

Database Implementation Report: Bookworm Books (Microsoft Access)

Sample student report (model answer) | Module: [code and title] | Student: [name and ID] | Date: [date]

1. Introduction

This report is the third part of my work on **Bookworm Books**, the small online bookshop described in my design report. In that report I analysed the scenario and drew an Entity Relationship Diagram (Part A), then normalised the data from the shop's paper order forms from Unnormalised Form (UNF) to Third Normal Form (3NF) (Part B). Both routes gave the same four tables: CUSTOMER, ORDER, BOOK and ORDER_ITEM.

In this report I **implement that design in Microsoft Access** (Part C). I create the tables and relationships, enter the sample data from the three paper order forms, and test whether the database behaves as my design predicts. The report follows the order in which I built the database: planning the table definitions (section 2), creating the tables (section 3), creating the relationships (section 4), entering the data (section 5) and testing (section 6). Each step is shown with a screenshot (Figures 1 to 8). The database file is `BookwormBooks.accdb`, built in Microsoft Access [insert version].

2. From Design to Access

2.1 Table definitions

Before opening Access, I turned each table in my final relational schema into a table definition with data types. Table 1 shows the result.

Table 1: Planned Access tables, fields and data types

Access table	Fields (data type)	Primary key	Foreign key
tblCustomer	CustomerID (Short Text, 10), CustomerName (Short Text, 50), CustomerEmail (Short Text, 100)	CustomerID	none
tblOrder	OrderID (Short Text, 10), OrderDate (Date/Time), CustomerID (Short Text, 10)	OrderID	CustomerID refers to tblCustomer
tblBook	BookID (Short Text, 10), Title (Short Text, 100), Price (Currency)	BookID	none
tblOrderItem	OrderID (Short Text, 10), BookID (Short Text, 10), Quantity (Number, Integer)	OrderID + BookID	OrderID refers to tblOrder; BookID refers to tblBook

2.2 Design decisions

- **Table names.** I prefixed each table with `tbl` so that tables and queries are easy to tell apart. The design table ORDER became `tblOrder` because ORDER is a reserved word in SQL (as in `ORDER BY`). A table called Order would need square brackets in every query.
- **Short Text IDs.** I kept the IDs from the paper forms (C1, B1, O101), so my data matches the tables in the design report exactly. A foreign key must have the same data type and size as the primary key it points to, so every ID is Short Text with Field Size 10.

- **Other data types.** Price is **Currency**, which stores exact two-decimal amounts and shows the £ sign. OrderDate is **Date/Time**, so dates can be sorted and compared. Quantity is a **Number**, because copies are whole numbers.
- **Rules.** I made every non-key field Required. Key fields cannot be empty anyway, because Access does not allow a Null value in a primary key. Quantity must be greater than zero, and Price cannot be negative.

3. Creating the Tables

I created the database with **Blank database**, then built each table in Design View. I made the parent tables first (tblCustomer, tblBook), then tblOrder, then tblOrderItem, so that every foreign key points at a table that already exists.

3.1 tblCustomer

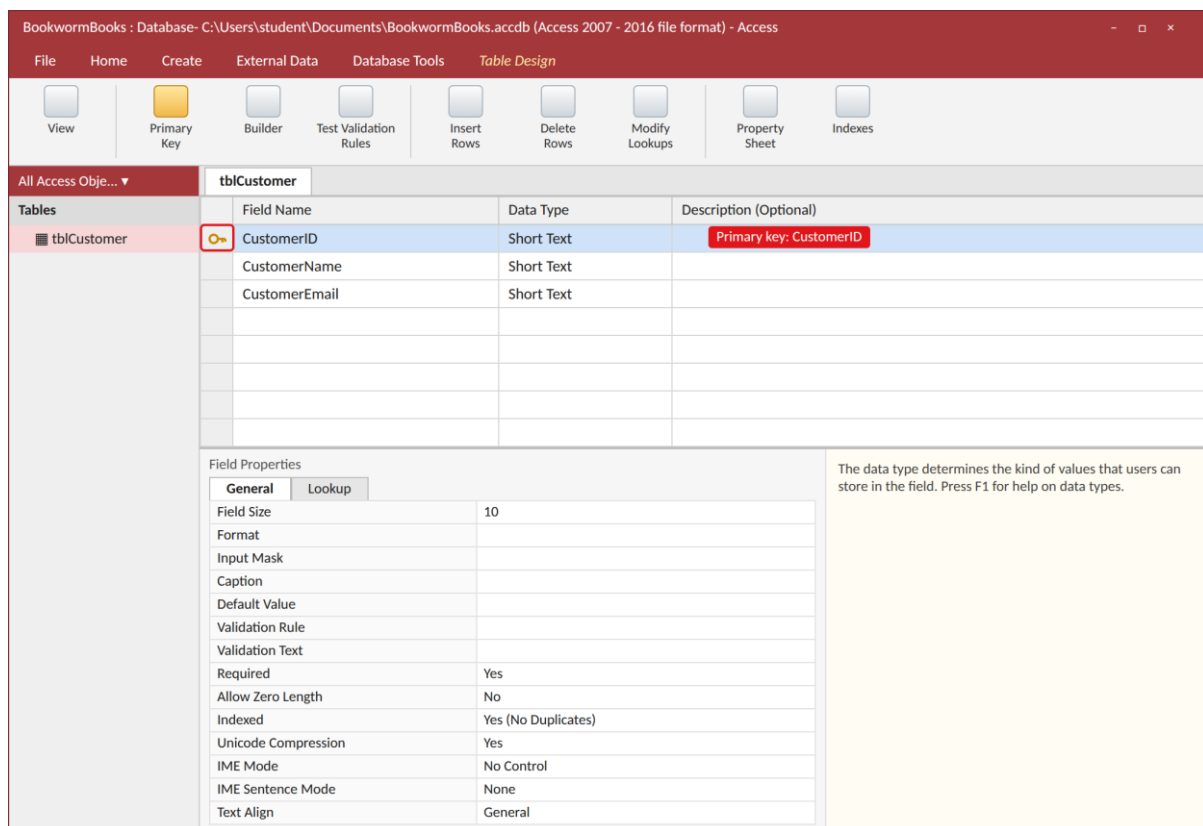


Figure 1: *tblCustomer in Design View, with CustomerID set as the primary key.*

Figure 1 shows tblCustomer. The key icon beside CustomerID shows it is the primary key. Access automatically sets the Indexed property to “Yes (No Duplicates)”, which is what stops two customers sharing an ID. CustomerName and CustomerEmail are Required, so a customer cannot be saved without them. I built tblBook and tblOrder in the same way, using the settings in Table 2.

Table 2: Field settings for *tblBook* and *tblOrder*

Table	Field	Data type	Size or format	Other settings
tblBook	BookID	Short Text	10	Primary key
tblBook	Title	Short Text	100	Required: Yes
tblBook	Price	Currency	Currency, 2 decimal places	Required: Yes. Validation rule: >=0
tblOrder	OrderID	Short Text	10	Primary key
tblOrder	OrderDate	Date/Time	dd-mm-yy	Required: Yes
tblOrder	CustomerID	Short Text	10	Required: Yes. Foreign key to tblCustomer

3.2 tblOrderItem and the composite primary key

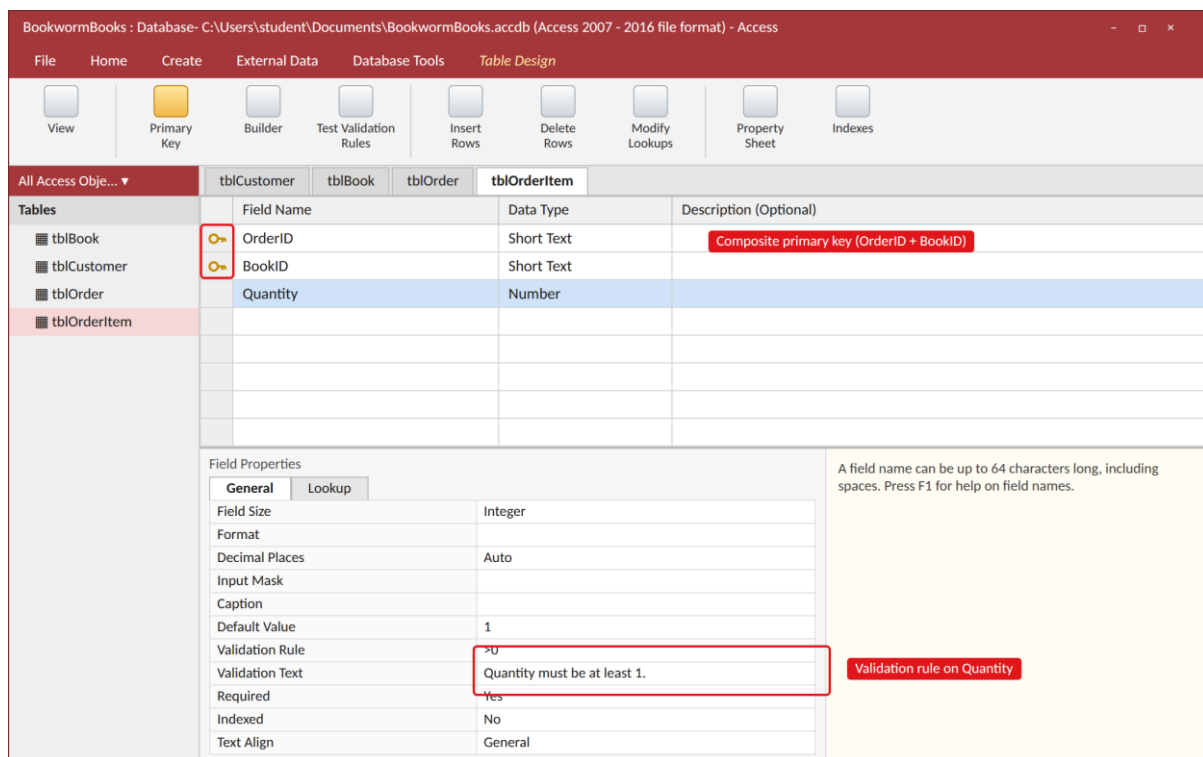


Figure 2: *tblOrderItem* in Design View, with a composite primary key and a validation rule on Quantity.

tblOrderItem is the linking table that resolved the many-to-many relationship between ORDER and BOOK in section 2.4 of my design report. Its primary key has two fields, so I clicked the row selector of OrderID, held down Ctrl, clicked the row selector of BookID, and then clicked **Primary Key**. Figure 2 shows a key icon beside both fields.

This key allows the same book to appear on many orders, and an order to contain many books, but it stops the same book being entered twice on one order. That matches the assumption in section 5 of my design report. Quantity is a Number (Integer) with a default of 1 and the validation rule >0 . The validation text is “Quantity must be at least 1.”, so zero or negative quantities are rejected.

4. Creating the Relationships

With all tables closed, I opened **Database Tools > Relationships** and added the four tables. For each relationship I dragged the primary key from the “one” side onto the matching foreign key on the “many” side: `tblCustomer.CustomerID` onto `tblOrder.CustomerID`, `tblOrder.OrderID` onto `tblOrderItem.OrderID`, and `tblBook.BookID` onto `tblOrderItem.BookID`.

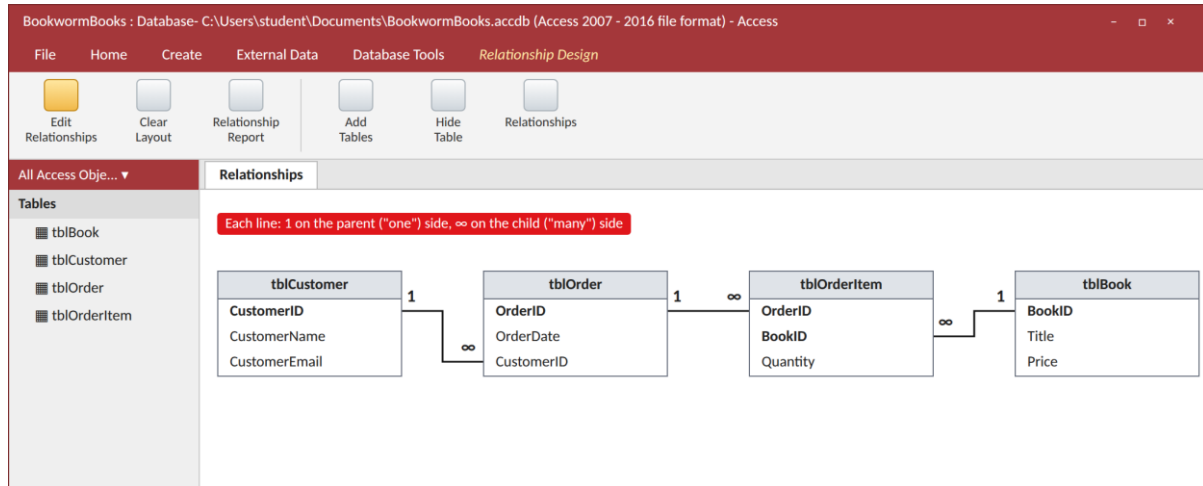


Figure 3: The Relationships window showing all four tables and their one-to-many relationships.

Figure 3 shows the finished Relationships window. The symbol **1** appears at the parent end of each line and **∞** (infinity) at the child end. The four boxes are in the same left-to-right order as the entities in my ERD, with the same primary keys and foreign keys. The Access relationships are therefore the ERD, built in a working database.

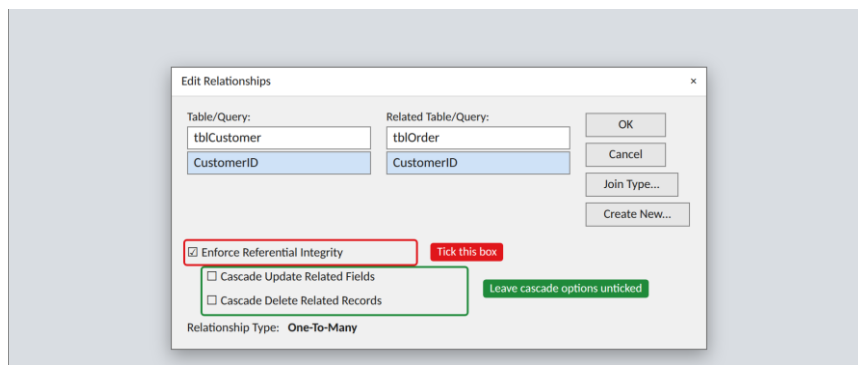


Figure 4: The Edit Relationships dialog for `tblCustomer` and `tblOrder`, with **Enforce Referential Integrity** ticked.

Each time I dropped a key, the dialog in Figure 4 opened. I ticked **Enforce Referential Integrity**. This tells Access to refuse any foreign key value that does not exist in the primary key it points to, and to refuse deleting a row that still has related rows. I left both Cascade options unticked, because I do not want deleting a customer to silently delete all of their orders. Access reported the relationship type as **One-To-Many**, as predicted in section 2.3 of my design report.

5. Entering the Data

I entered the data in Datasheet View in this order: `tblCustomer`, `tblBook`, `tblOrder`, `tblOrderItem`. The order matters. With referential integrity on, a row in `tblOrder` cannot be saved until its customer exists, and a row in `tblOrderItem` cannot be saved until both its order and its book exist.

The screenshot shows the Microsoft Access interface with four tables displayed in Datasheet View. The top ribbon includes 'Home', 'Create', 'External Data', and 'Database Tools'. Below the ribbon are icons for View, Paste, Filter, New, Save, Delete, Find, and Sort & Filter.

tblCustomer

CustomerID	CustomerName	CustomerEmail	Click to Add
C1	Aisha Khan	aisha@mail.com	
C2	Tom Evans	tom@mail.com	

Record: 1 of 2 | No Filter | Search

tblBook

BookID	Title	Price	Click to Add
B1	Python Basics	£20.00	
B2	Web Design	£15.00	
B3	Databases	£25.00	

Record: 1 of 3 | No Filter | Search

tblOrder

OrderID	OrderDate	CustomerID	Click to Add
O101	01-Oct-2026	C1	
O102	03-Oct-2026	C2	
O103	05-Oct-2026	C1	

Record: 1 of 3 | No Filter | Search

tblOrderItem

OrderID	BookID	Quantity	Click to Add
O101	B1	1	
O101	B2	2	
O102	B1	1	
O103	B3	1	

Record: 1 of 4 | No Filter | Search

Figure 5: The four tables in Datasheet View with the data from the three paper order forms.

Figure 5 shows the data: 2 customers, 3 books, 3 orders and 4 order lines. These are the same values as the 3NF tables in section 3.5 of my design report. Each fact is typed only once. For example, Aisha Khan's name and email appear in one row of tblCustomer, and orders O101 and O103 refer to her by CustomerID C1. Access displays the prices as £20.00 because of the Currency format, and the dates as 01-Oct-2026 because of the custom date format.

6. Testing the Database

6.1 Rebuilding the order forms with a query

To show that no information was lost when I split the data into four tables, I wrote a query called qryOrderLines. Access drew the join lines automatically from the relationships (Figure 6). I added a calculated field, LineTotal: [Quantity]*[Price].

The screenshot shows the Microsoft Access interface with the qryOrderLines query in Design View. The top ribbon includes 'File', 'Home', 'Create', 'External Data', 'Database Tools', and 'Query Design'. Below the ribbon are icons for Run, View, Select, Make Table, Append, Update, Totals, and Property Sheet.

qryOrderLines

Tables

- tblBook
- tblCustomer
- tblOrder
- tblOrderItem

Queries

- qryOrderLines

Field:

Field	OrderID	OrderDate	CustomerName	Title	Quantity	Price	LineTotal: [Quantity]*[Price]
Table:	tblOrder	tblOrder	tblCustomer	tblBook	tblOrderItem	tblBook	
Sort:	Ascending						
Show:	☒	☒	☒	☒	☒	☒	☒
Criteria:							
or:							

▲ Calculated field: LineTotal

Figure 6: qryOrderLines in Design View, joining all four tables with a calculated LineTotal field.

OrderID	OrderDate	CustomerName	Title	Quantity	Price	LineTotal	Click to Add
O101	01-Oct-2026	Aisha Khan	Python Basics	1	£20.00	£20.00	
O101	01-Oct-2026	Aisha Khan	Web Design	2	£15.00	£30.00	
O102	03-Oct-2026	Tom Evans	Python Basics	1	£20.00	£20.00	
O103	05-Oct-2026	Aisha Khan	Databases	1	£25.00	£25.00	

Figure 7: The result of *qryOrderLines*, with four rows and a *LineTotal* for each.

Figure 7 shows four rows. They match the 1NF table in section 3.3 of my design report (same orders, customers, titles, quantities and prices), with line totals of £20.00, £30.00, £20.00 and £25.00. This shows that the 1NF table can be rebuilt from the 3NF tables by joining them, even though each fact is now stored only once.

6.2 Testing referential integrity

I then tried to break the database. In *tblOrderItem* I typed a new row with OrderID O999, BookID B1 and Quantity 1. No order O999 exists in *tblOrder*, and Access refused the row with the message in Figure 8. I pressed Esc to cancel it, so the invalid row was never saved.

OrderID	BookID	Quantity	Click to Add
O101	B1	1	
O101	B2	2	
O102	B1	1	
O103	B3	1	
O999	B1	1	

Figure 8: Access refuses an order line for an order that does not exist.

This is the protection that the foreign key in the design is meant to give. Without enforced referential integrity, the database would hold an order line that belongs to no order.

6.3 Summary of tests

Table 3: Tests carried out and their results

No.	Action	Expected	Result	Pass?
1	Add an order line for non-existent order O999	Refused	Refused: "You cannot add or change a record because a related record is required in table 'tblOrder'." (Figure 8)	Pass
2	Delete customer C1, who has two orders	Refused	Refused: "The record cannot be deleted or changed because table 'tblOrder' includes related records." C1 remained.	Pass
3	Add an order line with Quantity 0	Refused by the validation rule	Access showed "Quantity must be at least 1."	Pass
4	Add book B4, Networking, £30.00, which nobody has ordered	Accepted	Saved without any order. The insert problem is solved.	Pass
5	Change "Aisha Khan" to "Aisha K. Khan" in tblCustomer, then re-run qryOrderLines	One edit changes every order	Rows for O101 and O103 both showed the new name. I changed it back. The repeated-data problem is solved.	Pass
6	Add a second line for O101 and B1	Refused (duplicate composite key)	Access refused: the change would "create duplicate values in the index, primary key, or relationship".	Pass

Tests 5, 4 and 2 match the three problems I listed in section 3.1 of my design report: repeated data, the insert problem and the delete problem. All three are now handled by the structure of the database.

7. Evaluation

The database does what the design predicted. Every table has a primary key, every relationship enforces referential integrity, and a single query rebuilds the original order forms. The six tests in Table 3 all passed. I also found some limitations:

- **Price history.** The price is stored only in tblBook. If a price changes, qryOrderLines would recalculate every past order at the new price. A real system should add a **PricePaid** column to tblOrderItem, as I noted in section 5 of the design report.
- **Typed IDs.** Short Text IDs such as O101 must be typed by hand, so staff could mistype or duplicate them. An AutoNumber key would be safer, but I used Short Text so that my data matched the design report.
- **Rules Access cannot enforce here.** Relationships cannot guarantee that every order has at least one order line, or that a customer's email address is unique. These would need a form with validation or a data macro.
- **Small sample.** I tested with only 2 customers, 3 books and 3 orders. A larger sample would give more confidence, especially for the query.

8. Conclusion

I implemented the 3NF design for Bookworm Books in Microsoft Access by creating four tables with suitable data types, a composite primary key on tblOrderItem, and three one-to-many relationships with referential integrity enforced. After entering the data from the three paper order forms, a query rebuilt the

1NF table, and six tests showed that the database refuses bad data and stores each fact once. The implementation confirms that the ERD and the normalised tables describe the same working design.

Marker's Notes (remove this page before issuing to students)

Suggested structure for the student brief: Introduction, planned table definitions, creating tables, relationships, data entry, testing (query and integrity tests), evaluation, conclusion.

Marking points to look for

- Tables are built from the design, with correct primary keys and **sensible, justified data types**.
- Each foreign key has the **same data type and size** as the primary key it points to.
- The linking table has a **composite primary key**, and the report explains why.
- Relationships run from the “one” side to the “many” side, with **referential integrity enforced** and cascade options chosen deliberately.
- Data is entered **parent tables first**, and the student can explain why.
- Testing includes evidence that bad data is refused, and a query that rebuilds the original forms.
- Every screenshot is **captioned and explained**, and the report links back to the ERD and the normalisation.
- Higher-level work evaluates the build (for example price history, typed IDs, rules Access cannot enforce).

Common errors

- A table called Order, which causes SQL problems later.
- A single-field primary key on tblOrderItem, so a second book cannot be added to an order.
- AutoNumber foreign keys that do not match a Short Text primary key.
- Entering child rows before parent rows, then being unable to save them.
- Screenshots pasted in with no caption or explanation.
- Ticking Cascade Delete with no justification.

Note on the figures Figures 1 to 8 are **illustrative recreations** drawn to look like Access screens. They are not captures from a live Access session. The ribbon and dialog layout differ between Access versions, so replace them with your own screenshots from the lab version of Access if you want an exact match before issuing.