10.4 SQL Mark Scheme

Q1.

All marks AO1 (understanding)

Level	Description	Mark Range
4	A line of reasoning has been followed to produce a coherent, relevant, substantiated and logically structured response. The response covers all three areas indicated in the guidance below and in at least two of these areas there is sufficient detail to show that the student has a good level of understanding. To reach the top of this mark range, a good level of understanding must be shown of all three areas.	10-12
3	A line of reasoning has been followed to produce a coherent, relevant, substantiated and logically structured response which shows a good level of understanding of at least two areas indicated in the guidance below.	7-9
2	A limited attempt has been made to follow a line of reasoning and the response has a mostly logical structure. At least four points have been made. Either a good level of understanding of one area from the guidance has been shown or a limited understanding of two areas.	4-6
1	A few relevant points have been made but there is no evidence that a line of reasoning has been followed. The points may only relate to one or two of the areas from the guidance or may be made in a superficial way with little substantiation.	1-3

<u>**Guidance – Indicative Response**</u>

For each guidance point, if the student expands on the point to explain in what way the measure will improve performance then this can be considered to be a second point. For example:

- “Using a processor with more cores” is one point.
- “Using a processor with more cores which will be able to execute multiple instructions simultaneously” is two points.

Note that just “faster” is not enough to count as an expansion point without an explanation of why.

1. Server Hardware

Replace the processor with one which has more cores

Replace the processor with one which has more cache memory // increase the amount of cache memory

Replace the processor with one which runs at a faster clock speed **NE.** faster processor

Use a parallel processor architecture // use more processors which can work in parallel

Use a processor with a bigger word size

Use a processor that makes (better) use of pipelining

Install more RAM // main memory // primary memory

Use RAM // main memory // primary memory with a faster access time

Replace HDDs with SSDs // Replace HDDS with HDDs that can read data at a faster rate

Defragment the HDD

Replace the motherboard with one which has buses which run at a faster clock speed

Replace the motherboard with one which has more lines in the data bus

Use the Harvard architecture

Distribute the processing across multiple servers

2. Network

Replace the network cable with cable that has a higher bandwidth // replace copper cable with fibre-optic cable **A.** Ethernet cable for fibre-optic NE. higher bandwidth network

Replace any wireless / WiFi connections with wired ones

Replace the network cards with ones that can transmit data at a higher bitrate

Consider the overall network design eg how the network is divided into subnets **A.** split the network into subnets

Use a star topology (instead of a bus)

Consider using a more efficient protocol for the data across the network

Add additional wireless access points

3. Database and Software

Use a more efficient technique for controlling concurrent access to the database // replace record/table locks with serialisation/timestamp ordering/commitment ordering

Replace the database software with software that uses more efficient algorithms for tasks **A.** examples eg replace linear search with binary search

Use the index feature of the database to speed up searching on fields that are commonly used for this purpose

Rewrite the database software in a language that is suitable for concurrent execution // use a functional programming language for the database software

Ensure the software is compiled rather than executed by an interpreter // rewrite the software in assembly language/machine code

Review the conceptual model of the database to see if it contains any inefficiencies such as data redundancy that could be eliminated **A.** normalise the database design

Consider if it would be appropriate to sacrifice normalisation of the conceptual model to improve performance

Use a non-relational database system **A.** examples eg NoSQL

Distribute the data across multiple servers

Try to reduce the amount of other (unrelated) software that might be running on the database server at the same time

Try to reduce the number of database accesses that need to be made simultaneously // run some tasks at quiet times / overnight

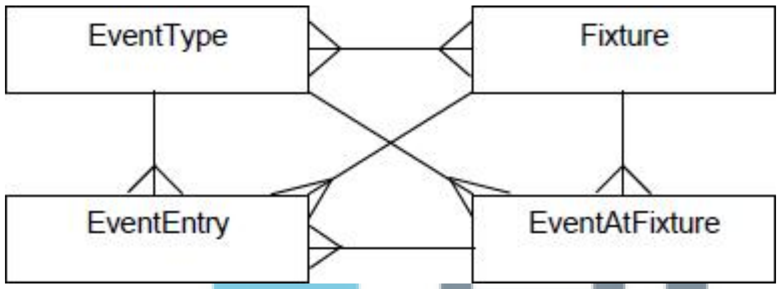
Purge / archive data that is no longer necessary / in use

[12]

Q2.

(a) **All marks AO2 (analyse)**

1 mark for any one correctly drawn relationship
OR
2 marks for three relationships drawn correctly
Max 1 if more than three relationships drawn and any are incorrect
A. a many:many relationship drawn between EventType and Fixture as this is modelled by a linking relation (EventAtFixture)



2

(b) **All marks AO2 (analyse)**

There is no data type for the primary key / `AthleteID` // The primary key / `AthleteID` needs a data type;
The data type is specified before the fieldname // fieldname should precede the data type // `PRIMARY KEY` is specified before the fieldname; **A.** an example of a specific field and data type which are the wrong way around
There is a semi-colon missing at the end;

Max 2

2

(c) **All marks AO1 (understanding)**

*Minimise data duplication // no unnecessary repeated data; **A.** reduce for minimise **R.** eliminate
*Eliminate data redundancy; **A.** reduce/minimise for eliminate
Eliminate data inconsistency // improve consistency // avoid inconsistency problems;
Eliminate update anomalies; **A.** example in context **A.** updates only need to be made in one place
Eliminate insertion anomalies; **A.** example in context
Eliminate deletion anomalies; **A.** example in context
NE. easier to update/insert/delete without concrete example or good explanation
NE. fewer errors when updating / inserting / deleting without concrete example or good explanation
NE. saving space / memory

NE. easier / faster to query

Note: Only award one of the two marks with *. ie a response cannot get two marks for discussion of only duplication and redundancy

2

(d) **3 marks for AO2 (analyse) and 2 marks for AO3 (programming)**

Mark Scheme

AO2 (analyse) – 3 marks:

1 mark for correctly analysing the data model and identifying the tables that data needs to be extracted from (`Athlete`, `EventEntry`, `Fixture`) and the fields that need to be extracted (`Surname`, `Forename`, `DateOfBirth`), and including these and no other tables or fields in the query

1 mark for correctly identifying how the data in the required tables should be combined to produce the desired result (the linking conditions - `Athlete.AthleteID = EventEntry.AthleteID` and `EventEntry.FixtureID = Fixture.FixtureID`)

1 mark for identifying the correct condition to use within the model for the `FixtureDate` field (`FixtureDate = "17/09/2018"`) and for using the correct logical operators between all of the conditions (if a linking condition is also used)

Note: The AO2 marks for analysing the data model should be awarded regardless of whether correct SQL syntax is used or not as they are for data modelling, not syntactically correct SQL programming

AO3 (programming) – 2 marks:

1 mark for fully correct SQL in two of the four clauses (`SELECT`, `FROM`, `WHERE`, `ORDER BY`)

OR

2 marks for fully correct SQL in all four clauses (`SELECT`, `FROM`, `WHERE`, `ORDER BY`)

Note: For an SQL clause to be counted as "fully correct", the syntax of the clause must be correct and the relevant AO2 decisions must also have been taken for the clause. eg the `SELECT` clause must have the correct fields in it only

Example Solutions

Example 1
```
SELECT Surname, Forename, DateOfBirth
FROM Athlete, EventEntry, Fixture
WHERE FixtureDate = "17/09/2018"
  AND Athlete.AthleteID = EventEntry.AthleteID
  AND EventEntry.FixtureID = Fixture.FixtureID
ORDER BY Surname
```

Example 2
```
SELECT Surname, Forename, DateOfBirth
```

```
FROM Athlete INNER JOIN EventEntry ON Athlete.AthleteID =
  EventEntry.AthleteID INNER JOIN Fixture ON
  EventEntry.FixtureID = Fixture.FixtureID
WHERE FixtureDate = "17/09/2018"
ORDER BY Surname
```

Overall Max 4 if solution does not work fully

Additional Guidance

AO2 marks:

Mark(s) can be awarded for the correct logical conditions even if the required tables are not identified as being used by the query

Allow the inclusion of the unnecessary table `EventAtFixture` for AO2 and AO3 marks but only if it is linked to the other tables with a correct condition i.e. `EventAtFixture.FixtureID` = `Fixture.FixtureID` or alternatively `EventAtFixture.FixtureID` = `EventEntry.FixtureID` or both

Allow omission of delimiters around date for AO2 marks only.

AO3 marks:

A. table names before fieldnames separated by a full stop
A. use of `Alias / AS` command e.g. `FROM Athlete AS A` then use of `A` as the table name but note that command `Alias` is not required e.g. `FROM Athlete A`
A. `INNER JOIN` written as one word i.e. `INNERJOIN`
A. `ORDER BY` written as one word i.e. `ORDERBY`
A. `ASC` at end of `ORDER BY` clause but **R.** `ASCENDING`
A. insertion of spaces into fieldnames
A. use of " # or ' as delimiters around date – **Note**: delimiters are required for AO3 correct code but not for AO2 mark for date condition
A. date parts given in any order so long as they are separated by /
A. 18 instead of 2018 in year
I. unnecessary brackets
DPT for unnecessary punctuation – allow one semicolon at the very end of the statement, but not at the end of each clause
DPT for fieldname before table name

Refer responses using nested SQL queries to team leaders.

5

[11]

Q3.

(a) **Mark is for AO2 (analyse)**

CarRegNo and JobDate;
A. Just both these attribute names written with no further explanation
R. “CarRegNo or JobDate”

1

(b) **1 mark for AO2 (analyse) and 1 mark for AO1 (understanding)**

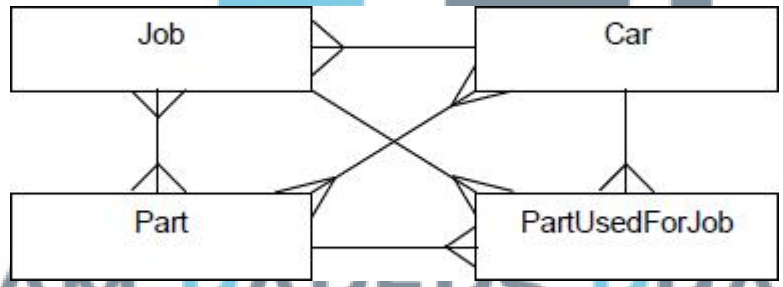
AO2 (analyse) – 1 mark:
A person may own more than one car // a person may bring different cars to the garage;

It might be desired to store details of an owner when the car they own is not yet known;
A. A car might be owned by more than one person (at different times)
A. Easier to transfer car from one owner to another

AO1 (understanding) – 1 mark:
Avoid storing owner details once for each car they own / multiple times;
Avoid having to input owner details once for each car they own;
To transfer car between owners would only have to change one attribute in the car relation;
Minimise data duplication // no unnecessary repeated data; **A**. Reduce for minimise
Eliminate data redundancy; **A**. Reduce/minimise for eliminate
Eliminate data inconsistency // improve consistency // avoid inconsistency problems;
Eliminate update anomalies; **A**. Example in context
Eliminate insertion anomalies; **A**. Example in context
NE. Fewer errors when updating/inserting/deleting without concrete example or good explanation
NE. Saving space/memory
NE. Easier to query

2

(c) **All marks AO2 (analyse)**
1 mark for any one correctly drawn relationship **OR**
2 marks for three relationships drawn correctly
MAX 1 if more than three relationships drawn and any are incorrect



2

(d) **1 mark for AO2 (analyse) and 2 marks for AO3 (programming)**

Mark Scheme

AO2 (analyse) – 1 mark:

1 mark for correctly identifying the table in the data model that needs to be updated (`Job`) and the condition that should be used to identify the correct record in the table to update (`JobID = 206`).

Note: The AO2 mark for analysing the data model should be awarded regardless of whether correct SQL syntax is used or not as it is for data modelling, not syntactically correct SQL programming

AO3 (programming) – 2 marks:

1 mark for correct SQL syntax in two of the three clauses (UPDATE, SET, WHERE) **OR**
2 marks for fully correct SQL

Example Solution

```
UPDATE Job
SET JobDuration = "01:30"
WHERE JobID = 206
```

Additional Guidance

AO3 marks:

A Any type of quotation marks or hashes for delimiters for `JobDuration` or no delimiters
A. The value `206` if it is delimited by any type of quotation mark
A. Any sensible format for the time data eg `"01.30"`, `"1:30"`, `"1:30.00"` etc.
A. Time given as a decimal ie 1.5
A. Table name given before fieldname
I. Quotation marks around fieldnames
I. Any attempt to also change value of `InGarage`

3

(e) **All marks AO3 (programming)**

Method 1:

```
INSERT INTO PartUsedForJob
VALUES (206,12,2)
```

Method 1:

```
INSERT INTO PartUsedForJob (JobID, PartID, QuantityUsed)
VALUES (206,12,2)
```

1 mark for correct `INSERT INTO` clause
1 mark for correct `VALUES` clause
MAX 1 if SQL not fully working eg because of extra clauses
A. List of fields in any order for method 2, but to get the VALUES mark in method 2, order of fields list in `INSERT INTO` must match order of values in `VALUES`
A. The value(s) `206` and `12` if they are delimited by any type of quotation mark

2

(e) **3 marks for AO2 (analyse) and 2 marks for AO3 (programming)**

Mark Scheme

AO2 (analyse) – 3 marks:
1 mark for correctly analysing the data model and identifying the tables that data needs to be extracted from (`Part, PartUsedForJob`) and the fields that need to be extracted (`PartID, Description, Price, QuantityUsed`), and including these and no other tables or fields in the query **A**. Including the table `Job` which is not needed, as long as it is correctly linked in by a condition
1 mark for correctly identifying how the data in the required tables should be combined to produce the desired result (the linking condition - `PartUsedForJob.PartID = Part.PartID`)
1 mark for identifying the correct conditions to use within the model for the `JobID` field (`JobID = 93`) and for using the correct logical operators between all of the conditions (if a linking condition is also used)

Note: The AO2 marks for analysing the data model should be awarded regardless of whether correct SQL syntax is used or not as they are for data modelling, not syntactically correct SQL programming

AO3 (programming) – 2 marks:

1 mark for correct SQL in two or three of the four clauses (SELECT, FROM, WHERE, ORDER BY) **OR**

2 marks for fully correct SQL

Example Solutions

Example 1

```
SELECT PartID, Description, Price, QuantityUsed
FROM Part, PartUsedForJob
WHERE JobID = 93
  AND PartUsedForJob.PartID = Part.PartID
ORDER BY PartID
```

Example 2

```
SELECT PartID, Description, Price, QuantityUsed
FROM Part INNER JOIN PartUsedForJob ON
PartUsedForJob.PartID = Part.PartID
WHERE JobID = 93
ORDER BY PartID
```

Overall MAX 4 if solution does not work fully

Additional Guidance

AO2 marks:

Mark(s) can be awarded for the correct logical conditions even if the required tables are not identified as being used by the query
Ignore unnecessary clause `PartUsedForJob.JobID = Job.JobID`

AO3 marks:

Accept table names before fieldnames separated by a full stop.
Accept use of `Alias/AS` command eg `FROM Part AS P` then use of `P` as the table
name but note that command `Alias` is not required eg `FROM Part P`.
Accept `INNER JOIN` written as one word ie `INNERJOIN` or just as `JOIN`
Accept `ORDER BY` written as one word ie `ORDERBY`.
Accept `ASC` at end of `ORDER BY` clause.
Accept insertion of spaces into fieldnames.
Accept use of " or ' as delimiters around number 93.
Ignore unnecessary brackets.

DPT for unnecessary punctuation – allow one semicolon at the very end of the statement, but not at the end of each clause.
DPT for fieldname before table name.
For the **DPT** points, the penalisation is in terms of number of clauses of SQL code not marks ie if fieldname is before table name in two out of four clauses of SQL then this could count as three clauses of correct SQL

Refer responses using nested SQL queries to team leaders.

Refer responses using RIGHT JOIN OR LEFT JOINT to team leaders.

5

(f) **All marks AO2 (analyse)**

1 mark: Create a new relation to identify which make/model(s) of car each part can be fitted to;
A. Use of a relation name that clearly identifies the purpose eg `PartToFitMakeModel` instead of an explanation
A. If it is just stated that a new relation is creation if the attributes in the relation make its purpose clear
NE. A relation to link the `Part` and `Car` relations

2 marks from:
Store the attributes `PartID, Make` and `Model` in the new relation;
I. Inclusion of additional attributes
Make the `PartID, Make` and `Model` / all the attributes the entity identifier;
A. The creation of a new field as an entity identifier for this relation if it is explained that a constraint would also need to be added to ensure that it is not possible to record twice in the relation that a particular part could be fitted to a particular make and model of car
Accept answers by example, such as: `PartToFitMakeModel(`<u>`PartID, Make, Model`</u>`)`

<u>**Alternative Response**</u>

1 mark:
Create two new relations, one to associate an entity identifier with each make and model of car (eg <u>`MakeModelID`</u>) and one to link the parts to this new relation
A. If it is just stated that new relations will be created if the attributes in the relations make their purpose clear

2 marks from:
Store the attributes `Make` and `Model` with a new entity identifier (eg `MakeModelID`) in one of the new relations;
Store the `PartID` in the other new relation together with the entity identifier from the first new relation (eg `MakeModelID`);
Make the `PartID` and `MakeModelID` the entity identified in the second new relation;
A. The creation of a new field as an entity identifier for this relation if it is explained that a constraint would also need to be added to ensure that it is not possible to record twice in the relation that a particular part could be fitted to a particular make and model of car
Accept answers by example, such as: `UniqueMakeModel(`<u>`MakeModelID`</u>`, Make, Model)` and `PartToFitMakeModel(`<u>`PartID, MakeModelID`</u>`)`

A. Table or entity for relation.
A. Field for attribute.
A. Primary key for Entity Identifier.

3

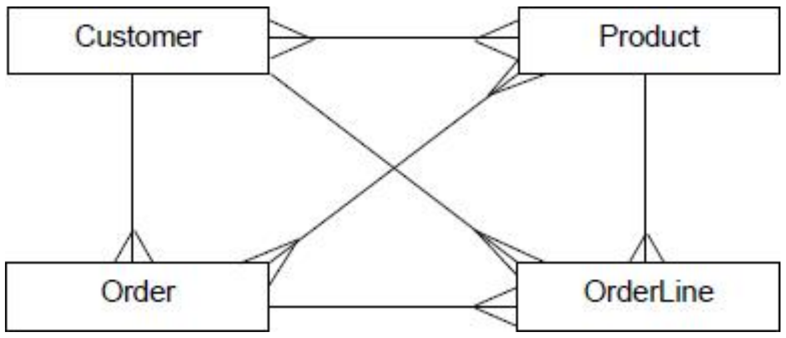
[18]

Q4.

(a) Only one (type of) product per order // Must make new order for each (type of) product;
as ProductNumber / product details stored in relation that has OrderNumber as primary key / product relation directly related to order relation // there is transitive/(**A**.non-key) dependency // as relations not (fully) normalised;
Difficult to query // requires (unnecessarily) complex queries;
as contains repeating groups (of attributes); **A**. Either way round
A. Table for relation

Max 2

(b)



1 mark for each correct relationship, up to **MAX 3**
MAX 2 if more than three relationships drawn and any of them are incorrect

3

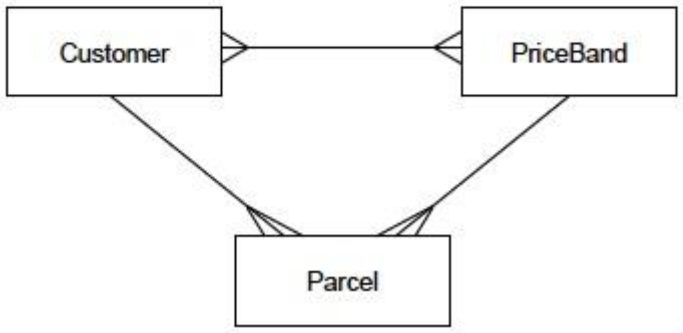
(c) Sequence of instructions / program / code; **NE**. Programming language **Note**: Do not award mark for program if candidate clearly means HTML
which is executed/run/interpreted on the server (instead of the client);
executed/run/interpreted when a web page is requested;
to generate a web page (and its contents) which the server returns to the client // generating of dynamic web pages;

Max 5

[10]

Q5.

(a) **1 mark** for any one correct relationship drawn
2 marks for all three correct relationships drawn
I. Any additional writing on diagram



2

(b)
```
UPDATE PriceBand
SET Price = 5.99
WHERE ServiceSpeed = "Express"
  AND MinWeight = 1000
  AND MaxWeight = 4999
```
1 mark for `UPDATE PriceBand`
1 mark for `SET Price = 5.99`

1 mark* for `ServiceSpeed = "Express"`
1 mark* for either `MinWeight = 1000` or `MaxWeight = 4999` (or both joined by `AND`). **A.** use of `>=` and `<=` instead of `=` if conditions given for both `MinWeight` and `MaxWeight`.

To award both marks indicated by * symbol, the conditions must be joined by `AND`s.
A. Double or single quotes around `Express`
A. `Express` written in any case
A. `£` symbol before `5.99`
A. Table names before fieldnames

DPT for fieldname before table name.
DPT for unnecessary punctuation e.g. quotes where they should not appear. Allow one semicolon at the very end of the statement, but not at the end of each clause.
DPT use of incorrect equality operator e.g. `==`

4

(c) **Alternative 1**

```
SELECT DateSent, Postcode, ServiceSpeed, Price
FROM Parcel, PriceBand
WHERE CustomerID = 109
   AND Parcel.ServiceSpeed = PriceBand.ServiceSpeed
   AND Parcel.Weight >= PriceBand.MinWeight
   AND Parcel.Weight <= PriceBand.MaxWeight
ORDER BY DateSent
```

Alternative 2

```
SELECT DateSent, Postcode, ServiceSpeed, Price
FROM Parcel INNER JOIN PriceBand ON
Parcel.ServiceSpeed = PriceBand.ServiceSpeed
   AND Parcel.Weight >= PriceBand.MinWeight
   AND Parcel.Weight <= PriceBand.MaxWeight
WHERE CustomerID = 109
ORDER BY DateSent
```

1 mark for `SELECT` clause with correct four fields
1 mark for `FROM` clause with correct two tables
1 mark for `CustomerID = 109`
1 mark for
`Parcel.ServiceSpeed=PriceBand.ServiceSpeed`
1 mark for `Parcel.Weight >= PriceBand.MinWeight AND Parcel.Weight <= PriceBand.MaxWeight`
1 mark for `ORDER BY DateSent`
MAX 2 of the 3 marks for conditions if not joined by `AND`s

Conditions linking the two tables can be present in either the `FROM` or `WHERE` clause or a mixture of both, as long as they are syntactically and logically correct.
Marks for correct files/tables in `SELECT` and `FROM` statements should not be awarded if additional fields/tables included, except allow the inclusion of the `CUSTOMER` table in the `FROM` statement so long as it has been correctly linked to the `PARCEL` table.
Marks can be awarded for the conditions in the `WHERE` statement even if the required tables are not present in the `FROM`.

A. Table names before fieldnames.
A. Use of `Alias/AS` command e.g. `FROM Parcel AS P` then use of `P` as table name (note some dialects of SQL do not require `AS` e.g. FROM `Parcel P`)
A. Insertion of spaces into fieldnames.
A. `109` with no delimiters or delimited using " or '.
A. Use of `BETWEEN` command for weight range e.g.
`Parcel.Weight BETWEEN PriceBand.MinWeight AND PriceBand.MaxWeight`
A. `ORDER BY` written as one word `ORDERBY`.
A. `ASC` at the end of `ORDER BY`.
I. Unnecessary brackets.

DPT for unnecessary punctuation – allow one semicolon at the very end of the statement, but not at the end of each clause.
DPT for fieldname before table name.
DPT use of incorrect equality operator e.g. ==

Refer responses using nested SQL queries to team leaders.

6

(d) Parcel(<u>ParcelID</u>, ServiceSpeed, Weight, DateSent, CustomerID, RecipientName, HouseNumber, Postcode)

PostcodeLookup(<u>Postcode</u>, Street, Town, County)

1 mark for identifying that a new PostcodeLookup relation is required. Purpose must be clear; it is not sufficient to just make a new relation. Purpose could be made clear by any one of: appropriate name of relation (**A.** Address), approximately the correct attributes (allow, for example, incorrect inclusion of house number or CustomerID) in relation or having Postcode as the primary key.
1 mark for correct attributes in PostcodeLookup relation and identifying the Postcode as the primary key
1 mark for correct attributes left in Parcel relation and correct primary key

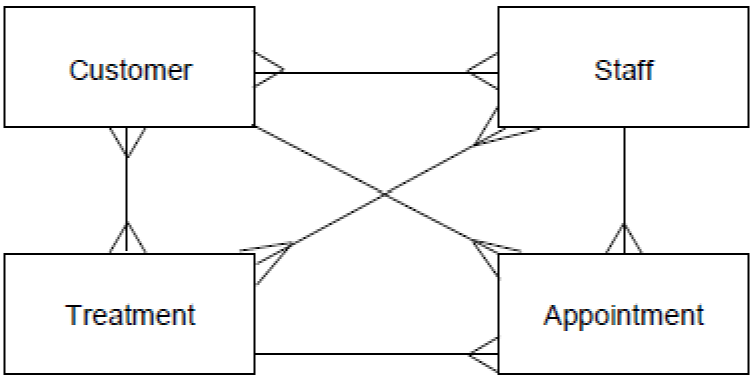
A. Answers given as SQL commands. As the question did not ask for this, perfect syntax is not required.
A. Alternative names for entities, so long as meaning is clear.
A. Spaces in entity and attribute names.
A. PostcodeLookup relation called Postcode even through this is the same as an attribute name.
A. Addition of unnecessary new relation for recipients which is not required for this question.
R. Do not award marks for correct attributes in a relation if additional attributes included.

3

[15]

Q6.

(a) **1 mark** for any one correct relationship drawn
2 marks for three correct relationships drawn



MAX 1 if any incorrect relationships drawn

2

(b)
```
TreatmentName VARCHAR(20) PRIMARY KEY
//
TreatmentName VARCHAR(20)
PRIMARY KEY(TreatmentName)
Price SMALLMONEY
TimeTaken INT
NeedsQualification BOOLEAN
```

1 mark for `TreatmentName`, with sensible data type and identified as primary key

1 mark for two other fields with sensible data types and lengths (if given) *OR* **2 marks** for all three other fields with sensible data types and lengths

A Any sensible types. Lengths do not need to be specified.

Valid alternative SQL types are:

- Alternative types for `TreatmentName`: char, nchar, nvarchar, ntext, longvarchar, varchar2, nvarchar2, text, tinytext, mediumtext, longtext
- Alternative types for `Price`: money, float, real, decimal, double, numeric, currency
  R integer only types for Price
- Alternative types for `TimeTaken`: tinyint, smallint, mediumint, integer, number, byte, time, date / time
- Alternative types for `NeedsQualification`: yes / no, bit, byte, bool, tinyint, enum(“yes”,“no”) − allow sensible alternative values in enum.

There should be a comma between the creation of each field, but ignore if these are missing, and accept a semi-colon at the end of the whole query.

Answers using a syntax that is clearly not SQL should be awarded zero marks. But:

- **ignore one punctuation error e.g. unnecessary colons between fieldname and type**
- **answers in SQL style syntax but using non-SQL data types can be credited but MAX 1 of 2 for other fields if any non-SQL types used.**

3

(c) **Alternative 1**
```
SELECT EmailAddress, Forename, Surname
FROM Customer, Appointment
WHERE TreatmentName = "Luxury Manicure"
   AND ApDate >= "01/01/2014"
   AND ApDate <= "31/12/2014"
   AND Customer.CustomerID =
        Appointment.CustomerID
```

1 mark for `SELECT` clause with correct three fields
1 mark for `FROM` clause with correct two tables
1 mark for `TreatmentName = "Luxury Manicure"`
1 mark for `ApDate >= "01/01/2014"`
1 mark for `ApDate = "01/01/2014"`
`AND ApDate = "01/01/2014"`
1 mark for `ApDate  "31/12/2013", ApDate`
`"01/01/2015"` as alternative date criteria.
Accept dates written in any format eg `"12-31-2013"`
Valid delimiters for dates are ", ' or #
Valid delimiters for strings are " or '
Valid symbols between date parts are /, - or no symbol
Ignore unnecessary clause
```
 Appointment.TreatmentName =
 Treatment.TreatmentName
```

Ignore unnecessary brackets.
Accept the following alternative methods for specifying the year, each of which are worth **2 marks**:

`YEAR(ApDate) = 2014,`
`DATEPART("yyyy",ApDate) = 2014` or no quotation marks
`DATEPART("yy",ApDate) = 2014` or no quotation marks
`DATEPART("Year",ApDate) = 2014` or no quotation marks
`LIKE(ApDate, "*2014*")` or `"*2014"`
`LIKE(ApDate, "%2014%")` or `"%2014"`
`LIKE "*/*/2014"`
`LIKE "*2014"`
`BETWEEN "01/01/2014" AND "31/12/2014"` or allow `"01/01/2015"` as upper limit

DPT for unnecessary punctuation – allow one semicolon at the very end of the statement, but not at the end of each clause.
DPT missing delimiters around data values, eg no quotation marks around dates.
DPT for fieldname before table name.

Refer responses using nested SQL queries to team leaders.

6

[11]

Q7.

(a) *Declaring PolicyNumber as primary key:*

```
PolicyNumber INT PRIMARY KEY(NOT NULL)

  / /
  PolicyNumber INT
  PRIMARY KEY(PolicyNumber)
```

Optional

Declaring RegistrationNumber as foreign key:

```
RegistrationNumber CHAR(7) FOREIGN KEY REFERENCES
Vehicle(RegistrationNumber)
/ /
RegistrationNumber CHAR(7)
FOREIGN KEY (RegistrationNumber) REFERENCES
Vehicle(RegistrationNumber)
```

Declaring three other fields:

```
DateStarted DATE
PolicyType VARCHAR(13)
ExcessAmount SMALLMONEY
```

1 mark for PolicyNumber with sensible type and length (if required), and identified as primary key. Type can be either numeric or text.

1 mark for two other fields from RegistrationNumber, DateStarted, PolicyType, ExcessAmount with sensible data types and lengths (if required by the type) *OR* **2 marks** for all four other fields with sensible data types and lengths (if required by the type)

- Length of RegistrationNumber, if specified, must be 7.
- Length of PolicyType, if specified, must be at least 13.

1 mark for identifying RegistrationNumber as a foreign key.

MAX 3

Valid alternative SQL types are:

- Alternative types For *PolicyNumber*: smallint, mediumint, integer, any text field type (see below)
- Alternative types For *DateStarted*: smalldatetime, datetime, datetime2, datetimeoffset
- Alternative types For *PolicyType*: ENUM('Comprehensive', 'Third Party') - accept any type of quotation marks around values - accept data values in any order - accept if ENUM defined as a type separately first
- Alternative types for *ExcessAmount*: money, currency, float, real, decimal, double, numeric, int, smallint, mediumint, integer
- Alternative types for *text fields*: char, varchar, nchar, nvarchar, text, ntext, longvarchar, varchar2, nvarchar2, text, tinytext, mediumtext, longtext

Sensible non-SQL data types can also be credited but MAX 2 if any non-SQL types used.

3

(b)
```
UPDATE Vehicle
SET Colour = "pink"
WHERE RegistrationNumber = "DF24JUT"
```

1 mark per correct line
A double or single quotes around pink and DF24JUT
A table names before fieldnames
A pink written in any case
DPT no quotes
DPT for fieldname before table name
DPT for unnecessary punctuation – allow one semicolon at the very end of the statement, but not at the end of each clause
MAX 2

2

(c)
```
SELECT Model, Colour, Forename, Surname
FROM Owner, Vehicle
WHERE RegistrationNumber = "AB72XHC"
AND Owner.OwnerID = Vehicle.OwnerID
```

1 mark for correct four fields in SELECT clause

1 mark for correct two tables in FROM clause
1 mark for WHERE RegistrationNumber = "AB72XHC"
1 mark for Owner.OwnerID = Vehicle.OwnerID, joined to other condition with AND

--- OR ---

```
SELECT Model, Colour, Forename, Surname
FROM Owner INNER JOIN Vehicle ON Owner.OwnerID = Vehicle.OwnerID
WHERE RegistrationNumber = "AB72XHC"
```

1 mark for correct four fields in SELECT clause
1 mark for correct two tables in FROM clause
1 mark for INNER JOIN using Owner.OwnerID = Vehicle.OwnerID
1 mark for WHERE RegistrationNumber = "AB72XHC"

Marks for SELECT and FROM statements should not be awarded if additional fields / tables included.
Accept table names before fieldnames.
Accept use of Alias / AS command eg FROM Vehicle AS V then use of V as table name.
Accept insertion of spaces into fieldnames
DPT for unnecessary punctuation – allow one semicolon at the very end of the statement, but not at the end of each clause.
DPT for fieldname before table name.

Refer responses using nested SQL queries to team leaders.

4

(d) (i) Sequence of instructions / program / code;
NE programming language
Note: Do not award mark for program if candidate clearly means HTML
which is executed / run / interpreted on the server (instead of the client);
executed / run / interpreted when a web page is requested;
to generate a web page (and its contents) / result which the server returns to
the client / / generating of dynamic web pages;
MAX 2

2

(ii) **1 mark for this point:**
Retrieve RegistrationNumber / value input by user and store in variable;
R responses that suggest the command makes the user input the values at the point in time when the script is run

MAX 1 point from this list:
from the web page / web site / form / web server / browser / url / request;
using POST / GET methods;

2

(iii) Output the forename and surname;
Back to the web server / web browser / client / terminal;
A display forename and surname on web page (or alternative) for both marks
R responses that imply output is made directly to screen

2

(e) Create a new table / / suitable table name given eg SafetyCertificates;

with CertificateNumber as the primary key ;
Include these fields in new table: CertificateNumber, DateIssued, GarageName;
Add RegistrationNumber into the new table as a foreign key / / as link to Vehicle table;
A relation for table
A different fieldnames for new fields if meaning the same
A adding the extra field ExpiryDate, but not as an alternative to DateIssued
A answers by example eg writing out the new table definition, SQL script to achieve changes
R a composite key in new table
Do not award any marks unless it is clear that a new table has been created

3

[18]

Q8.

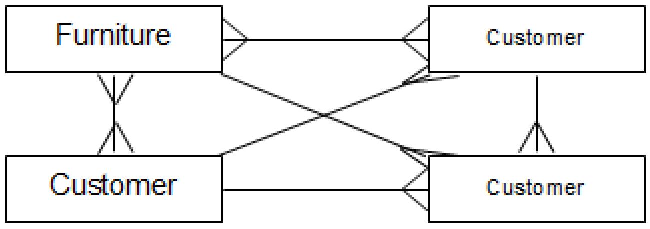
(a) **What means:**

every attribute (in relation) is dependent on the key;
the whole key;
and nothing but the key;
R Everything
OR
(relations) contain no repeating groups (of attributes) // data is atomic;
no partial dependencies;
no non-key dependencies;
R No repeated columns / attributes / data
OR
every determinant (in the relation) is a candidate key;;
Max 2

Why important:
Eliminate update anomalies; **A** Example
Eliminate insertion anomalies; **A** Example
Eliminate deletion anomalies; **A** Example
Eliminate data inconsistency // improve consistency // avoid inconsistency problems;
*Minimise data duplication // no unnecessaryrepeated data; **A** Reduce for minimise **R** eliminate
*Eliminate data redundancy; **A** Reduce / minimise for eliminate
NE Easier to update / insert / delete without concrete example or good explanation
NE Less errors whenupdating / inserting / deleting without concrete example or good explanation
NE Saving space / memory
NE Easier to query
Award marks to points made anywhere across (a)
Can only award one of the two marks indicates by asterisks (*)
Max 2

4

(b) One mark per correct relationship.



Max 2 if any incorrect relationships drawn
Max 3

3

(c) FurnitureID INT PRIMARY KEY NOT NULL

//

FurnitureID INT
PRIMARY KEY(FurnitureID)

$\boxed{\text{Optional}}$ ↥

Note that currency is not a valid SQL type

FurnitureName VARCHAR(30)
Category VARCHAR(10)
Price SMALLMONEY
SupplierName VARCHAR(20)

Allow lengths after numeric types e.g. INT(11) as these are allowed in MySQL.

1 mark for FurnitureID, with sensible data type and identified as primary key

1 mark for two other fields with sensible data types and lengths *OR* **2 marks** for all four other fields with sensible data types and lengths

A any sensible types. Lengths do not need to be specified.

Valid alternative SQL types are:

- Alternative types For *FurnitureID:* smallint, mediumint, integer, any text type (see below)
- Alternative types for *Price:* money, float, real, decimal, double, numeric, int, smallint, mediumint, integer
- Alternative types for *text fields:* char, varchar, nchar, nvarchar, ntext, longvarchar, varchar2, nvarchar2, text, tinytext, mediumtext, longtext

Answers using a syntax that is clearly not SQL should be awarded zero marks. But:

- **ignore punctuation errors e.g. unnecessary colons or commas.**
- **answers in SQL style syntax but using non-SQL data types can be credited but Max 1 of 2 for data types if any non-SQL types used.**

3

(d) SELECT CustomerName, TelephoneNumber
FROM Customer, CustomerOrder,
CustomerOrderline
WHERE FurnitureID=10765

```
AND
Customer.CustomerID= CustomerOrder.CustomerID
AND CustomerOrder.OrderID= CustomerOrderLine.OrderID
ORDER BY CustomerName (ASC)
```

1 mark for correct two fields in SELECT clause
1 mark for correct three tables in FROM clause
1 mark for FurnitureID = 10765
1 mark for Customer.CustomerID = CustomerOrder.CustomerID, joined to other conditions with AND
1 mark for CustomerOrder.OrderID = CustomerOrderLine.OrderID, joined to other conditions with AND
1 mark for ORDER BY CustomerName, ASC is optional

--- OR ---

```
SELECT CustomerName, TelephoneNumber
FROM Customer INNER JOIN CustomerOrder
ON
Customer.CustomerID=CustomerOrder.CustomerID INNER JOIN
CustomerOrderLine ON
CustomerOrder.OrderID=CustomerOrderLine.OrderID
WHERE FurnitureID = 10765
ORDER BY CustomerName (ASC)
```

1 mark for correct two fields in SELECT clause
1 mark for correct three tables in FROM clause
1 mark for INNER JOIN using Customer.CustomerID=CustomerOrder.CustomerID
1 mark for INNER JOIN using CustomerOrder.OrderID=CustomerOrderLine.OrderID
1 mark for FurnitureID = 10765
1 mark for ORDER BY CustomerName, ASC is optional

Marks for SELECT and FROM statements should not be awarded if additional fields / tables included.
Marks can be awarded for the conditions in the WHERE statement even if the required tables are not present in the FROM.
Accept FurnitureID with no quotation marks, single quotation marks or double quotation marks.
Accept table names before fieldnames.
Accept use of Alias / AS command e.g. FROM Customer AS C then use of C as table name.
Accept insertion of spaces into fieldnames
Ignore unnecessary clause CustomerOrderLine.FurnitureID=Furniture.FurntiureID
I unnecessary brackets
DPT for unnecessary punctuation – allow one semicolon at the very end of the statement, but not at the end of each clause.
DPT for fieldname before table name.

Refer responses using nested SQL queries to team leaders

6

(e) One mark for tick in correct row. Do not award mark if more than one row is ticked.

Command	Correct? (Tick One)
ALTER TABLE	✔
CREATE FIELD	
INSERT COLUMN	

1

[17]

Q9.

(a) Composite (key);
A Compound (key)

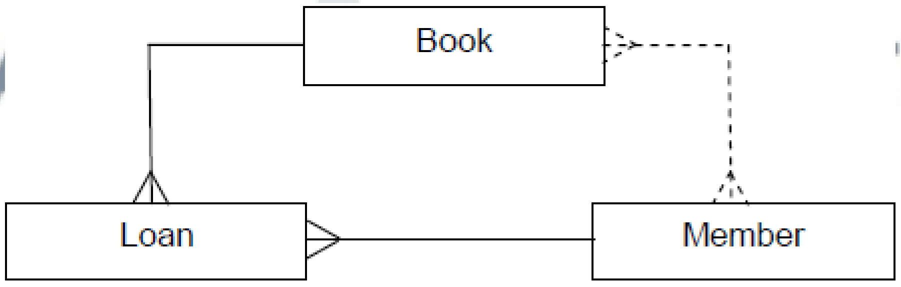
Note: The word key is not required

1

(b) Data is atomic // no repeating groups (of attributes);
R No repeated columns / attributes / data / values
No partial (key) dependencies // No (non-key) attribute depends on part of the primary key but not the whole of it // all non–prime attributes are (functionally) dependent on the whole of every candidate key // (non-key) attributes depend on the whole key;
No non-key dependencies // No transitive dependencies // (non-key) attributes depend on nothing but the key;
Every (non-key) attribute is dependent upon the key;
Every determinant is a candidate key;
A "field" for "attribute"
A "part" for "partial"

Max 2

(c)



1 mark per correct relationship (the dashed one is given)
Max 1 if more than two relationships drawn

2

(d) **Solution 1:**

```
SELECT EmailAddress, Forename, Surname
FROM Book, Member, Loan
WHERE Author = 'Lucas Bailey' AND
          Book.BookID=Loan.BookID AND
          Member.MemberID=Loan.MemberID
```

1 mark for correct three fields in SELECT clause
1 mark for correct three tables in FROM clause
1 mark for Author = 'Lucas Bailey'.

1 mark for Book.BookID=Loan.BookID linked by AND
1 mark for Member.MemberID=Loan.MemberID linked by AND

Solution 2:

```
SELECT EmailAddress, Forename, Surname
FROM Book INNER JOIN Loan ON Book.BookID=Loan.BookID
        INNER JOIN Member on
        Member.MemberID=Loan.MemberID
WHERE Author = ‘Lucas Bailey’
```

1 mark for correct three fields in SELECT clause
1 mark for correct three tables in FROM clause
1 mark for join from Member to Loan
1 mark for join from Loan to Book
1 mark for Author = ‘Lucas Bailey’
Note: Joins do not need to be done in same order as example
Do not award mark for SELECT clause if extra attributes listed.
Do not award mark for ‘Lucas Bailey’ unless it is enclosed in single or double quotation marks.

***A** table names before fieldnames.*
***A** use of Alias / AS command e.g. FROM Member as M then use of M as table name.*
***A** insertion of spaces into fieldnames*
***DPT** for unnecessary punctuation – allow one semicolon at the very end of the statement, but not at the end of each clause. Also, allow insertion of brackets at logically allowable places in the WHERE/FROM clauses.*
***DPT** for fieldname before table name.*

Refer responses using nested SQL queries to team leaders.

5

(e) **Alternative 1:**

```
INSERT INTO Book
VALUES ( 837023, “Kenyan Safari”, “Karen Matu”, “African Travel Guides” )
```

Alternative 2:

```
INSERT INTO Book (BookID, Title, Author, Publisher)
VALUES (837023, “Kenyan Safari”, “Karen Matu”, “African Travel Guides” )
```

1 mark for INSERT INTO Book;
1 mark for correct field values. If alternative 2 is used, the order of the values and fieldnames must correspond to each other;
The values Kenyan Safari, Karen Matu and African Travel Guides must be in single or double quotation marks for the mark to be awarded.

***A** the value 837023 with or without quotation marks.*
***A** Minor errors in transcribing the data from the question into the answer.*
***A** omission of brackets*

2

(f) *One mark for principle and max two marks for implementation.*

Principle:
Create a new table (**A** link table) (BookCopy); through which Book and Loan tables will be (indirectly) linked;

Implementation details using a new primary key:

Create a new <u>unique ID/key field</u> (e.g. CopyID) (for each copy);
Store the BookID and the CopyID in the new table;
Replace the BookID in the Loans table with this CopyID;

Note: In this implementation, CopyID is unique, i.e. BookID 1 and 2 cannot both have CopyID 1.

Implementation details using a composite key:
Create a new field CopyID;
Composite key formed by BookID and CopyID; **TO** if composite key is clearly in book table or loan table
Store the BookID and the CopyID in the new table;
R adding CopyID to Book table as this would created data redundancy but this does not talk out other mark scheme points
Add the CopyID field to Loans table;
R replace BookID with CopyID

Note: In this implementation, CopyID is not unique, e.g. BookID 1 and 2 can both have CopyID 1.
Marks can be awarded for principle and/or implementation details.

A Relation for Table
A Answers if candidates have rewritten new relations, awarding marks where the points above can be observed in the redrawn relations;
A alternative name for CopyID

Max 3

(g) (i) So that searching, adding and deleting can be done efficiently // To speed up searching, adding and deleting;
A just one of searching, adding, deleting
NE organise efficiently
NE easily for efficiently

1

(ii) **Alternative 1 (context-specific):**

A function/calculation that computes a record position/address; within a specified range; from a key field value;
A an example of a hashing function e.g. calculate an integer from certain letters in a field for one mark

Alternative 2 (generic):
A function (**A** algorithm) H, applied to a key k; which generates a hash value (H(k)) (of range smaller than the domain of values of k);

Max 2

(iii) **What is** *(1 mark)*:
When more than one key value maps to the same record position/address // when two keys compute the same hash value;
A “two records”, “two items” or “two pieces of data“ for “two keys” but **R** “two files” – both in this question part only

How dealt with *(1 mark):*
Store the record in the next available location in the file // store a pointer (in each file location) that points to a list of records that have all collided at the file location;
A idea that each storage location could store more than one record e.g. five records per location, if explained.
A example of what “next available” might be

A key is rehashed
A table for file

2

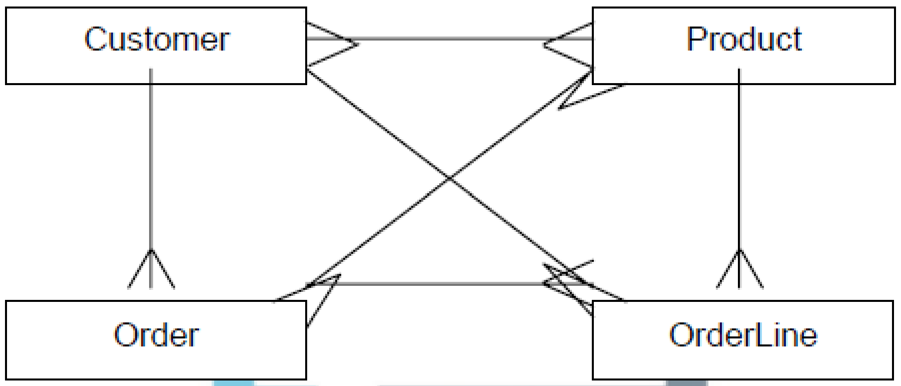
[20]

Q10.

(a) Only one (type of) product per order // Must make new order for each (type of)
product; as ProductNumber / product details stored in relation that has OrderNumber as primary key / product relation directly related to order relation // as relations not (fully) normalised;
Difficult to query // requires (unnecessarily) complex queries; as contains repeating groups (of attributes);
A either way round
A table for relation

Max 2

(b)



1 mark for each correct relationship, up to Max 3
Max 2 if more than three relationships drawn.

Max 3

(c) ProductNumber INTEGER PRIMARY KEY//

ProductNumber INTEGER
PRIMARY KEY(ProductNumber) }

ProductPrice SMALLMONEY
ProductDescription VARCHAR(50)
QuantityInStock INTEGER

1 mark for ProductNumber correct with appropriate type and identified as primary key
1 mark for two other fields correct with appropriate types OR 2 marks for all three other fields correct with appropriate types

A any sensible types / field lengths. eg:
For ProductNumber: integer, numeric, char, varchar, text, nchar, nvarchar, ntext, longvarchar, varchar2, nvarchar2
For ProductPrice: smallmoney, money, currency, float, real, decimal, dec, double, double precision, numeric
For ProductDescription: varchar, char, varchar, text, nchar, nvarchar, ntext,

longvarchar, varchar2, nvarchar2
For QuantityInStock: integer, numeric, float, real, decimal, dec, double, double precision, numeric

A insertion of other unnecessary but valid SQL commands e.g. AUTO INCREMENT, NOT NULL
I Spaces inserted into fieldnames e.g. Product Number

Max 2 if additional fields added

3

(d) Sequence of instructions / program / code;
NE programming language

Note: Do not award mark for program if candidate clearly means HTML

which is executed/run/interpreted on the server (instead of the client);
executed/run/interpreted when a web page is requested; [to generate a web page (and its contents) which the server returns to the client // generating of dynamic web pages;

Max 2

(e) (i) Max 1 point from this list:
Retrieve ProductNumber and Quantity // retrieve values input by user;
stores values in variables;
R responses that suggest these two commands are making the user input the values

Max 1 point from this list:
from the web page/web site/form/web serve/browser;
using POST/GET methods;

Max 2

(ii) Query/retrieve data from the products table;
to retrieve the <u>price of product</u> being ordered/selected on form/product that has correct product number/product number in ProdNum;
Store the set of records/data/price returned in ProdDetails;

Max 2

(iii) To send/output the Total Price back to the web server / web browser / client;
A display price on web page
R sent to user / customer

1

(f) **Either**

```
SELECT ProductNumber, ProductDescription, ProductPrice, Quantity
FROM Product, OrderLine
WHERE OrderNumber = 4013
      AND Product.ProductNumber = OrderLine.ProductNumber
ORDER BY ProductNumber ASC
```

1 mark for SELECT clause with correct four fields
1 mark for FROM clause with correct two tables
1 mark for OrderNumber = 4013
1 mark for clause linking tables on the common field with no additional unnecessary clauses added

1 mark for ORDER BY ProductNumber, ASC is optional

Or

SELECT ProductNumber, ProductDescription, ProductPrice, Quantity
FROM Product INNERJOIN OrderLine ON

Product.ProductNumber = OrderLine.ProductNumber
WHERE OrderNumber = 4013
ORDER BY ProductNumber ASC

1 mark for SELECT clause with correct four fields
1 mark for correct two tables in FROM clause
1 mark for INNERJOIN together with ON
Product.ProductNumber = OrderLine.ProductNumber and no other joins
1 mark for OrderNumber = 4013
1 mark for ORDER BY ProductNumber, ASC is optional

In both solutions:
Do not award mark for SELECT clause if extra attributes listed.
Do not award mark for FROM clause if extra tables listed.
Do not award mark for ORDER BY clause if order descending.
Only award two marks for conditions if they are connected by AND.
Otherwise just award one of the marks.
If candidate appears to have written two queries e.g. there are two SELECT commands then mark the first query.

A table names before fieldnames. i.e. TableName.FieldName
A " or ' as delimiters for 4013
A ascending, (ASC) for ASC
R if ASC written before ProductNumber in ORDER BY
I Spaces inserted into fieldnames e.g. Product Number
A answers that candidates have surrounded by "ExecuteSQL()".

If any of the errors listed below are made, they should result in at most one mark being lost. If the mistake is made more than once then on subsequent occasions, providing that the meaning is clear, the mistake should be ignored:

- the addition of unnecessary punctuation such as semicolons
- the fieldname being written before the tablename

5

[20]

Q11.

(a) (i) member ID / user name; password/PIN;
A account name instead of memberID;
A <u>answers to</u> security questions;

2

(ii) Member (MemberID, CreditCardNo, Member(Full)Name, Address, DrivingLicenceNo, EmailAddress, Mobile(Tel)No/TelNo); + attributes from b(i)

I bars over attributes

1

(iii) ParkingArea (<u>LocationCode</u>, ParkingAreaName, PostCode);
A ParkingAreaID instead of LocationCode

R ParkingArea
R Name as attributes

1

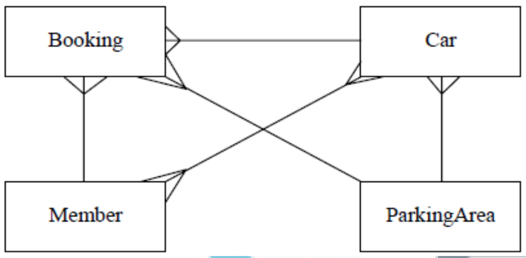
(iv) Car (<u>CarRegNo</u>, LocationCode);
A RegNo/CarReg instead of CarRegNo
Allow follow through on foreign key from (iii)

1

(v) Booking (<u>BookingRefCode</u>, CarRegNo, MemberID, StartDateTime, EndDateTime, LocationCode);;;
1 mark for CarRegNo and MemberID;
1 mark for StartDateTime and EndDateTime;
1 mark for LocationCode;
1 mark for BookingRefCode as primary key;
A 2 separate attributes for DateTime
A BookingRef/BookingID instead of BookingRefCode
Follow through on attribute names

Max 3

(b)



1 mark for each correct relationship,
If 4 or 5 relationships given, mark as follows:
All 4/4 or 5/5 correct: 3 marks
3/4 or /54 correct: 2 marks

2/4 or3/5 correct: 1 mark
All other cases: 0 marks
I *relationship between Member and Parking Area*

Max 3

(c) SELECT MemberID, (MemberFullName,) CarRegNo, StartDateTime, (EndDateTime) FROM (Member,) Booking

1

WHERE Member.MemberID = Booking.MemberID

1

AND EndDateTime BETWEEN 1/12/07 AND 31/12/07

1

ORDER BY MemberID (ASC/DESC)

1

A other attributes if present in candidate's booking table

1

Alternative Answer:
SELECT *; FROM Booking;
WHERE EndDateTime LIKE "*/12/07"

A StartDateTime instead of EndDateTime

P1 if attribute.table notaion used
P1 for extra punctuation or tbl in front of table name
I punctuation around dates/times
I case of keywords etc
A other wildcard characters

Alternative Answer:
SELECT MemberID, MemberFullName, CarRegNo,
StartDateTime, EndDateTime
FROM Member INNER JOIN Booking
ON Member.MemberID = Booking.MemberID
WHERE EndDateTime >= 1/12/07 AND
 EndDateTime <= 31/12/07
ORDER BY MemberID

Max 4

[15]

Q12.

(a) (i) Recipe table;
A Figure 2;

1

(ii) **Why:** contains multiple values in Ingredients field/attribute/column
// data in Ingredients column not atomic // repeating groups;

1

(b) (i) **Fully normalised:**
Every attribute is dependent on the key, the whole key and nothing but the key;;
OR (tables contain no repeating groups of attributes,) no partial dependencies;
No non-key dependencies;
A rely on instead of depend on
OR if (and only if) every determinant in the relation is a candidate key;

2

(ii) **Why:** to aid consistency of data // to avoid potential data inconsistency problems
// to eliminate data inconsistency // to minimise data duplication
// to eliminate data redundancy; A reduce instead of eliminate
R saving space

1

(c) (i) Recipe (<u>RecipeID</u>, Dish, PrepTime, CookTime, NoOfServings, CookInstructions);

1

(ii) FoodItem (<u>FoodItemID</u>, FoodItemName, PackSize, Price);

1

(iii) RecipeIngredient(<u>FoodItemID, RecipeID</u>, Quantity)

4

1 mark for each correct field, 1 mark for correct primary key
(take off 1 mark for every extra field included)

(d) SELECT FoodItemName, Quantity, PackSize, Price (1)
FROM FoodItem, RecipeIngredient, Recipe (1)

WHERE (Recipe.RecipeId = RecipeIngredient.RecipeId) (1)
AND (RecipeIngredient.FoodItemId = FoodItem.FoodItemId) (1)
AND (Recipe.Dish = "Feta Salad") (1)
ORDER BY FoodItemName ASC (1)
Field names **F/T**
P1 for fieldname.tablename
P1 tbl prefix
A ORDER BY FoodItemName
A Dish instead of Recipe.Dish
A 'feta salad' instead of 'Feta Salad' A #feta salad# instead of 'Feta Salad'

Max 5

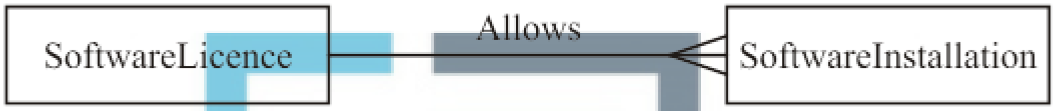
[17]

Q13.

(a) <u>Copyright</u>, Designs and Patents Act (1998); *if other laws included* **T.O.**

1

(b) Boxes for correct entities: SoftwareLicence SoftwareInstallation *one mark*
Correct degree of relationship: 1 to many one mark
Suitable name for relationship: one mark



3

(c) Any sensible field length accepted except for SoftwareID, ComputerID, StaffID

(i) SoftwareID VARCHAR(<u>10</u>) PRIMARY KEY (NOT NULL)

// SoftwareID VARCHAR(<u>10</u>)

PRIMARY KEY(SoftwareID);
Could appear at end of list. It doesn.t have to be with softwareID VARCHAR(10) In fact, this would provide a syntax error

SoftwareName VARCHAR(30)

Supplier VARCHAR(20)

DatePurchased DATE

Version VARCHAR(10)

ExpiryDate/DateValidTo DATE

NoOfLicences INT

1 mark for any 3 attributes correct
***P1** if extra symbols used*
Ignore spaces and case in attribute

3

(ii) SoftwareID VARCHAR(<u>10</u>)
A char/string/text/alphanumeric instead of VARCHAR
A Date/Time instead of Date
A Integer instead of INT

BOD any attributes which are clearly more than 1 word

ComputerID VARCHAR(6)

DateInstalled DATE

StaffID VARCHAR(3)

PRIMARY KEY (SoftwareID, ComputerID);

FOREIGN KEY (SoftwareID)

REFERENCES Software Licence(SoftwareID);

1 mark for any 2 attributes correct
If not DDL give 1 mark If composite key identified
I NOT NULL

4

(d) SELECT ComputerID, SoftwareName, Version ;
Extra attributes: T.O.

FROM SoftwareLicence, SoftwareInstallation ;

WHERE SoftwareLicence.SoftwareID=SoftwareInstallation.SoftwareID ;

ORDER BY ComputerID;

A ASC or DESC
Accept (instead of FROM WHERE): FROM SoftwareLicence INNER JOIN SoftwareInstallation ON SoftwareLicence.SoftwareID = SoftwareInstallation.SoftwareID
P1 for other spurious punctuation inc semicolons
A LEFT JOIN
Table names prefixed with tbl, ***P1***
If table name and attribute transposed, ***P1***

4

[15]

Q14.

(a) (i) A HardwareItem
B EquipmentLoan
C 'is out on';
A any wording with similar meaning
R one to many relationship

1 mark for A and B, 2 marks for C

2

(ii) Entity-Relationship Diagram;
A E-R diagram
A E-R D
R E-A-R diagram

1

(b) CREATE TABLE Hardware Item

(Description VARCHAR (30)
Make VARCHAR(15)
Model VARCHAR(15)
A text/string instead of char/varchar
1 mark

(Inventory)RefNo CHAR(20) PRIMARY KEY,
A string/text/character/VARCHAR (20) instead of CHAR(20)
1 mark

PurchaseDate DATE,
A DateOfPurchase DATE
A Date/Time instead of Date
PurchasePrice CURRENCY,
Location VARCHAR(4))
A DECIMAL/MONEY/Number/Real/Float/Single instead of CURRENCY
A Room VARCHAR(4)
A INT/number instead of VARCHAR
1 mark

Alternative for InventoryRefNo:

(Inventory)RefNo CHAR(20), PRIMARY KEY(InventoryRefNo),
(Inventory)RefNo CHAR(20), NOT NULL,, PRIMARY KEY(InventoryRefNo),
A VARCHAR(20) instead of CHAR(20)
Note: string lengths do not have to be exact/present except for InventoryRefNo

CREATE TABLE EquipmentLoan
(Inventory)RefNo VARCHAR(20),
A NOT NULL
If not DDL but composite key identified, give 1 mark
Location VARCHAR(4),
(Staff)Initials VARCHAR(3),
DateRemoved DATE,
DateReturned DATE,
A NOT NULL
1 mark
PRIMARY KEY (InventoryRefNo, DateRemoved)
1 mark
FOREIGN KEY (InventoryRefNo) REFERENCES HardwareItem (InventoryRefNo))
1 mark
***P1** for extra attributes*

6

(c) SELECT (HardwareItem.)Description, (EquipmentLoan.)DateRemoved,
1 mark

EquipmentLoan. (Inventory)RefNo,
A HardwareItem.InventroryRefNo
1 mark

FROM HardwareItem, EquipmentLoan
1 mark

WHERE HardwareItem. (Inventory)RefNo = EquipmentLoan.(Inventory)RefNo
1 mark

AND (EquipmentLoan.)DateRemoved > givenDate
A > =
A = >
1 mark

ORDER BY (EquipmentLoan.) (Inventory)RefNo;
1 mark

or

SELECT (HardwareItem.)Description, (EquipmentLoan.)DateRemoved,
1 mark

EquipmentLoan. (Inventory)RefNo
A HardwareItem.InventroryRefNo
1 mark

FROM HardwareItem
INNER JOIN EquipmentLoan
Note: can swap tables
1 mark

ON HardwareItem. (Inventory)RefNo = EquipmentLoan. (Inventory)RefNo
1 mark

WHERE (EquipmentLoan.)DateRemoved > givenDate
A > =
A = >
1 mark

ORDER BY (EquipmentLoan.)(Inventory)RefNo;
A HardwareItem.InventroryRefNo
1 mark

F/T with attribute names
P1 for tbl prefix
P1 if table name after attribute name
I extra punctuation

6

[15]

Q15.

(a) CandidateNumber;

1

(b) Table contains repeating groups;
R repeated data/fields/attributes
ModuleCode, ExamSession, ModuleMark, Level, TotalMark, Grade contain multiple values; *mention at least one attribute by name (forename/ surname T.O.)*

There is redundant data ***T.O.***

Max 1

(c) 1 mark for correct primary key, 1 mark for correct other attributes,
I spaces/underscores in attribute names

Extra attributes = ***T.O.***

(i) Pupil (PupilForenames, Pupil Surname, <u>CandidateNumber</u>);
A (Forename,Surname,CandidateNo)

2

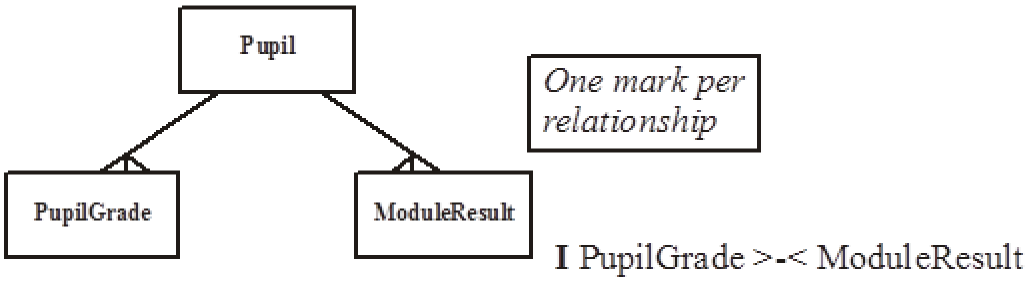
(ii) ModuleResult (<u>CandidateNumber, ModuleCode, ExamSession</u>, ModuleMark)

2

(iii) PupilGrade (<u>CandidateNumber, Level</u>, TotalMark, Grade)

2

(d)



2

(e) Must use same attributes as in (c) above (mark as F. T.) **I** case
SELECT PupilForenames,Pupil Surname, Grade *I pupil. I pupilgrade.*(1)
FROM Pupil, PupilGrade(1)
WHERE Pupil.CandidateNumber = PupilGrade.CandidateNumber
<u>AND</u> Level=“ A”
accept Level=‘A’ *or* Level=A(1)
ORDER BY TotalMark DESC; **A** Descending (1)

OR
SELECT PupilForenames,PupilSurname, Grade(1)
FROM Pupil INNER JOIN PupilGrade ON Pupil.CandidateNumber = PupilGrade. CandidateNumber(2)
WHERE Level = “A”
accept Level = ‘A’ *or* Level=A(1)
ORDER BY TotalMark DESC; **R** = Desc(1)

If pupilForename.pupil penalise once

5

[15]

Q16.

(a) Contains a <u>repeating group</u>;
OR
Cells for one or more of SubjectID, SubjectName, ExamBoardSubjectOfficerName, NumberOfCandidatesEntered contain multiple values;
R Repeating attributes, etc

1

(b) Attribute names must not be redefined (exception: allow Center).
1 mark for attributes(lose this mark if extra attributes), one mark for correct primary key

(i)

1

Centre(<u>CentreNo</u>, CentreName, CentreAddress)

1

2

(ii)

1

CentreEntryNumber(<u>CentreNo, SubjectID</u>, NumberOfCandidatesEntered)

1

2

(iii)

1

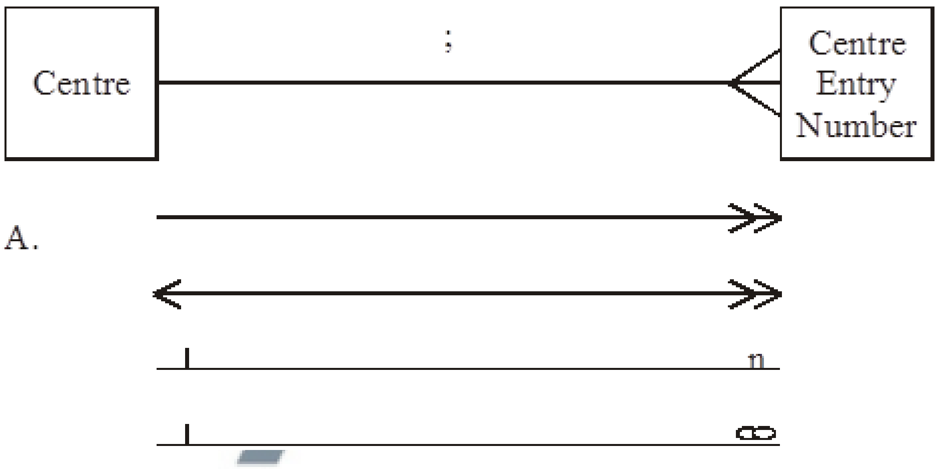
Subject(<u>SubjectID</u>, SubjectName, ExamBoardSubjectOfficerName)

1

2

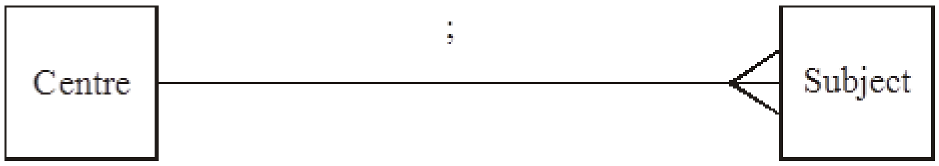
Penalise misspelling once
***I** spaces in attribute names*
***I** capitalisation*

(c) (i)



1

(ii)



1

(d) **I**. Inner join, Join
A Without commas
R Brackets in Select and anywhere else except:
A Brackets around (DateReported < 1/3/2005) and (ExaminationOfficer.CentreNo=Problem.CentreNo) as shown
Asc is optional but if present it must be at end of **Order By** line
(**A** Ascending)

Penalise brackets once

1
Select ExamOfficerSurname, CentreNo, ProblemDescription — Penalise **And** in this line

1
From ExaminationOfficer, Problem — R. If And used unless penalised above

1
Where DateReported < 1/3/2005 — I. Quotes/hashes/Absence of separators

A. Date in any representation

1
And ExaminationOfficer.CentreNo = Problem.CentreNo

A. IsLessThan or LessThan for <

1
Order By CentreNo Asc — Asc is optional

A. If candidate uses relation name in front of attributes, e.g. Select ExaminationsOfficer.ExamOfficerSurname

Penalise incorrect use of relation name once

Or

1
Select ExamOfficerSurname, CentreNo, ProblemDescription

1
From ExaminationOfficer,Problem — R. If And used

Where ExaminationOfficer.CentreNo = Problem.CentreNo

1
And DateReported < 1/3/2005

1
Order By CentreNo Asc

Or

1
Select ExamOfficerSurname, CentreNo, ProblemDescription

1
From ExaminationOfficer Inner Join Problem

1
On (ExaminationOfficer.CentreNo = Problem.CentreNo)

1
Where DateReported <1/3/2005

1
Order By CentreNo Asc

Asc is optional

Max 5

(e) **R** Brand names word processor//word processor with e-mail support;

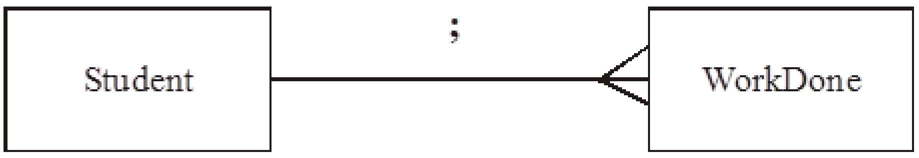
1

[15]

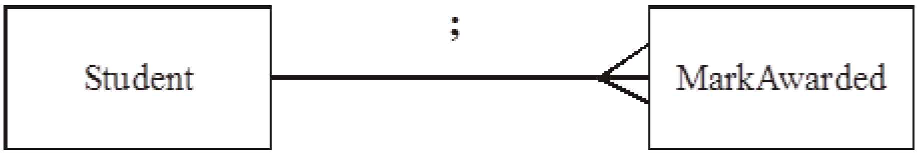
Q17.

(a) **I** Minor spelling

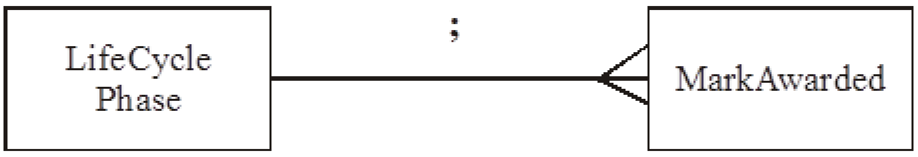
(i)



1

(ii) 

1

(iii) 

1

(b) Penalise table name. field name in reverse order once
R Quotes and additional constructs
I Table names unless in wrong order or wrongly expressed

(i) Select FirstName, Surname
From Student;

1

(ii) Select Student.FirstName, Student.Surname,
MarkAwarded.Mark;
I table names unless incorrect

From Student, MarkAwarded;
Where MarkAwarded.LifeCyclePhaseID = 1;
And Student.StudentID = MarkAwarded.StudentID ;
I table names unless incorrect

Order By Student.Surname;

I table names unless incorrect
Order By Student Surname Asc
A Ascending
Asc/Asending must be in correct position
A OrderBy

5

[9]

Q18.

(i) Name and TotalOfFines; (Accept slight mis-spelling/spaces)

1

(ii) NB Borrowers with FinesOwed > 0 required surnames and fines owed of borrowers who owe fines;

1

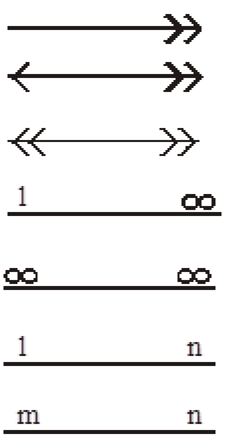
[2]

Q19.

(a) NB Take note of labelling inside boxes because candidate's positioning of labels may be opposite to that shown below

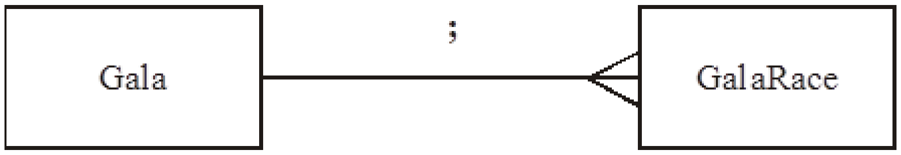
Alternative symbols

one to many
one to many
many to many
one to many
many to many
one to many
many to many



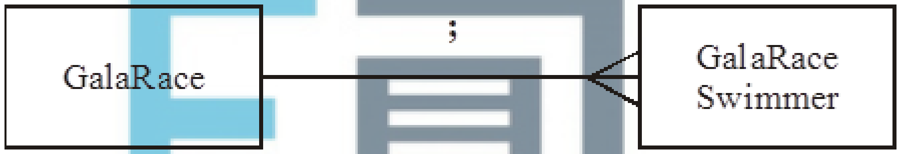
Accept minor mis-spelling or spaces between parts of entity name.
A. plural names
I. Box outlines

(i)



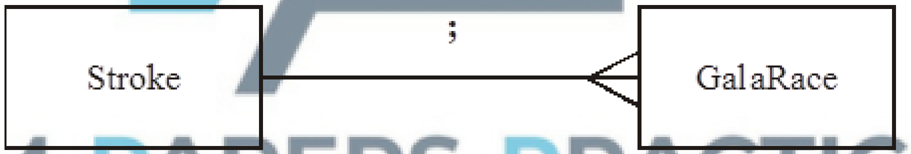
1

(ii)



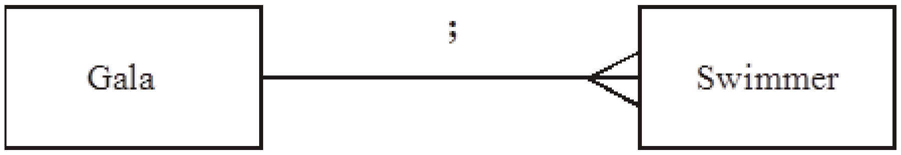
1

(iii)



1

(iv)



1

(b) **R** Tbl in front of table name - penalise once

(i)
```
Select Surname
   From Swimmer
   Where SwimmerN0 = 6;
```

Select Swimmer.Surname is OK

I Brackets surrounding attributes
R Extra attributes, tables, criteria
I Quotes around value 6
I ;

1

(ii)
```
Select SwimmerNo
  From GalaRaceSwimme
  Where (RaceN0 = 5); And (GalaNo = 2);
  Order By TimeRecordedFOrRace;
```

I Brackets surrounding attributes, table names in from of attributes unless correct
A Criteria without brackets
R Extra attributes, tables, criteria in this solution but be careful because candidate may give an alternative involving extra attributes, tables criteria that will work
I Quotes around value 5 and value 2
A Asc or Ascending in correct place i.e. after TimeRecordedForRace
R Asc/Ascending in any other position and/or with other words

3

(iii)
```
Select Swimmer Surname
  From Swimmer, GalaRace;
  Where (GalaNo = 4);
    And (GalaRace.SwimmerNoOfWinner
              =Swimmer.SwimmerNo);
```

OR

```
Select Surname
  From Swimmer
  Where SwimmerNo In; (Select SwimmerNoOfWinner
                         From GalaRace;
                         Where GalaNo = 4);
```

R = in place of In
I Brackets
A Select Swimmer.Surname, GalaRace.RaceNo From..
OR
A Select Surname, RaceNo
From..
Brackets may be omitted.
A GalaRace.GalaNo = 4

AndSwimmerNoOfWinner = SwimmerNo is OK

3

[11]

Q20.

(a) Contains a <u>repeating group</u>;
OR
One or more of RouteId, RouteName, RouteArea, RouteDescription contain multiple values;

1

(b) Attribute names must not be redefined

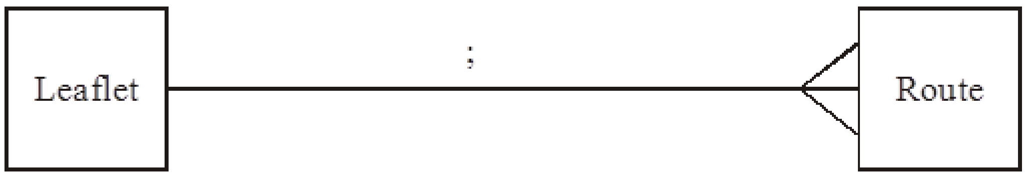
(i) Leaflet

2

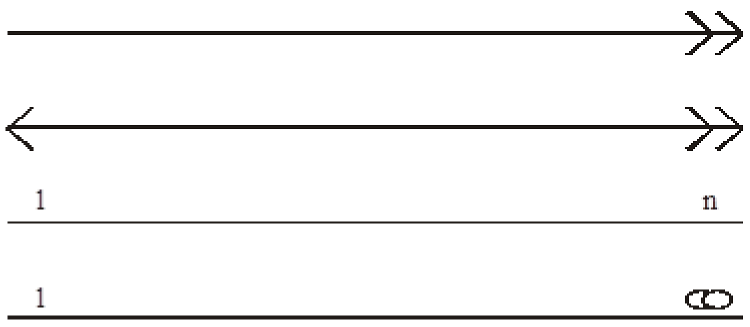
(ii) Route

3

(c)



A.



1

(d) **I.** Inner join, Join Asc is optional but if present it must be at end of **Order By** line (**A.** Ascending)
Penalise **And** once in this line
R. If And used
I. Quotes/hashes/Absence of separators
OR
Candidate may use relation name in front of attributes, eg. Select Person.Surname
Asc is optional
to max

6

(e) **R**. Brand names

(i) Word processor//word processor with e-mail support;

1

(ii) Desktop publishing; A. <u>publishing</u> package or anything with publishing

1

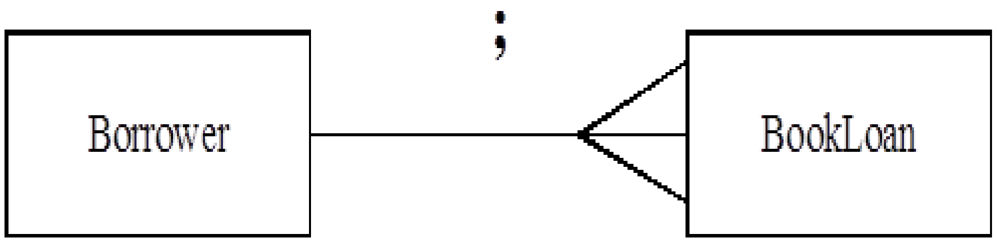
(iii) Spreadsheet;
R. Database & spreadsheet
R. Finance package.
R. Accounting package

1

[16]

Q21.

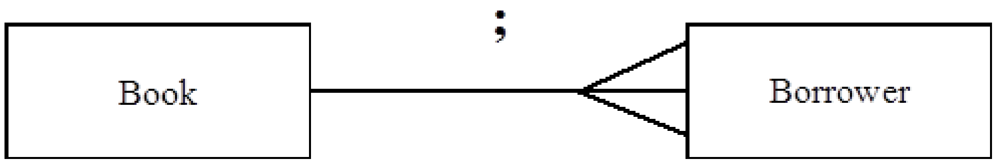
(a) (i)



A Alternative notations for degree of relationship

1

(ii)



I. other entities

1

(b) (i) Select Book.Title
A Title
From Book;
Where Book.ISBN = “1-57820-082-2”; **A** ISBN = “1-57820-082-2”;
Any extra attributes lose mark where extra attributes used
R 1-57820-082-2 Need quotes
A ‘1-57820-082-2’
R TblBook – penalise once
R Title.Book, wrong order

2

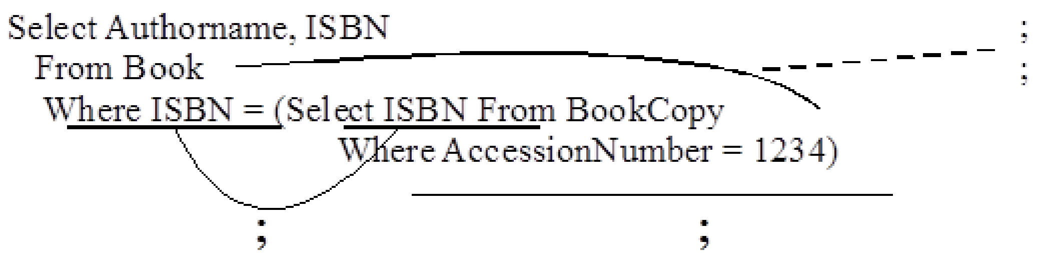
(ii)

```
Don’t need Book in Select
Select Book.AuthorName, Book.ISBN;
```

A BookCopy.ISBN in place of Book.ISBN

```
  From Book, BookCopy;
  Where (Book.ISBN = BookCopy.ISBN);
    And (BookCopy.AccessionNumber = 1234);
```

A AccessionNumber in place of BookCopy.AccessionNumber
R quotes on 1234
Any extra attributes lose mark where extra attributes used
Brackets non-essential. May see conditions interchanged, this is Ok
A in for =



```
Select Authorname, ISBN
  From Book
   Where ISBN = (Select ISBN From BookCopy
                   Where AccessionNumber = 1234)
```

; ; ;

Penalise TblBook/TblBookCopy once

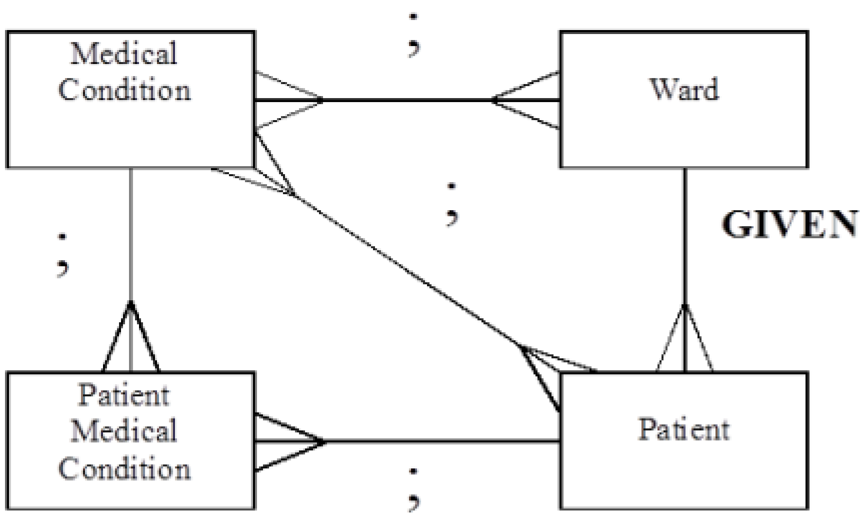
(c) Mail-merge//Mail-merging

1

[9]

Q22.

(a)



(NB don't allow relationship between Ward and PatientMedical Condition)

Max 3

(b) *For each extra attribute lose one mark*

(i) Ward(<u>WardName</u>, NurseInCharge, NoOfBeds) ;
A NumberOfBeds, NameOfNurseInCharge, NurseInChargeName
R WardId, Name, NurseName. NameOfNurse, BedNo, BedNumber, NumOfBeds

1

(ii) ; ;

Patient(<u>PatientNo</u>, Surname, Forename, Address, DOB, Gender, *WardName*)
A PatientId, PatientNumber, PatientSurname, PatientForename, PatientAddress, DateOfBirth, PatientDateOfBirth, PatientGender, Sex, PatientSex

2

(iii) MedicalCondition(<u>MedicalConditionNo</u>, Name,RecommendedStandardTreatment)
A MedicalConditionId, MedicalConditionNumber, MedicalConditionName, ;
ConditionName, StandardTreatment, Treatment, RecommendedTreatment
R ConditionNumber, ConditionID

1

(iv) ; ;
PatientMedicalCondition(<u>*PatientNo*</u>. <u>*MedicalConditionNo*</u>)
A Attributes rejected in (ii) and (iii) for PatientNo and MedicalCondition No R If attributes used are not consistent with (ii) and (iii)

2

(c) **Accept tbl in front of table name;**

```
 Select Patient.Forename, Patient.Surname,
PatientMedicalCondition.MedicalConditionNo
     From Patient, PatientMedicalCondition                    ;
        Where Patient.WardName ='Victoria'    ;
            And Patient.PatientNo = PatientMedicalCondition.PatientNo     ;
```

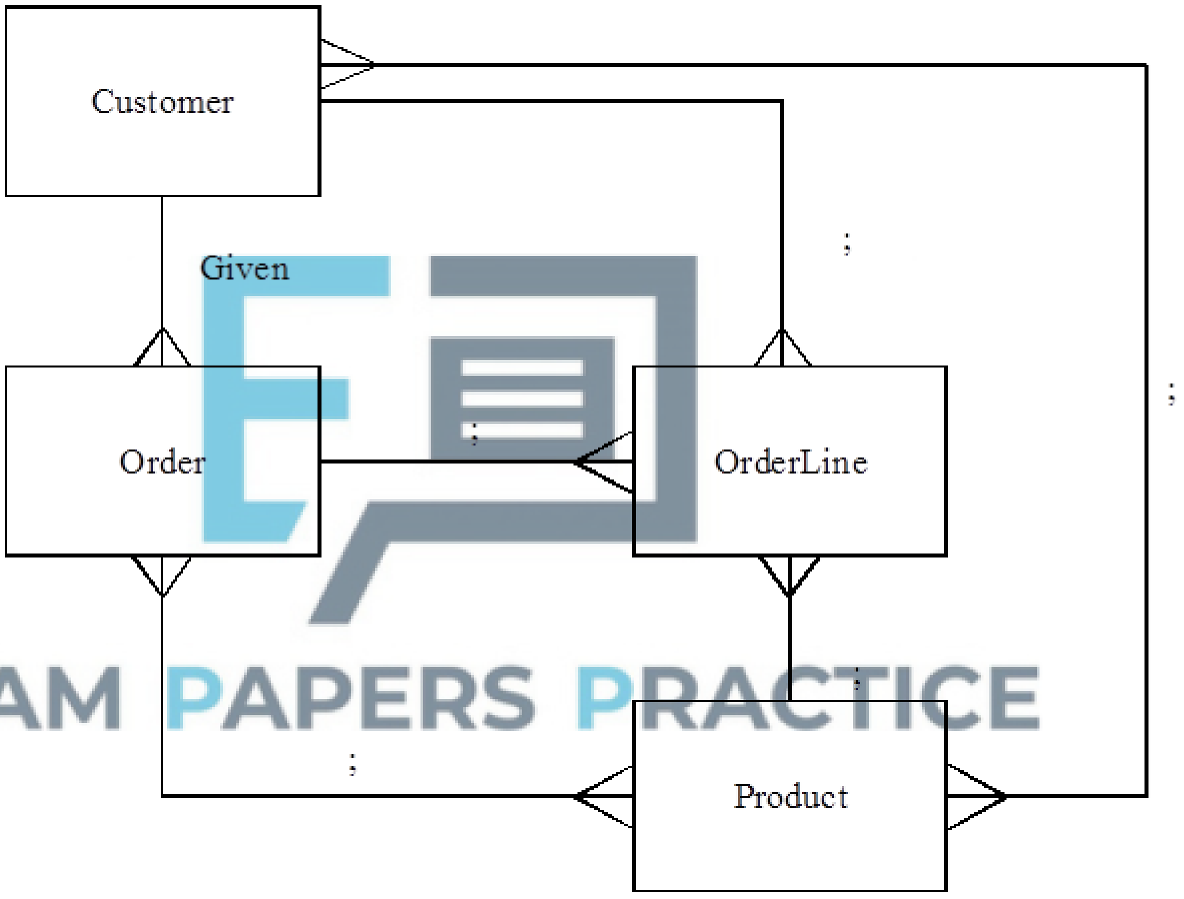
A Forename, Surname, MedicalConditionNo, WardName

Max 3

[12]

Q23.

(a)



If candidates gives more than required mark all to a max of three

3

(b) NB Order of attributes is immaterial

(i) *1 mark for primary key 1 mark for other attributes unless additional attributes*
A Quantity, StockLevel, Stock, AmountInStock, Amount, NumberInStock, Qty_Stock, Qty
A Description
Product (<u>ProductId</u>, ProductDescription, QuantityInStock)

2

(ii) *1 mark for primary key 1 mark for other attributes*

(unless extra other attributes)
A Name, Address, (Customer)TelephoneNo, (Customer) TelephoneNumber, TelNo, (Customer)TelNumber,
R Phone, Telephone, **A** Multiple lines for address including postcode, etc,
A Breakdown of name into forename, surname.
Customer (<u>CustomerId</u>, CustomerName, CustomerAddress, CustomerTelNo);

2

(iii) *1 mark for primary key 1 mark for foreign key CustomerId 1 mark for other attributes unless extra other attributes*

Order (<u>ABCOrderNo</u>, CustomerId, CustomerOrderNo, OrderHasBeenDespatched)
A Despatched, OrderDespatched, OrderStatus

3

(iv) *1 mark for primary key 1 mark for ABCOrderNo, LineNo,*
1 mark for foreign key ProductId, 1 mark for QuantityOrdered, unless additional attributes
A OrderLineNo, OrderLineNumber LineNumber, Line
A Quantity, Amount, Qty, Number
OrderLine (ABCOrderNo,LineNo, ProductId, QuantityOrdered)

4

(c) Attributes must correspond with table attributes given in part (b)
A Yes, ‘Yes’, “Yes”, Y, ‘Y’, “Y”,
A Despatched, ‘Despatched’, “Despatched” in place of True.(1)

Select CustomerName
(Score zero for extra attributes)
1 mark

From Order, Customer
(Each extra table cancels one of these marks)
2 marks

Where Customer.CustomerId = Order.CustomerId 1
And Order.OrderHasBeenDespatched = True
1 mark

Order By ABCOrderNo
1 mark

Mark first Select statement, but give credit for Order By ABCOrderNo.
Associate And with one of the conditions.
OK for parts of statement to be on same line, e.g. Select CustomerName From Order, Customer
A Customer.CustomerName
A Order By ABCOrderNo ASC
A Order By Order.ABCOrderNo
A OrderBy for Order By
A the use of aliases, e.g. Select D1.CustomerName From Customer D1, Order D2
Etc.
A “Customer.Db” Or Customer.Db , “Order.Db” Or Order.Db
A OrderHasBeenDespatched = True

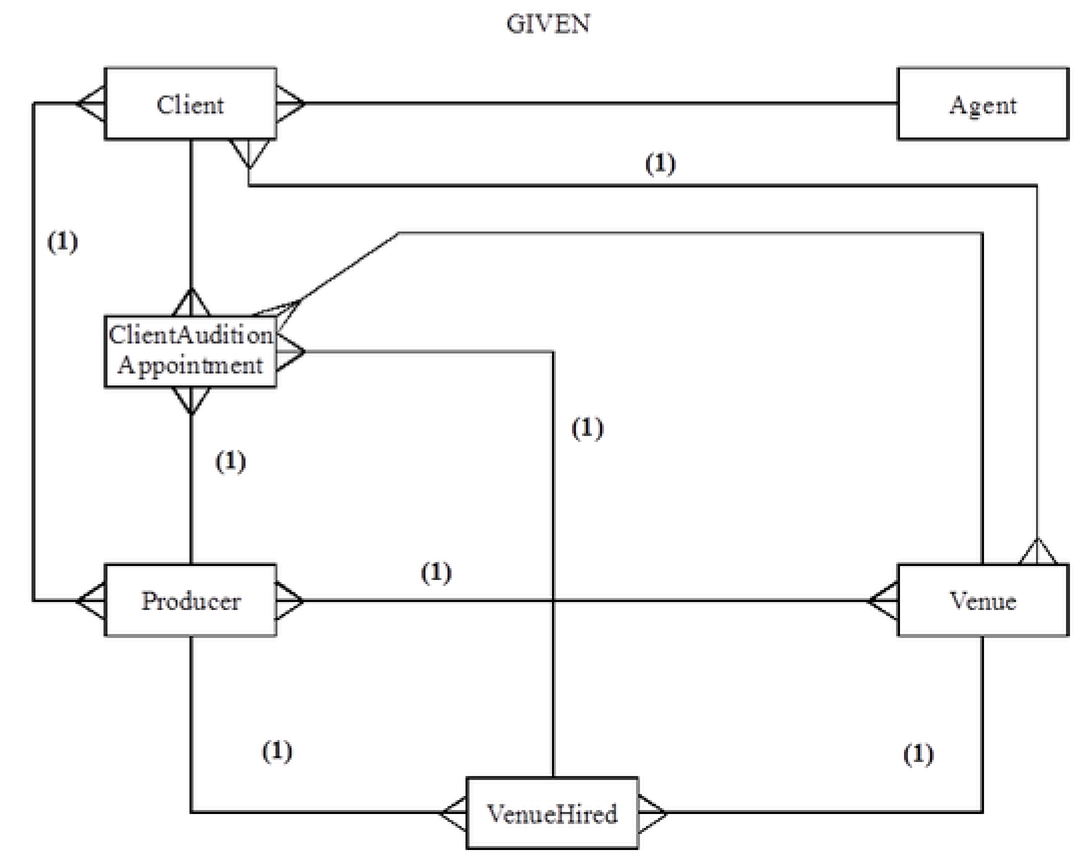
A Order By ABCOrderNo Ascending
A From tbl Customer, tbl Order

6

[20]

Q24.

(a)



No marks for Id on its own.
No marks for Name, Address on their own. But abbreviations for Producer, etc which are clear are allowed

Max 4

(b) (i) **Producer** (<u>ProducerID</u>, ProducerName)
No extra attributes

1

(ii) **Venue** (<u>VenueId</u>, VenueName, VenueAddress/Address)
No extra attributes

1 1

(iii) **Client** (<u>ClientId</u>, ClientName, AgentId)

Max 1 if extra attributes

2

1 1

(iv) **VenueHired**(<u>ProducerId,ApptDate</u>, VenueId)

1 for correct primary key

3

Or

__________1______ 1
VenueHired(<u>VenueId,ApptDate</u>, ProducerId)

1 for correct primary key
For each extra attribute lose mark from allocation except primary key mark
Accept Date and Time

Max 1

______________1_____________
(v) **ClientAuditionAppointment** (<u>ClientId,ApptDate</u>, ApptTime, ProducerId) and/or VenueID

1 for correct primary key
Accept Date and Time

3

(c) **Select** *ClientId, ClientName, Client/Agent.AgentId, AgentName*
1 1
From *Client, Agent*
Where *Client.AgentId = Agent.AgentId*
1 1

Mark only first Select... if more than one
1 for 4 correct attributes
1 for correct table name in front of AgentId

6

[20]

Q1.

A very good range of responses was received to this question, with approximately half of students achieving five or more marks. Most students addressed all three aspects of the question (hardware, network, database and software). Students tended to make more points about how the hardware could be improved than about the other two areas. This was acceptable but students needed to have covered all three areas to achieve a mark of ten or above.

Some students wrote too vaguely to achieve marks, for example by writing that a “faster processor” would improve performance, without referencing a factor such as the clock speed that would make the processor faster. Other mistakes included believing that the question required students to contrast thin-client and thick-client and that the system was web based.

A small number of students wrote about issues which might be causing the system to perform poorly instead of explaining how the performance of the system could be improved. Such responses were not worthy of a mark.

Q2.

(a) Over three quarters of students achieved some marks for completing the E-R diagram, with approximately a third achieving some marks. A straightforward way to determine the nature of the relationships is to identify which attributes are used to link the relations and in which relation such an attribute is the entity identifier/primary key and in which it is a foreign key.

(b) Good responses to this question part recognised that the data types and fieldnames were in the wrong order and that the primary key had no data type. Two thirds of students identified at least one error and just over a quarter identified two.

(c) Over half of students achieved some marks for this question. The most common valid responses referred to the elimination of data redundancy and data inconsistency.

(d) This question received a wide range of responses, with four fifths of students achieving some marks but only slightly over 10% achieving full marks. The most commonly achieved mark was for including the condition that identified records for the correct date. It was pleasing to see how many students were also able to identify the conditions required to join the tables. Common errors were failing to include all of the required tables in the FROM clause and missing out delimiters around the date.

Q3.

(a) Approximately a third of students correctly identified that a composite key made up of `CarRegNo` and `JobDate` could have been used. The most common incorrect response was `CarRegNo` on its own. This could not have been used as it would not have allowed a car to be booked into the garage for more than one job.

(b) Good responses recognised that a customer might own more than one car so by separating the owners into a new table data redundancy and the possibility of data inconsistency would be reduced. Where students have been given a scenario in the question it is important that they make reference to this when responding.

Responses such as "it would make the database more normalised" were not considered to be enough to be mark worthy as the student had not shown that they understood why the original design was not normalised nor that they appreciated the benefits of making it normalised.

(c) Three quarters of students gained some marks for completing the entity-relationship diagram. Some students used non-standard notations which were not considered creditworthy or drew more than three relationships on to the diagram. Students who draw more relationships than are asked for in this type of question face having their maximum mark limited.

(d) Whilst three quarters of students achieved a mark in this question part for correctly identifying the data that would be required to perform this update, only a third achieved marks for producing syntactically correct SQL. Common errors were to use `INSERT INTO` or `SELECT` instead of `UPDATE`, to specify the fieldname in the `UPDATE` clause instead of the table name and to miss out the condition necessary to identify which record to update.

(e) This was the least well-answered of the question parts that required students to write SQL code. More students believed that they should have used an `UPDATE` query than realised that they needed to use `INSERT`. Of those who did use `INSERT`, the most common errors were to miss out the keyword `INTO`, to miss out brackets where they were required or to put the numeric quantity value into quotation marks.

(f) This was the best-answered of the SQL question parts. The majority of students knew the correct structure of query to search for data although there was considerable variability in the accuracy of their solutions and just under a fifth of students gained full marks. Common syntactical errors were to add in brackets where they were not required, eg into the list of fieldnames or around `ASC`, to put commas or semi-colons between each clause or to use the keyword `AND` in the `SELECT` and `FROM` clauses in place of a comma. When a query requires that data from more than one table is used, students need to remember to include an appropriate condition to join the data from the two tables (either using an `ON` clause or additional `WHERE` conditions). A small minority of students attempted to write two distinct queries to retrieve the required data separately from each table.

(g) Just under half of students achieved some marks for this question. A common incorrect response was to add the `Make` and `Model` attributes to the `Part` relation. This was not an appropriate solution as it would have only allowed each part to be fitted to a single make/model of car, without introducing data duplication in the parts table. Students who identified the appropriate solution of creating a new relation containing the attributes `PartID`, `Make` and `Model` often achieved two marks rather than three as they did not identify the appropriate primary key for the table.

Q5.

This question was about relational databases. For part (a) candidates had to draw relationships onto an Entity-Relationship diagram. Most candidates achieved at least one mark, but only about a third achieved both. A surprising number drew only two relationships when the question had stated that three were required.

For part (b) candidates had to complete an update query. This was well answered. The most common errors were to include quotation marks where they were not required or to be confused about the syntax of the first line, for example by including the fieldname instead of just the table name after the keyword UPDATE

Part (c) required candidates to write a query to retrieve data from the database. The vast majority of candidates understood the basic structure of a query and were able to achieve some marks. Common mistakes were to include the Customer table in the FROM clause, which was not necessary, to miss out the part of the link between the Parcel and PriceBand tables that related to the parcel weight and to put the list into descending instead of ascending order. Some responses were seen that included (ASC) in brackets in the ORDER BY clause. The inclusion of brackets would prevent this from working; the brackets are included in mark schemes to indicate to examiners that the ASC is optional, they are not part of the language syntax.

Candidates needed to redesign part of the database to eliminate some redundancy that was identified for part (d). The majority of candidates recognised that a new relation was required to represent the postcodes, but the number who correctly designed the new relation and redesigned the Parcel relation was relatively small. The most common error was to include the house number in the new PostcodeLookup relation instead of leaving it in the Parcel relation.

Q6.

This question part (a) required that students completed an Entity-Relationship diagram. The vast majority of students were able to achieve one mark but less than half achieved both.

For question part (b) students had to write a DDL definition of the Treatment relation. This question part was very well answered. Students need to ensure that they use correct SQL syntax and data types if they are to achieve full marks. A small number of students wrote definitions for a different table.

For part (c) students had to write an SQL query. This question part achieved an excellent range of responses. The requirement to search for appointments that fell between two dates seemed to cause some confusion, with students not realising that they could simply use the > and < operators. Some students used other operations to extract the year part from the date, and whilst some of these were correct and markworthy many were not. This year there appeared to be fewer students who lacked any knowledge of how to form a query correctly in SQL.

Q7.

Overall, students demonstrated a satisfactory ability to use SQL in question parts (a) to (c).

(a) This part was the worst tackled of the three. Commonly made mistakes when defining the table in this part were to use incorrect SQL data types, to include the field EngineSize, to declare RegistrationNumber to be an integer or to use non-SQL syntax. Some students attempted to put a constraint on the PolicyType to ensure that it could only be one of the two valid possibilities. Students were not expected to know how to do this, so incorrect attempts at doing so were not penalised.

(b) In this part, just under half of students got full marks. Many students did not appear to really know the correct syntax of the SQL Update command, but achieved an easy mark by writing the first line of the command as “Update Vehicle”. A common mistake was to fail to put quotation marks around the registration number or colour values.

(c) Responses to this part were disappointing, given the frequency that students have been asked to write SQL queries in the exam over the years, and the fact that this was a fairly simple example, involving only two tables. Only one third of candidates achieved full marks, though half managed to achieve three of the four marks. As in part (b), one common mistake was to miss quotation marks around the registration number. Other mistakes were to use the keyword GET instead of SELECT and to miss out the AND operator in the WHERE clause. Pleasingly, most students realised that a condition was needed to link the two tables together.

(d) Parts (i) to (iii) were about server side scripts. Just under half of candidates were able to achieve full marks for part (i) by explaining that a server side script was program code that was executed on a server. Some also recognised that the trigger for this could be the requesting of a web page and that the output would be a web page. Part (ii) was the least well tackled part, with only around one third of students achieving any marks. The Request object is used to fetch user-inputted data from the web server that it will have been sent when the web page that the data was entered on was submitted. The majority of students believed that the Request object was used to either query a database or that the execution of the command would trigger a request to the user to input data at that time rather than retrieving already input data. Almost all of the candidates got one of the two marks for part (iii) but few clearly explained that the output would be written to a web page which the web server would return to the web browser on the Police Officer’s handheld terminal.

(e) This part was about extending the design of the database to store safety certificate information. The majority of candidates scored at least two of the three marks. Most candidates correctly identified that a new table would need to be created, what the fields in this table would need to be, and that CertificateNumber would be the primary key. Those candidates who scored two but not three marks usually made a mistake with regard to how the new table would be linked to the existing database, stating that the CertificateNumber would be added to the Vehicle table as a foreign key. This solution would not work as this would only allow one certificate to be associated with each vehicle, and it was required that previous certificates could also be stored. The correct solution was to put the RegistrationNumber from the Vehicle table into the new table as a foreign key.

Q8.

Part (a) was much better answered than when a similar question about the meaning and purpose of normalisation was asked in 2010. As in 2010, the meaning was better addressed than the purpose, with many candidates able to give a good definition. A few candidates failed to achieve marks by just writing “it relies on the key, whole key and nothing but the key" without making clear what “it" was.

The most commonly made creditworthy point in relation to the purpose of normalisation was that it would eliminate data redundancy. Candidates need to be aware that there is a difference between redundant data and duplicated data. Redundant data is data that is unnecessarily duplicated, for example, storing a person's address twice because they have purchased two items. Data duplication is simply storing data twice. Primary key values which are also foreign keys can be validly duplicated in a database without being redundant as they are required for the relationship. Therefore a goal of normalisation is to eliminate data redundancy but only to minimise data duplication.

Other good responses included eliminating the possibility of data inconsistency and preventing insert, update and delete anomalies.

Part (b) was moderately well answered with over two thirds of candidates achieving at least one mark, but only around a quarter achieving all three. Candidates who struggle to draw this type of diagram would benefit from looking at the relation definitions and establishing which fields are primary keys and which other relations these are used in as foreign keys.

Part (c) was well answered. In common with 2012, we only accepted valid SQL types and this year we also insisted on accurate SQL syntax, albeit allowing minor errors. The most common mistake was to not give a data type to the field that was the primary key.

For part (d) the SQL query was well tackled, but only around ten per cent of candidates achieved full marks. There is a discernible improvement in candidates' knowledge of SQL syntax. The most common mistakes were to either only include the Customer table in the FROM clause (presumably as both of the output fields were from this table) or to unnecessarily include the Furniture table in it.

In part (e) candidates were asked to correctly identify that the ALTER TABLE command was the appropriate one from the list of three that were given. The other two, incorrect, responses were both chosen more frequently than the correct response.

Q9.

Part (a): Just over half of students correctly identified that the key was a composite key. On this occasion, as the question did not refer directly to the example on the question paper, the answer compound key was also accepted – although it is important to note that these two terms are not equivalent.

Part (b): This question was not tackled well, but good students recognised that relations in a normalised database would have no repeating groups of attributes, no partial dependencies and no non-key dependencies.

Part (c): As in previous years, students had difficulty completing the entity-relationship diagram with many incorrectly identifying the degree of the relationships. Only a third of students achieved both marks for this question part.

Part (d): A pleasing number of students (approximately a quarter) achieved the full five marks on this question part, which required that a query was written in SQL. Common errors made by those who did not score full marks were: to forget to cross-reference the tables using either linking conditions in the WHERE clause or INNER JOIN in the FROM clause, to forget to enclose the string “Lucas Bailey” in quotation marks, and to miss out the AND keywords if they were required in the WHERE clause.

Part (e): This is the first time that students have been asked to use the INSERT INTO command in SQL. Most students scored at least one mark, for recognising that the first line of the command needed to be INSERT INTO BOOK. More mistakes were made in the second line, which should have been 837023, “Kenyan Safari”, “Karen Matu”, “African Travel Guides”. The most common error was to try to set the values by using assignment commands eg Author = “Karen Matu”. Another commonly made mistake was to leave out the quotation marks for the values which were clearly strings. Students were awarded credit regardless of whether the value 837023 was in quotation marks as the data type could have been either a string or numeric type.

Part (f): Students were required to redesign the database structure so that it would be able to store data about multiple copies of the same book whilst remaining normalised. Good responses recognised that a new relation would need to be created, which would contain the existing BookID and a new primary key Accession ID. The BookID in the Loans relation would then need to be replaced by the AccessionID. Alternative solutions that maintained normalisation were also accepted as were answers in which students rewrote new relations. The most common errors were to create a new primary key and store this in the Book relation, which would have resulted in redundant data (eg author details being repeated for each book), or to produce a solution which stored a quantity in stock field and updated this as books were loaned out. The latter would not have allowed the library to keep track of the individual copies of a book when loans were made.

Part (g)(i): This question part asked about hashing, which is used so that searching, adding and deleting records can be performed efficiently. Some students were able to explain the purpose of hashing adequately, but many confused hashing in this context with hashing for security purposes.

Part (g)(ii): This question part was moderately well answered, but most students did not include enough detail in their responses to achieve both of the available marks. In particular, whilst students recognised that the hash function would calculate a record position, many did not explain that the calculation would be based on the value in the record's key field.

Part (g)(iii): This question part was better answered than part (g)(ii). Many students recognised that a collision would occur when two key field values mapped to the same storage location and suggested an appropriate method, eg using the next available location or a linked list, to deal with the problem. A small number confused collisions in this context with collisions on a bus network.

Q10.

Part (a): Most candidates got at least one of the two marks for this question part. Good responses explained both the theoretical aspects of the problem with the order relation and also the real-world consequences of this. Some candidates mistakenly stated that the database contained partial key or non-key dependencies.

Part (b): Most candidates got some marks for this question, but full mark responses were quite rare. A small number of candidates ignored the instruction to draw three relationships and thus limited the maximum mark they were allowed to two out of three. The most common error was to show an incorrect degree for the relationship between the Product and Orderline relations.

Part (c): This question part was very well answered with most candidates getting some marks and many getting all three. The most common mistakes were to define ProductNumber as the primary key, but then to forget to give it a data type, and to declare Price to be an integer data type. When marking this question we allowed the use of data types that were taken from other languages rather than SQL, so long as they were clearly equivalent. In the future it is likely that we will require the use of correct SQL data types and syntax for full marks to be awarded to a response.

Part (d): A server-side script is a sequence of instructions that is executed on a web server to generate web pages dynamically at the time that a request to view a page is received. The majority of candidates showed a reasonable understanding of this and got at least one of the two available marks. Instead of explaining what a server-side script is, some candidates gave examples of usage or described them in such a way that they might have been templates rather than executable programs. A common statement that was insufficient to be creditworthy was that the scripts were stored on a server or accessed from the server. This was not adequate as the same could be said of static HTML pages.

Part (e)(i): This question part was well answered. To achieve both marks, candidates had to make clear that the values were being retrieved from the web server or from the input made in the web browser.

Part (e)(ii): This question part was well answered with many candidates getting both available marks.

Part (e)(iii): Only a minority of candidates achieved the mark for this question part. Most recognised that a calculation would be performed and the result of this would be output, but to achieve the mark a candidate needed to explain that these results would be displayed on the web page or sent back to the client computer and displayed in the web browser.

Part (f): This question part was well answered. Most candidates clearly understood the structure of an SQL query and many scored high marks. The most common mistakes were to include spurious punctuation such as semicolons or commas in responses and to include the Order relation in the FROM clause, which was not required. The correct command to sort the results was ORDER BY ProductNumber, or alternatively, ORDER BY ProductNumber ASC. Some candidates put brackets around the ASC which is not correct syntax.

Q11.

For part (a) Many candidates could list suitable additional details required to be stored about members (apart from those given in the question stem) such as MemberID or unique user name and a password or the answer to a security question. However, few candidates then used these in the Member table definition, only listing those explicitly stated in the question. Many candidates correctly defined the table *ParkingArea*. Common errors included using the same identifier for the table and for one of the attributes, or just *Name* instead of *ParkingAreaName*. The member's name is also stored in this database, and so more descriptive identifiers need to be used. The car table definition should only consist of the *CarRegistration* and the *LocationCode*. Many candidates did not seem to notice that the primary key of the *ParkingArea* table was required, rather than inventing another attribute such as *DesignatedParkingArea*. The booking table needs to store the details required for a booking. Essential are MemberID, *CarRegistrationNumber* and *StartDateTime*, *EndDateTime*. It should also store the *LocationCode* where the car is to be picked up. The question stated that a *BookingReferenceCode* is given to the member; this is unique and can therefore be used as a primary key. Many candidates lost marks by adding redundant attributes, such as *ParkingAreaName* or not using the correct identifiers for the foreign keys. Candidates should note that bars over attributes is not a standard way of showing foreign keys. Foreign keys were not asked to be identified, but if they were, a wavy underline would be expected.

In part (b) many candidates completed the entity relationship diagram correctly. However, incorrect relationships included in the answer were taken into account and therefore such answers could not score full marks. A member can make many bookings, but each booking could only relate to one member. A car could be booked many times but each booking could only be made for one car at a time. Each parking area could have many cars parked there, but a car could only be parked at one parking area at any one time. Non-standard notations are still in evidence, the most ambiguous one being arrow heads. Candidates should be reminded of the standard notation of 'crow's feet'.

Part (c) of the question did not state explicitly which attributes should be shown in the list of bookings, so reasonable answers gained credit. The search criteria caused many candidates considerable difficulties. A possible answer was: WHERE StartDateTime BETWEEN 1/12/07 AND 31/12/07. Another creditable answer, using a wildcard, was WHERE StartDateTime LIKE "*/12/07". The most popular answer was: WHERE StartDateTime >=1/12/07 AND StartDateTime <=31/12/07. However, a common error was to miss out the attribute in the part after the AND.

Q12.

(a) Some candidates did not identify the table clearly enough. Just writing down the number 2 was ambiguous because this could have referred to Figure 2 or Table 2. Most candidates correctly identified the Recipe table as the table not in first normal form. Far fewer candidates could describe correctly the reason. Many misunderstood the idea of repeating attributes and stated that certain ingredients were stored in the table more than once. Better candidates could explain that the data in the Ingredients column was not atomic, that is, there were multiple values in the Ingredients field.

(b) Few candidates could explain what fully normalised means. Only the best candidates could state that such tables would not contain any partial dependencies, or non-key dependencies. Most candidates knew that it is desirable to have fully normalised tables because it eliminates data inconsistency and unnecessary data duplication.

(c) Most candidates gained credit for completing the relations Recipe and FoodItem. Many candidates listed the correct attributes for the relation RecipeIngredient, but often did not gain credit because they also included many other attributes belonging to the other relations. This highlights the fact that candidates may rote learn the reasons for having fully normalised tables (see (b) above) but can not see that their answers do not in fact reflect the concept of eliminating unnecessary data duplication. Candidates should be aware that attribute identifiers should not contain spaces. Although this did not reduce the credit of their answers this time, in future papers this may reduce marks given.

(d) It was pleasing to see that many candidates gained high marks on this part of the question. Although this was not an easy question, most understood what was required, especially the Where part of the statement was well done. Common reasons for not gaining credit were still the prefixing of the table names with tbl. The names of the relations were clearly stated in the question part (c), and therefore these were expected to be used in the SQL statement. A minority of candidates also used the unconventional notation of attribute.tablename rather than tablename.attribute. When searching for a string value, this value must be enclosed in string delimiters. Different conventions were given credit, such as “Feta Salad” or ‘Feta Salad’.

Q13.

(a) A very small minority of candidates could state the full name of the law: Copyright, Designs and Patents Act 1998. Credit was given to those who managed to state the term Copyright, but candidates should be reminded that this may not be sufficient in future examinations.

(b) Few candidates scored all three marks for the Entity-Relationship diagram. Very few were able to name the relationship. Some did not use the standard E-R diagram shapes.

(c) Although DDL statements were asked for in a previous paper, only a small minority of candidates were able to write the statements correctly to create the required tables. Credit was given for identifying the composite key in the SoftwareInstallation table. Candidates need to be reminded of the fact that identifiers should not contain spaces when naming attributes. The choice of identifiers should convey some meaning. For example the question required the date the software was purchased and the expiry date to be stored. Suitable attribute identifiers might be PurchaseDate and ExpiryDate. Date on its own is not sufficient, especially if the data type used is also Date.

(d) The SQL statement was mostly correct, although few candidates gained all 4 marks. A common error still is to introduce more attributes in the SELECT part than required or include semicolons and other punctuation where not required. Some candidates did not include the ORDER BY part which was required to get a list of each computer with its installed software.

Q14.

(a) (i) Many candidates had the entity names the wrong way round and most candidates were unable to identify the label for C, with “one to many” a frequent, albeit insufficient, response.

(ii) The majority of candidates were able to identify the type of diagram correctly as an Entity Relationship Diagram

(b) This was the first time that DDL statements were required to create tables. Very few candidates seemed to have any experience of DDL and answered this part of the question using the standard table notation. Credit was given for correctly identifying the attributes and datatypes.

For the entity HardwareItem, many candidates were able to identify the attributes and data types for two marks, however they failed to identify the primary key. The use of “Autonumber” for a data type occurred too many times. Candidates were unable to identify the primary key in the EquipmentLoan table and most candidates used a made up primary key called LoanID rather than a composite key. Candidates were rarely able to successfully identify the foreign key.

(c) The SQL was generally well produced, with most candidates using SELECT, FROM, WHERE and ORDER BY clauses successfully. A majority of candidates failed to put the entity name before the InventoryReferenceNumber in the SELECT statement. Some candidates used the word “AND” between fields in the SELECT and between the table names in the FROM clause. Candidates are continuing to use “tbl” before the entity name, which is not appropriate. Many candidates were able to join the tables successfully in the WHERE clause and a few candidates made the mistake of using the wrong symbol in the DateRemoved > givenDate. Some candidates used the DateOfRemoval in the ORDER BY rather than Inventory Reference Number.

Q15.

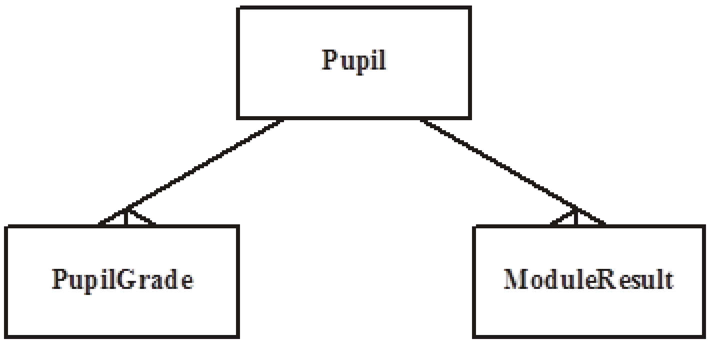
(a) Most candidates could identify the CandidateNumber as the primary key for ResultsTable.

(b) Many candidates are under the misconception that the example table shown in the question contained redundant data. Certain values appeared more than once, but this does not necessarily imply redundancy. The fact that there was another Ali Patel with a different candidate number is not data duplication either. The inclusion of this record was intended to steer candidates to understanding the importance of CandidateNumber as the primary key.

Other misconceptions were that any database consisting of a single table was by definition not normalised. Those candidates who could state that there was a repeating group of attributes gained credit. This repeating group of attributes were ModuleCode, ExamSession, ModuleMark, Level, TotalMark and Grade (they contain multiple values).

(c) A pleasing number of candidates could correctly identify which attributes belonged in which table for a fully normalised solution. However, it seemed to cause much more difficulty correctly identifying primary keys for these tables. Some candidates invented new attributes to act as primary keys. This did not gain credit.

(d) There are still unconventional ER diagrams in use. Candidates should show one-to-many relationships as below:



(e) The SQL is now generally well done. The majority of candidates gained 3 or 4 out of the 5 marks for the SQL statement. Most did not appreciate that there needs to be two criteria in the WHERE clause: SELECT PupilForenames,PupilSurname, Grade

FROM Pupil, PupilGrade

WHERE Pupil.CandidateNumber = PupilGrade.CandidateNumber AND Level=“A”

ORDER BY TotalMark DESC

Q16.

Very few candidates were able to identify *repeating group* as the reason why the table was unnormalised.

Many candidates stated that the reason was repeating attributes without thinking about what they were describing. “SubjectID value 4400 is repeated in the first and second rows of the table.

Therefore, this attribute is repeated in the table but this will happen in a fully normalised set of tables as well. *Repeating group* is the correct term to describe the state of the table. Other candidates gained this mark by implying *repeating group*. Their answers referenced one or more of SubjectID, SubjectName, ExamBoardSubjectOfficerName, NumberOfCandidates stating that for a given value of CentreNo these attributes contained multiple values.

Question (b) was answered reasonably well with most candidates forming relations that gained marks.

Candidates were also reasonably successful in (c) at identifying the relationships correctly.

Many candidates were able to structure a solution for (d) that employed SQL in an appropriate manner.

Several candidates placed relation names after attributes and not before as expected. These candidates were penalised only once for this error. Some candidates attached “tbl” to the beginning of every relation despite the relation names being given in the question. This is a practice that has no basis in database theory. These candidates were therefore penalised once for this error. The commonest error was an error of omission. The SQL *Where* sub clause “ExaminationOfficer.CentreNo = Problem.CentreNo” was frequently missing from candidates’ answers.

Very few candidates were unable to name the type of package most suitable for a mail merge operation, i.e. word processor, in (e).

Q17.

This was answered reasonably well on the whole. Although in part (a) some candidates were unable to give the correct degree of each relationship. The critical thinking skills demanded in part (a) appeared to be beyond some candidates.

Part (b) was answered reasonably well by most candidates. Errors that crept in were lexical rather than syntactical. For example, several candidates used a colon after some SQL keywords - “Select: “. Several candidates added “tbl” before each table name, e.g. “Select FirstName, Surname From tblStudent”. Other candidates wrote “FirstName.Student” instead of “Student.FirstName”. Very pleasingly, many candidates were able to join the two tables, Student and MarkAwarded, together using “Where Student.StudentID = MarkAwarded.StudentID”. Some candidates believed wrongly that an attribute used in the where clause must also appear in the select list - “Select ...LifeCyclePhaseID ..Where MarkAwarded.LifeCyclePhaseID = 1...”.

Q18.

(i) Candidates fared less well at identifying the attributes made visible by the Create View statement. Candidates wrongly identified Surname and FinesOwed. The correct answer was Name and TotalOfFines. Candidates who lacked experience of DDL would not necessarily have been prevented from gaining credit if they possessed some programming skill with and understanding of interfaces. The correct attributes could have been deduced.

(ii) Candidates fared better at explaining the data accessible through this view although some candidates lost this mark because of a lack of accuracy in their response. Candidates need to be made aware that precision of thought leads to precision of expression. Their practical work in this subject should aim to develop this precision of thought and expression. This is best done through the learning and applying of a programming language on a range of problem solving tasks. Although written paper questions do not often directly assess this skill and understanding it is clear to the examiners that it is reflected in the quality of candidates’ answers and therefore contributes indirectly. It is worth the candidate’s effort to learn to program because it makes many aspects of this subject more readily accessible.

Q19.

(a) This was answered reasonably well by the majority of the candidature. Part (iv) caused some candidates difficulty. The relationship between Gala and Swimmer is a many-to-many relationship. Some candidates seemed to think that such a relationship could not exist.

(b) Knowledge of and skill using SQL varied from very little to fully competent. Some candidates placed “tbl” in front of every table name and were penalised. Others made minor errors of syntax such as putting brackets around each attribute name. A careful reading of the question paper might have helped reduce syntax errors since figure 2 in question 4 gave an example of a simple Select statement. Candidates must gain some experience of writing SQL directly. Using QBE and then toggling the display to view the automatically generated SQL is not ideal. Often the SQL generated contains additional, unnecessary elements. Some candidates seemed to have learnt their SQL directly from past papers. These candidates included the “;” symbol where it has appeared in past paper mark schemes.

Q20.

On the whole this question was well answered. However, many candidates failed to identify accurately the reason why the table is un-normalised. The correct reason is that the table contains a repeating group. Simply stating that the table contains ‘repeating attributes’ is not sufficiently precise. An alternative acceptable answer was ‘one or more of RouteId, RouteName, RouteArea, RouteDescription contain multiple values’. Such an answer offered sufficient precision to be creditworthy.

Many candidates correctly identified the degree of the relationships. However, some of these candidates then failed to apply the rule that the primary key is exported into the many-side of the relationship when the relations are created. These candidates therefore failed to obtain the foreign key mark in part (b) (ii). Some candidates clearly had sufficient experience of SQL to write correct SQL statements. Others used incorrect syntax and therefore failed to gain full credit.

In part (e) many candidates attempted to name a package type as the question stipulated, but some gave brand names and so failed to gain credit.

Q21.

Candidates this year did not perform as well as in previous years on this type of question. The greater emphasis on writing SQL exposed gaps in some candidates’ knowledge. Whilst many candidates coped with a single table query, fewer were able to give the correct syntax for a query involving the joining of two tables. This question also exposed a lack of formal treatment of the theory of SQL. Several candidates referenced the tables **Book** and **BookCopy** as **tblBook** and **tblBookCopy**, possibly because they have experienced a software package that creates tables with this notation. Candidates lost marks for using quotes on the accession number 1234 and not using quotes on the ISBN value of “1-57820-082-2”.

In part (a) (ii) several candidates could not cope with the many-to-many relationship between entity **Book** and entity **BookCopy** and proceeded to add intersection entities so that two or more one-to-many relationships could be drawn, not accepting that the many-to-many relationship entity still existed. Other candidates drew an incorrect relationship. Many candidates successfully identified that the relationship between **Borrower** and **BookLoan** was one-to-many.

Many candidates correctly gave mail merge as the process in part (c).

Q22.

Many candidates correctly identified three relationships. In part (b) some candidates lost marks by not mapping the data requirements into the relations accurately. For example, some candidates used WardID and not the given WardName, NurseName and not the given NameOfNurseInCharge. Many candidates managed an acceptable SQL statement correctly joining the tables Patient and PatientMedicalCondition. The complete SQL statement was

```
Select Patient.Forename, Patient.Surname,
                       PatientMedicalCondition.MedicalConditionNo
From Patient, PatientMedicalCondition
Where Patient.WardName = ‘Victoria’
            And Patient.PatientNo = PatientMedicalCondition.PatientNo
```

Q23.

In the past, candidates have scored highly on a data analysis question of this type with full marks having been achieved by many candidates. Disappointingly, this time, very few candidates achieved full marks although the majority of the candidature managed at least twelve marks out of twenty. The analytical skill necessary for full marks in parts (a) and (b) was not in evidence on many scripts. In others, candidates omitted to underline the primary key despite being instructed to do so in the question stem or invented new attributes such as ProductCode. Few candidates identified the composite key correctly in part (b)(iv) ABCOrderNo, LineNo.

Candidates faired better at writing SQL with many candidates scoring at least four marks. Few candidates were able to construct a compound Where condition,

"Where Customer.CustomerId = Order.CustomerId
And Order.OrderHasBeenDespatched = True"

but managed to score a mark for stating one of the two required conditions. Common errors were including extra attributes in the Select clause and omitting one of the two required tables in the From clause. The correct SQL statement was:

```
Select CustomerName
From Order, Customer
Where Where Customer.CustomerId = Order.CustomerId
       And Order.OrderHasBeenDespatched = True
       Order By ABCOrderNo
```

Q24.

This was a popular question but not as well answered as similar questions in previous years. Pleasingly the majority of candidates did use the notation supplied in the question although some altered the given relationship to a many-to-many. No candidate was penalised for this error.

(a) Very few candidates were able to correctly show the degree of four relationships.

(b) Despite an incorrect relationship in part (a), the same candidates were often able to select the correct foreign key for the paired entities. Candidates seemed to be more at home working with tables (relations) than entities and their relationships.

Many candidates were able to describe the **Producer, Venue** and **Client** tables correctly. Some candidates used **ID** where **ProducerID**, **VenueID** and **ClientID** were required and therefore lost marks. Some candidates used **Name** where **ProducerName** and **VenueName** were required. Candidates must be precise when naming attributes.

Candidates need to ensure that their tables (relations) are normalised, i.e. contain no extra attributes. Several candidates committed normalisation errors that should have been easy to eliminate, such as including in **VenueHired**, the attribute **VenueAddress** in addition to the attribute **VenueID.**

Composite primary keys, as in previous years, are a source of difficulty for some candidates. The need to find a combination of attributes that is unique and minimal is not well understood or is a skill that not many candidates have mastered. Possession of the knowledge and ability to be able to select the correct combination of attributes is the hallmark of a good A Level student.

(c) This question revealed a lack of knowledge and experience in the use of SQL. A Select statement was required that performed a selection of data covering several attributes and from more than one table. The candidates' responses indicated little knowledge and experience beyond selection of data for one attribute within one table. Many candidates simply wrote four separate **Select** statements, one for each attribute involving a single table each time, either **Client** or **Agent**. Such answers were awarded credit but for the first **Select** statement only. The **Where**-clause was badly handled by the majority of candidates. To handle this correctly required experience of multiple table selections.