

Solutions

Each answer shows the steps, then the **result**.

2.1

- (a) BOOLEAN – it has exactly two states
- (b) DATE
- (c) REAL – it has a fractional part (also accepted: DECIMAL(6,2))
- (d) VARCHAR(100) – text of varying length; any sensible n is accepted

2.2

- CREATE DATABASE Library;

2.3

- ALTER TABLE LOAN
- ADD FOREIGN KEY (MemberID) REFERENCES MEMBER(MemberID);

2.4

- SELECT COUNT(*) – or COUNT(LoanID)
- FROM LOAN;
- no GROUP BY, because the question wants one number for the whole table

2.5

- SELECT, FROM, WHERE, GROUP BY, ORDER BY
- an INNER JOIN ... ON goes with the FROM, between FROM and WHERE

3.1

- CREATE TABLE LOAN (with the closing);
- LoanID INTEGER PRIMARY KEY
- MemberID INTEGER, LoanDate DATE
- BookTitle VARCHAR(100), Fine REAL
- FOREIGN KEY (MemberID) REFERENCES MEMBER(MemberID)

```
CREATE TABLE LOAN (  
    LoanID INTEGER PRIMARY KEY,  
    MemberID INTEGER,  
    BookTitle VARCHAR(100),  
    LoanDate DATE,  
    Fine REAL,  
    FOREIGN KEY (MemberID) REFERENCES MEMBER(MemberID)  
);
```

3.2

- SELECT M.Name, L.BookTitle

- FROM MEMBER M INNER JOIN LOAN L
- ON M.MemberID = L.MemberID;

```
SELECT M.Name, L.BookTitle  
FROM MEMBER M INNER JOIN LOAN L  
    ON M.MemberID = L.MemberID;
```

- a query that lists both tables without an ON earns the first mark only

3.3

- SELECT M.Name, COUNT(L.LoanID)
- the join with its ON clause
- GROUP BY M.Name
- ORDER BY the count DESC

```
SELECT M.Name, COUNT(L.LoanID) AS NumLoans  
FROM MEMBER M INNER JOIN LOAN L  
    ON M.MemberID = L.MemberID  
GROUP BY M.Name  
ORDER BY NumLoans DESC;
```

- M.Name is in the SELECT and is not inside an aggregate, so it **must** be in the GROUP BY

3.4

- SELECT AVG(L.Fine)
- the join with its ON clause
- WHERE M.JoinDate < '2024-01-01';

```
SELECT AVG(L.Fine)  
FROM MEMBER M INNER JOIN LOAN L  
    ON M.MemberID = L.MemberID  
WHERE M.JoinDate < '2024-01-01';
```

3.5

- the ON clause is **missing**, so nothing says which rows of the two tables belong together
- every member is therefore paired with every loan, giving $100 \times 400 = 40\,000$ rows
- the corrected query adds ON M.MemberID = L.MemberID;, which keeps only the 400 pairings where the foreign key matches the primary key