teachers' and include the fields 'teacherID', 'teacher_surname', 'teacher_subject'. [4]
2. For the given database table (below) called 'students', write an SQL command which will update the surname of 'Sam Sampson', to 'Hickman'. [2]

student_ID	first_name	surname	group	age
001	Sam	Sampson	9AB	14
002	Rob	Dale	8SW	13
003	Tom	Franks	9AB	13
004	Megan	Pope	8SW	13

3. Explain the difference between a record and an entity. [3]

4. For the database below (called 'students'), write the SQL statement that would find all surnames names that start with the letter S. [2]

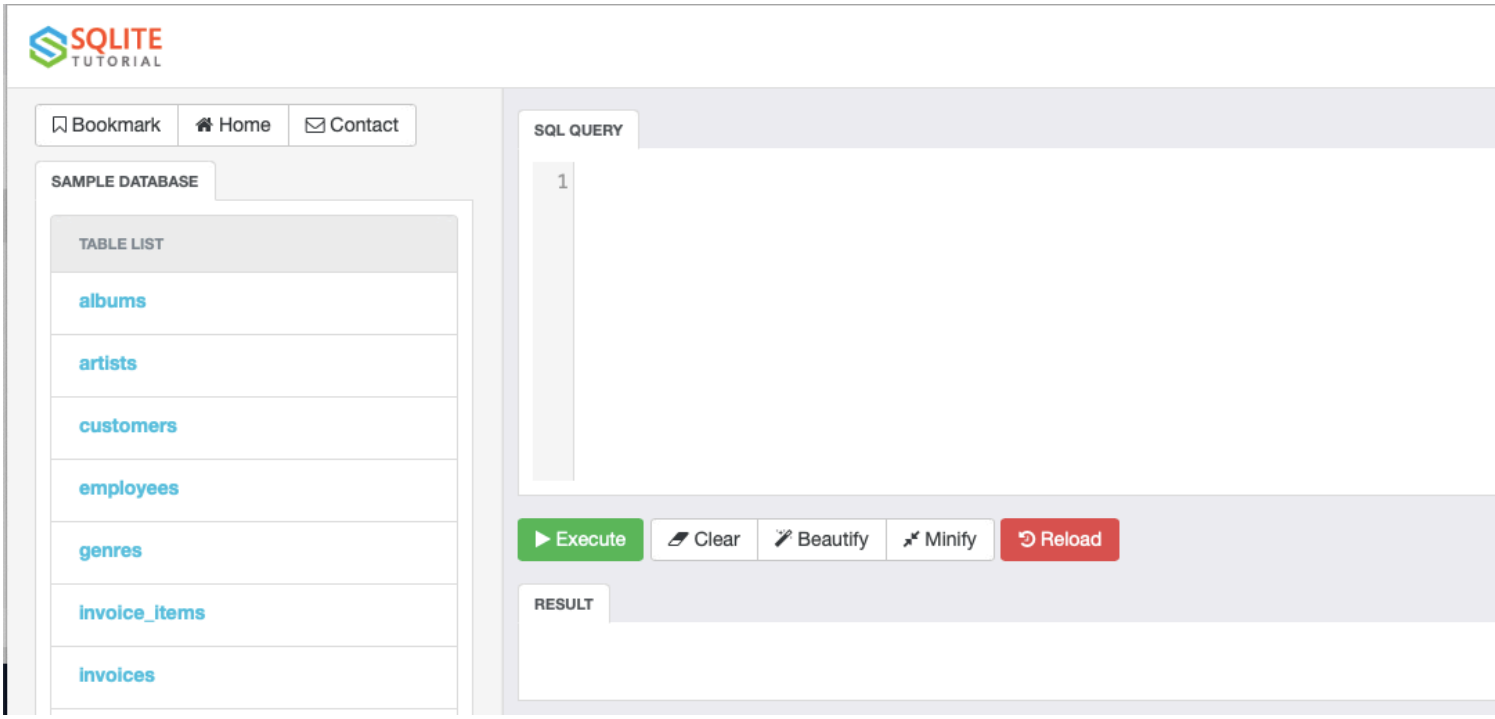
student_ID	first_name	surname	group	age
001	Sam	Sampson	9AB	14
002	Rob	Dale	8SW	13
003	Tom	Franks	9AB	13
004	Megan	Pope	8SW	13
005	Alfie	Jones	9DJ	14
006	Tess	Smith	9AB	14
007	Anna	Hale	8HS	14
008	Molly	Richards	8HS	13
009	Jim	Smith	9AB	14

5. What is meant by the term 'Transaction' in relation to databases? [2]
6. What are the ACID principles and what are their purpose? [8]
7. What is 'Record Locking' and why is this an important feature of database management? [2]

Practical Exercises.

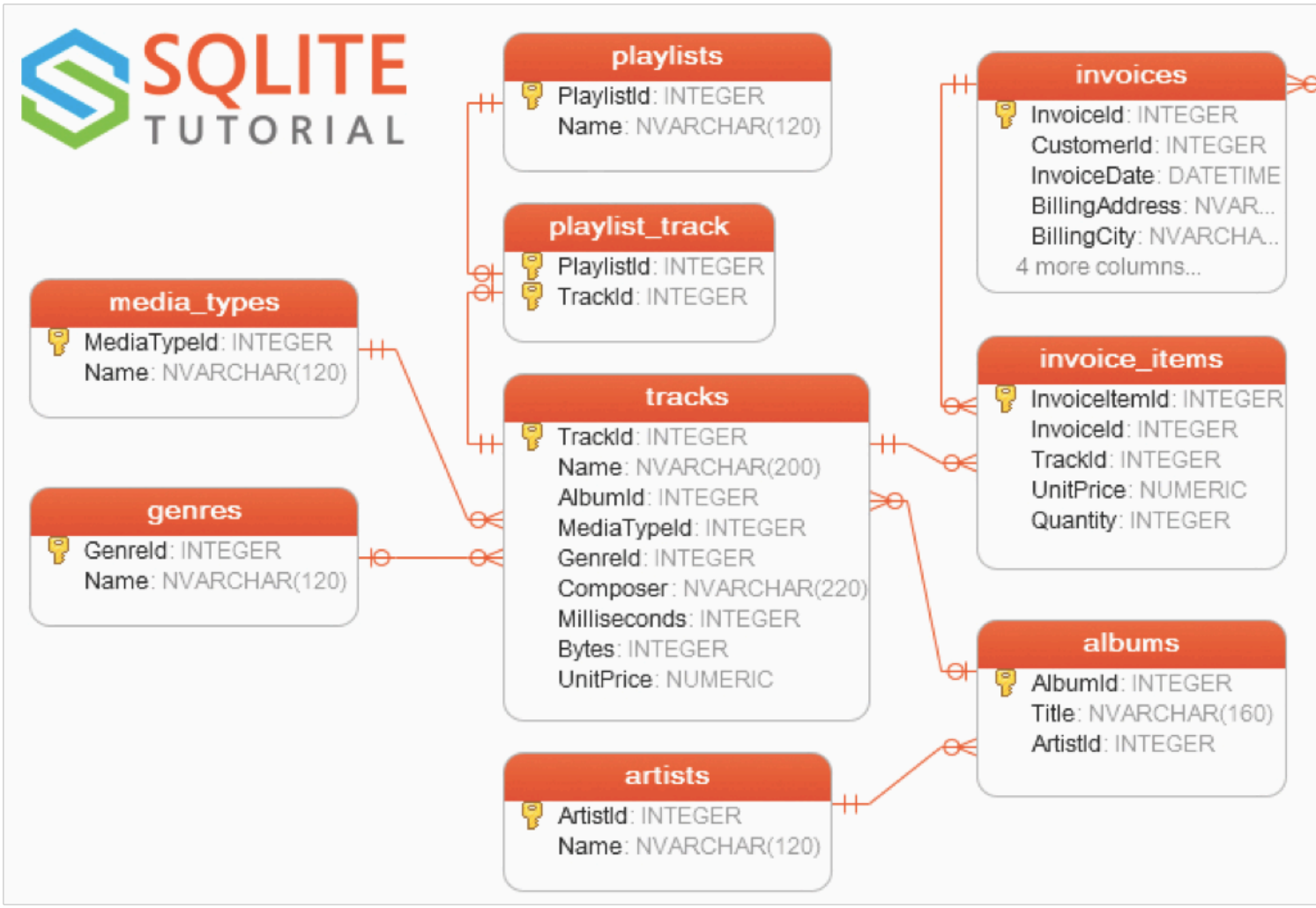
SQL and database exercises:

Go to the SQLiteTutorial tryit page that allows you to practice and improve your SQL skills:
https://www.sqlitetutorial.net/tryit/



This allows you to create and run SQL commands against a sample database that contains example data.

The structure of the database is represented in this Entity Relationship Diagram, ERD:



We will focus on the Invoices and Invoice_items tables:

a) Select the Invoices and CustomerIDs for the invoices to the UK newest items first?

b) Join the Invoices and invoice_items tables to display invoices and items for the UK?

c) Add the Customer Name and email to the list of invoices and items for the UK?

d) Add the Track Names to the query in c) above and sort the list by Track Name?

Checklist:

- ☐ Date and title, clearly presented
- ☐ Spelling & grammar checked
- ☐ Question numbers in the margin
- ☐ Handwriting neat & legible
- ☐ Punctuation / Capital letters

Keywords / Key Terms:

SQL:. *Structured Query Language- A standard database language used to create and manipulate databases.*

Query:. *A request for data from a database.*

Transaction: *A single operation that is performed on a database.*

- **State/Identify/Give/Name**: Simply label a diagram, fill out a table or write a few words
- **Describe**: Describing is 'saying what you see' (E.G.: A computer will have a CPU, Primary and Secondary storage etc)
- **Explain:** Explaining is 'saying WHY/HOW something is like that'. (E.G.: A computer will have a CPU so that it can process all of the data the computer needs to perform a range of tasks. Primary and Secondary storage is needed because…)
- **Discuss**: Discussing is 'looking at two sides of an issue, weighing up the two views and giving a conclusion'. Often these require a mini essay answer. (E.G.: New technology could be seen as being bad for the environment because…, but on the other hand, new technology has led to… In conclusion I believe that…)
- Describe/Explain/Discuss **using examples**: Finally, if you are asked to give examples in any of these types of questions – YOU MUST **GIVE EXAMPLES**!

Stick answer sheet here

Reflections: ***Score:*** / ***Percentage:*** % ***Grade:*** ***Progress:*** On / Above / Below

What Went Well?:

- ☐ I demonstrated a good level of understanding.
- ☐ I responded to the command words effectively.
- ☐ My answers were detailed / were written in depth.
- ☐ My work was well presented / legible.
- ☐ My answers effectively incorporated technical terminology.
- ☐ My responses were well structured / organised.
- ☐ My revision strategy was effective as I showed depth of understanding in my answers.
- ☐ My answers contained enough points / examples / explanations to achieve the marks available.

Even Better If…:

- ☐ My answers need to be more accurate / specific.
- ☐ I must respond correctly to the command words.
- ☐ My answers need more detail / greater depth.
- ☐ I must take greater care over my work / write neatly.
- ☐ I must incorporate key terminology into my answers.
- ☐ I must better organise my answers to improve its clarity.
- ☐ I need to improve my revision strategy, as I did not demonstrate a depth of understanding in my answers.
- ☐ My answers didn't contain enough points / examples / explanations to achieve the marks available.

Further thoughts: