1	**1**	**All marks AO2 (analysis)** Torquay / the town name is missing quotations marks // needs quotation marks around it; There is no linking condition/join between the two tables using the SurgeryName // a join needs to be added to the `FROM` clause using the SurgeryName // a linking condition needs to be added to the `WHERE` clause using the SurgeryName // the condition `Surgery.SurgeryName = Vet.SurgeryName` is missing / needs adding; **NE.** the tables have not been linked.	**2**

2	1	**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 (`Property, Buyer`) and the fields that need to be extracted (`PropertyID, Street, Bedrooms, AskingPrice`), and including these and no other tables or fields in the query.
		
		1 mark for correctly identifying two conditions relating to how the data in the required tables should be combined to produce the desired results OR **2 marks** for identifying all four required conditions. The four conditions are:
		
		• `BuyerID = 23`
		• `Buyer.DesiredArea = Property.Area`
		• `Buyer.MinBedrooms <= Property.Bedrooms`
		• `Buyer.MaxPrice >= Property.AskingPrice`
		
		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 three/four clauses (SELECT, FROM, WHERE, ORDER BY)
		OR
		2 marks for fully correct SQL in all three/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 PropertyID, Street, Bedrooms, AskingPrice`
		`FROM Buyer, Property`
		`WHERE BuyerID = 23`
		`  AND DesiredArea = Area`
		`  AND MinBedrooms <= Bedrooms`
		`  AND MaxPrice >= AskingPrice`
		`ORDER BY AskingPrice DESC`
			5

3	**1**	**3 marks for AO1 (knowledge) – 3 marks:** • RFID reader/scanner (at warehouse entrance) transmits/sends signal; • Signal activates/energises/induces current in RFID transponder/tag; • RFID transponder/tag transmits/sends data by radio(wave); **3 marks for AO2 (analyse) – Max 3 marks:** • RFID signals processed into a format suitable for querying the database; • (SELECT) query used to check if there is already a record for the product/ProductID in the database // return of empty data set could be used to identify if the product is not in the table; **NE.** used to lookup record • UPDATE statement used to increase the QuantityInStock/stock level (by the number of items delivered) <u>if the product is already in the database</u>; • INSERT statement used to create new record for product <u>if it is not already in the database</u>; user will need to enter some details manually as these are not contained in the RFID tag.	**6**

Qu	Pt	Marking guidance	Total marks
4	1	**Mark is AO2 (analyse)** **B**; (FacilityID, BookingDate, EndTime) **R.** if more than one lozenge shaded	1

Qu	Pt	Marking guidance	Total marks
4	2	**All marks AO2 (analyse)** The design is not normalised // there is (unnecessary) data duplication // there is data redundancy // inconsistent data could occur // (one of the) attributes are determined by attributes that are not (part of) the primary key; If a customer made more than one booking then their details would need to be entered more than once / each time // there would be redundancy in relation to the customer data // customer data could be stored multiple times; If customer details were entered more than once they could be inconsistent // there could be inconsistency in the customer data // updates may need to be made to multiple records if a customer's details changed; **A.** updates to customer details would be harder to perform as **BOD** Deleting all of the bookings that a customer made would also delete the data about the customer; It would not be possible to store details about a customer before they had made a booking; It would be harder to identify all the bookings for one customer (as they did not have a unique identifier) // it may be impossible (**A.** difficult) to distinguish between two customers with the same name (if they did not have an email address); **Notes:** • For all mark points (other than the first) it must be stated that it is the customer data that is the issue to award the mark. • Accept points stated the other way around, ie as advantages of the new design instead of reasons to reject the original design. **Max 2**	2

Qu	Pt	Marking guidance	Total marks

4	3	**All marks AO3 (programming)**
		
		`FacilityID INT PRIMARY KEY,`
		//
		`FacilityID INT,`
		`PRIMARY KEY(FacilityID),`
		
		`Description VARCHAR(100),`
		`MaxPeople INT,`
		`PricePerHour SMALLMONEY`
		
		1 mark: `FacilityID`, with sensible data type and identified as primary key.
		
		1 mark: two fields other than the primary key have sensible data types and lengths (if given).
		
		1 mark: fully correct definition, with syntactically correct SQL including commas separating each line of code.
		
		A. any sensible types. Lengths do not need to be specified.
		I. brackets at the start / end of the code
		
		Valid alternative SQL types are:
		
		• Alternative types for `FacilityID` and `MaxPeople`: tinyint, smallint, mediumint, integer, number, byte.
		• Alternative types for `Description`: char, nchar, nvarchar, ntext, longvarchar, varchar2, nvarchar2, text, tinytext, mediumtext, longtext, string.
		• Alternative types for `PricePerHour`: money, float, real, decimal, double, numeric, currency. **R.** integer only types.
			3

Qu	Pt	Marking guidance	Total marks

4	4	**5 marks for AO2 (analyse) and 2 marks for AO3 (programming)**
		
		Mark Scheme
		
		AO2 (analyse) – 5 marks:
		
		1 mark for correctly analysing the data model and identifying the tables that data needs to be extracted from (`FacilityForSport, Booking`) and the fields that need to be extracted (`FacilityID, StartTime, EndTime`), and including these and no other tables or fields in the query
		
		1 mark for correctly identifying the condition to select facilities suitable for the correct sport: `Sport = "Basketball"` or correctly identifying the condition to select bookings on the required date: `BookingDate = "15/06/2021"`
		
		1 mark for correctly identifying the condition to link the two tables: `Booking.FacilityID = FacilityForSport.FacilityID`
		
		1 mark for at least one condition that would identify some overlapping bookings and no bookings that don't overlap, or **2 marks** for conditions that would identify all overlapping bookings and no bookings that don't overlap. Example conditions (not the only ones) that would identify all overlapping bookings:
		
		Example set of conditions 1
		
		`StartTime <= "14:15" AND EndTime >= "16:15"`
		(existing booking starts before and ends after new booking)
		`StartTime >= "14:15" AND StartTime <= "16:15"`
		(existing booking starts during new booking)
		`EndTime >= "14:15" AND EndTime <= "16:15"`
		(existing booking ends during new booking)
		
		Example set of conditions 2
		
		`StartTime <= "16:15" AND EndTime >= "14:15"`
		(existing booking starts before or at the same time as the end of new booking and ends after or at the same time as the start of new booking)
		
		Note: Award a **maximum of 2 of the 3 marks** for the correct conditions if they are not joined by the correct logical operators.
		
		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
		
		A. mark(s) can be awarded for the correct logical conditions even if the required tables are not identified as being used by the query
		A. > instead of >= and < instead of <=
			7

AO3 (programming) – 2 marks:

1 mark for fully correct SQL in two of the three clauses (SELECT, FROM, WHERE)

OR

2 marks for fully correct SQL in all three clauses (SELECT, FROM, WHERE)

Notes:

- For the SELECT clause to count as correct SQL it must have the correct field names in it and no others.
- For the FROM clause to count as correct SQL it must have the correct table names in it. Other, unnecessary tables can also be included so long as they are correctly linked into the query by conditions so would not break it.
- For the WHERE clause to count as correct SQL it must include at least one valid condition, but does not have to include them all.

A. instead of `FacilityForSport.FacilityID` accept `Booking.FacilityID` or just `FacilityID` in the SELECT clause for non-nested queries. For a nested query accept `X.FacilityID` where `X` is the alias of the relation produced by the nested query eg `BookingsAtTime` in example 3.
A. table names before fieldnames separated by a full stop.
A. use of `Alias`/`AS` command eg `FROM Booking AS B` then use of `B` as the table name and note that command `AS` is not required eg `FROM Booking B`.
A. `INNER JOIN` written as one word ie `INNERJOIN`.
A. insertion of spaces into fieldnames.
I. unnecessary brackets so long as they would not stop the query working.
A. use of any type of quotation marks, hashes or no delimiters around dates and times.
A. month in date as `6` instead of `06`
A. > instead of >= and < instead of <=
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.

Overall Max 6 if solution does not work fully

Example Solutions

Example 1 – All conditions in WHERE clause

```
SELECT FacilityForSport.FacilityID, StartTime, EndTime
FROM FacilityForSport, Booking
WHERE Sport = "Basketball"
  AND Booking.FacilityID = FacilityForSport.FacilityID
  AND BookingDate = "15/06/2021"
  AND
  ( StartTime <= "14:15" AND EndTime >= "16:15"
    OR StartTime >= "14:15" AND StartTime <= "16:15"
    OR EndTime >= "14:15" AND EndTime <= "16:15" )
```

Example 2 – Use of INNER JOIN

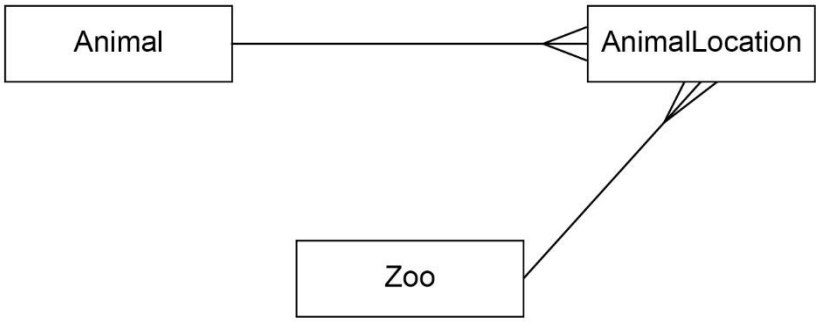
```
SELECT FacilityForSport.FacilityID, StartTime, EndTime
FROM FacilityForSport INNER JOIN Booking ON
  Booking.FacilityID = FacilityForSport.FacilityID
WHERE Sport = "Basketball"
  AND BookingDate = "15/06/2021"
  AND
  ( StartTime <= "14:15" AND EndTime >= "16:15"
    OR StartTime >= "14:15" AND StartTime <= "16:15"
    OR EndTime >= "14:15" AND EndTime <= "16:15" )
```

Example 3 – A Nested Solution

```
SELECT FacilityForSport.FacilityID, StartTime, EndTime
FROM ( SELECT FacilityID, StartTime, EndTime
       FROM Booking
       WHERE BookingDate = "15/06/2021"
         AND
         ( StartTime <= "14:15" AND EndTime >= "16:15"
           OR StartTime >= "14:15" AND StartTime <= "16:15"
           OR EndTime >= "14:15" AND EndTime <= "16:15" )
     ) AS BookingsAtTime INNER JOIN FacilityForSport
       ON BookingsAtTime.FacilityID =
         FacilityForSport.FacilityID
WHERE Sport = "Basketball"
```

Refer nested solutions to team leaders for marking

Question			Marks
5	1	**Mark is AO1 (understanding)** **C** The primary key in each relation consists of only one attribute; **R.** if more than one lozenge shaded	**1**

Question			Marks
5	2	**All marks AO2 (analyse)**  **1 mark**: one-to-many relationship between Animal and AnimalLocation **1 mark:** one-to-many relationship between Zoo and AnimalLocation If no marks awarded then award **1 mark** for many-to-many relationship between Animal and Zoo **Max 1** if any incorrect relationships drawn (ignore the inclusion of a many-to-many relationship between Animal and Zoo)	**2**

Question			Marks

5	3	**All marks AO3 (programming)**
		
		`AnimalID INT PRIMARY KEY,`
		//
		`AnimalID INT,`
		`PRIMARY KEY(AnimalID),`
		
		`IndividualName VARCHAR(50),`
		`Species VARCHAR(40),`
		`DateOfBirth DATE,`
		`Sex VARCHAR(6)`
		
		These **are AO3 marks so syntax must be correct** (including commas) to award them
		
		1 mark: `AnimalID`, with sensible data type and identified as primary key
		
		1 mark: two other fields with sensible data types and lengths (if given)
		
		1 mark: two other fields with sensible data types and lengths (if given)
		
		DPT. data type before fieldname (**Note: penalisation is of marks not mistakes**)
		DPT. incorrect punctuation - missing commas, unnecessary semi-colons, brackets etc but ignore bracket or semi-colon added at very end
		A. Any sensible types. Lengths do not need to be specified
		I. Case
		
		Valid alternative SQL types are:
		
		• Alternative types for `AnimalID`: tinyint, smallint, mediumint, integer, number, byte
		• Alternative types for `IndividualName`, `Species` and `Sex`: char, nchar, nvarchar, ntext, longvarchar, varchar2, nvarchar2, text, tinytext, mediumtext, longtext, string
		• Alternative types for `DateOfBirth`: datetime, datetime2, datetimeoffset, smalldatetime **R.** time
			3

Question			Marks

5	4	**5 marks for AO2 (analyse) and 2 marks for AO3 (programming)**
		
		Mark Scheme
		
		AO2 (analyse) – 5 marks:
		
		1 mark for correctly analysing the data model and identifying the tables that data needs to be extracted from (`Animal`, `AnimalLocation`) and the fields that need to be extracted (`IndividualName`, `DateArrived`), and including these and no other tables or fields in the query
		A. inclusion of unnecessary table `Zoo` as long as it is correctly linked to the `AnimalLocation` table by a linking condition
		
		1 mark for correctly identifying the condition to select the correct species of animal: `Species = "Red Panda"` or correctly identifying the condition to select the correct zoo: `ZooName = "Ashdale Park"`
		
		1 mark for correctly identifying the condition to link the two tables: `Animal.AnimalID = AnimalLocation.AnimalID` - see example 3 for how to apply this to nested solutions.
		R. do not award mark if additional linking conditions for tables that the query does not use are included
		
		1 mark for at least one pair of conditions that would identify some animals that were at the zoo during the required period, or **2 marks** for conditions that would identify all animals that were at the zoo during the period. Example conditions (not the only ones) that would do this are:
		
		Example full set of conditions 1 – award 2 marks for all conditions or 1 mark for any pair of conditions that would identify some animals at the zoo
		
		`DateArrived < "01/04/2020" AND DateLeft > "31/05/2020"`
		(animal arrived before and left after time period)
		
		`DateArrived <= "31/05/2020" AND DateLeft = "01/01/0001"`
		(animal arrived before end of time period and has not left)
		
		`DateArrived >= "01/04/2020" AND DateArrived <= "31/05/2020"`
		(animal arrived during the time period)
		
		`DateLeft >= "01/04/2020" AND DateLeft <= "31/05/2020"`
		(animal left during the time period)
		
		Example full set of conditions 2 – award 1 mark for the DateArrived condition and either of the DateLeft conditions or 2 marks for all three conditions
		
		`DateArrived <= "31/05/2020" AND (DateLeft >= "01/04/2020" OR DateLeft = "01/01/0001")`
		(animal arrived before end of time period and left after start of time period or has not left)
			7

Example incomplete conditions – award 1 mark for pair of conditions

`DateArrived >= "01/04/2020" AND DateLeft <= "31/05/2020"`
(animal arrived and left during the time period)

Note: Award a **maximum of 2 of the 4 available marks** for the correct conditions if they are not joined by the correct logical operators.

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.

A. mark(s) can be awarded for the correct logical conditions even if the required tables are not identified as being used by the query
A. > instead of >= and < instead of <=
A. ≥, ≤, => and =<

AO3 (programming) – 2 marks:

1 mark for fully correct SQL in two of the three clauses (SELECT, FROM, WHERE)

OR

2 marks for fully correct SQL in all three clauses (SELECT, FROM, WHERE)

Note:
- For the SELECT and FROM SQL clauses to count as correct SQL, they must have the correct field and table names in them.
- For the WHERE clause to count as correct it must include at least one correct condition, but does not have to include them all (ignore missing conditions or irrelevant conditions), however the whole WHERE clause must have correct syntax.

A. table names before fieldnames separated by a full stop
A. use of `Alias`/`AS` command eg `FROM AnimalLocation AS AL` then use of `AL` as the table name and note that command `AS` is not required eg `FROM AnimalLocation AL`
A. `INNER JOIN` written as one word ie `INNERJOIN`
A. insertion of spaces into fieldnames
I. unnecessary brackets so long as they would not stop the query working
A. use of any type of quotation marks, hashes or no delimiters around dates and times
A. > instead of >= and < instead of <=
R. ≥, ≤, => and =<
I. inclusion of an ORDER BY clause
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

Overall Max 6 if solution does not work fully

Example Solutions

Example 1 – All conditions in WHERE clause

```
SELECT IndividualName, DateArrived
FROM Animal, AnimalLocation
WHERE Species = "Red Panda"
  AND ZooName = "Ashdale Park"
  AND Animal.AnimalID = AnimalLocation.AnimalID
  AND
  ( DateArrived < "01/04/2020" AND DateLeft > "31/05/2020"
    OR DateArrived <= "31/05/2020" AND DateLeft = "01/01/0001"
    OR DateArrived >= "01/04/2020" AND DateArrived <=
       "31/05/2020"
    OR DateLeft >= "01/04/2020" AND DateLeft <= "31/05/2020" )
```

Example 2 – Use of INNER JOIN

```
SELECT IndividualName, DateArrived
FROM Animal INNER JOIN AnimalLocation ON
  Animal.AnimalID = AnimalLocation.AnimalID
WHERE Species = "Red Panda"
  AND ZooName = "Ashdale Park"
  AND
  ( DateArrived < "01/04/2020" AND DateLeft > "31/05/2020"
    OR DateArrived <= "31/05/2020" AND DateLeft = "01/01/0001"
    OR DateArrived >= "01/04/2020" AND DateArrived <=
       "31/05/2020"
    OR DateLeft >= "01/04/2020" AND DateLeft <= "31/05/2020" )
```

Example 3 – A Nested Solution

```
SELECT IndividualName, DateArrived
FROM (SELECT AnimalID, IndividualName
      FROM Animal
      WHERE Species = "Red Panda"
     ) AS RP INNER JOIN AnimalLocation
       ON RP.AnimalID = AnimalLocation.AnimalID
WHERE ZooName = "Ashdale Park"
  AND
  ( DateArrived < "01/04/2020" AND DateLeft > "31/05/2020"
    OR DateArrived <= "31/05/2020" AND DateLeft = "01/01/0001"
    OR DateArrived >= "01/04/2020" AND DateArrived <=
       "31/05/2020"
    OR DateLeft >= "01/04/2020" AND DateLeft <= "31/05/2020" )
```

Refer nested solutions to team leaders for marking

Question			Marks
5	5	**All marks AO2 (analyse)** **Advantage (Max 1):** It will be quicker to lookup an animal's current location; The current location of an animal can be identified without having to query/search the AnimalLocation relation // only the Animal/one relation needs to be searched to identify the location of an animal // the current location of an animal can be identified with a less complex query/search; **NE.** <u>easier</u> to lookup an animal's current location **R.** it will be possible to identify an animal's current location **Disadvantage (Max 1):** Additional storage space will be required; This will introduce data redundancy (as the information can already be found from the AnimalLocation relation); Data inconsistency could occur (as the current location in the Animal relation might not match the current location in the AnimalLocation relation); More updates will be required when an animal is moved between zoos; **A.** the database will no longer be normalised	**2**

Qu	Pt	Marking guidance	Total marks
6	1	**All marks AO2 (analysis)** Customer(<u>CustomerID</u>, FirstName, LastName, TelephoneNumber) Booking(<u>BookingID</u>, ShowingID, CustomerID) AssignedSeat(<u>BookingID, SeatNumber</u>) **1 mark:** Customer relation created and contains the correct attributes and no others. **1 mark:** Booking relation created and contains the correct attributes and no others (but see accept point below). **A.** Inclusion of NumberOfPeople or NumberOfSeats attribute – must be clear from attribute name that this is a count/quantity **I.** BookingID not included (as ShowingID and CustomerID could be composite entity identifier) **1 mark:** AssignedSeat relation created and contains the entity identifier from the Booking relation, the SeatNumber and optionally AssignedSeatID. **A.** Entity identifier from Booking relation not identified in Booking relation **1 mark** for correct entity identifiers in one or two relations or **2 marks** for correct entity identifiers in all three relations. The correct entity identifiers are: • Customer: CustomerID **R.** composite entity identifier of FirstName and LastName, identifier based on TelephoneNumber • Booking: BookingID // composite entity identifier of ShowingID and CustomerID • AssignedSeat: Composite entity identifier of entity identifier from Booking relation and SeatNumber (**A.** including AssignedSeatID) // AssignedSeatID // ShowingID and SeatNumber **For all mark points** **A.** Spaces in relation and attribute names. **A.** Alternative names for relations and attributes created by candidate, as long as meaning is clear. **R.** Use of incorrect attribute names for attributes already used in relations defined on the question paper. **I.** Any representation for foreign keys. **Accept responses written in SQL – ignore syntactical errors and data type errors in such responses.**	5

Qu	Pt	Marking guidance	Total marks
6	2	**All marks AO2 (analysis)** The `Film` table should not be included // only the `Showing` table should be included; The date is missing quotation marks/hashes/delimiters; **A.** An asterisk / list of attributes is missing <u>after</u> `DELETE` **NE.** Not specified what to delete **Max 2**	**2**

Qu	Pt	Marking guidance	Total marks
6	3	**1 mark AO2 (analysis) and 1 mark AO1 (understanding)** **1 mark AO2 (analysis):** There might already be bookings for (a showing/showing on) this date; **A.** "There might already be bookings for these showings" without date reference **R.** "There might already be bookings for a/the showing" without date reference **1 mark AO1 (understanding):** The database would prevent the query from running as there would be records in the bookings table that referenced showings that no longer existed // if executed the query could leave records/bookings (in the bookings table) that referenced showings that no longer existed // there will be ShowingIDs that reference showings that do not exist; **A.** The foreign key rules might be violated **A.** Referential integrity rules might be violated **A.** Any bookings for the showings would also need to be deleted **Award both marks if stated that all bookings for the 29th March would also need to be deleted**	**2**

Qu	Pt	Marking guidance	Total marks
7	1	**Mark is AO2 (analyse)** **B**; (Each product is only supplied by one supplier) **R.** if more than one lozenge shaded	**1**

Qu	Pt	Marking guidance	Total marks
7	2	**All marks are AO3 (programming)** **1 mark:** `INSERT INTO Sale` // `INSERT INTO Sale (SaleID, CustomerID, SaleDate)` If field list given in `INSERT INTO` command then allow fields in any order, but must include all three fields. **1 mark:** `VALUES (4072, 48, "29/09/2024")` If field list given in `INSERT INTO` command then values must match order in that command. If field list not given then values must be in order shown, otherwise this mark cannot be awarded. **A.** use of # or ' as date delimiters instead of " **A.** other date formats **R.** no delimiters around date **R.** use of delimiters around SaleID or CustomerID **Max 1** if VALUES command before `INSERT INTO` command. **Max 1** if command would not work.	**2**

Qu	Pt	Marking guidance	Total marks

7	3	**All marks are AO3 (programming)**
		
		Mark against Alternative 1 unless the response includes the command FROM and the SaleLine table, in which case mark against Alternative 2.
		
		Alternative 1
		
		1 mark: `UPDATE Product`
		
		1 mark: `SET QuantityInStock = QuantityInStock - 3`
		
		1 mark: `WHERE ProductID = 1`
		
		Alternative 2 (works in some SQL implementations)
		
		Part mark A: `UPDATE Product`
		
		1 mark: `SET QuantityInStock = QuantityInStock – 3` //
		`SET QuantityInStock=QuantityInStock–QuantitySold`
		
		Part mark B: `FROM Product INNER JOIN SaleLine ON`
		`Product.ProductID = SaleLine.ProductID`
		
		1 mark: `WHERE SaleLine.SaleID = 4072`
		
		Note that both part marks (A and B) must be achieved to award one mark.
		
		A. table name before fieldname
		I. case, spaces in fieldnames
		NE. use of variable names instead of 3
		DPT. fieldname before table name
		R. use of delimiters around 1, 3 or 4072
		
		Max 2 if commands not given in the correct order or would not work
			3

Qu	Pt	Marking guidance	Total marks
7	4	**All marks are AO1 (understanding)** Timestamps are generated for each transaction // timestamps indicate the order that transactions occurred in; **A.** timestamps generated for edits / queries / requests as BOD **NE.** transactions are put in a queue without reference to timestamps **NE.** transactions are processed in order by time when they were made **R.** timestamps generated for sales Database records time(stamp) of last read / last write transaction for each record / data item; **A.** just one of read/write **A.** "edit" for "write", "access" for "read" **R.** file Database server applies rules to determine if processing a transaction <u>will result in loss of data integrity / inconsistency</u> (and if so aborts the transaction); **A.** Examples of rules for third mark point: • if a transaction tries to **write** to a record / data item, then the transaction should be aborted if the read/write timestamp on the record / data item is greater than the time at which the transaction started • if a transaction tries to **read** a record / data item, then the transaction should be aborted if the write timestamp on the record / data item is greater than the time at which the transaction started.	**3**

Qu	Pt	Marking guidance	Total marks
7	5	**All marks are AO1 (understanding)** Redundant/duplicated data may waste storage space; **NE.** data redundancy, data duplication If data is stored more than once then it could be inconsistent // two copies of the 'same' data item might store different values; **NE.** data inconsistency If data is stored more than once then each copy of the data would need to be updated if it changed; **NE.** eliminate update anomalies It might not be possible to store data about one type of entity without creating a record for another type of entity // if a record for one type of entity does not exist then it might not be possible to store data about another type of entity; **NE.** eliminate insertion anomalies **A.** example in context When a record for one type of entity is deleted it might delete the data about another type of entity // it might not be possible to delete a record for one type of entity without deleting the data about another type of entity; **NE.** eliminate deletion anomalies **A.** example in context May be difficult to select/edit data if it is not atomic // if there are repeating groups; **NE.** harder to update / insert / delete without concrete example or good explanation **NE.** more errors when updating / inserting / deleting without concrete example or good explanation **NE.** uses more space / memory **NE.** harder / slower to query	**2**