

Solutions

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

2.1

- DDL defines/changes the **structure** (tables, keys)
- DML works with the **data** (insert, update, delete, query)

2.2

- SELECT ... FROM the student table, then filter Year 12
- SELECT Name FROM Student WHERE Year = 12;

2.3

- INSERT INTO ... VALUES
- INSERT INTO Student (StudentID, Name, Year) VALUES (9, 'Sam', 11);

2.4

- ALTER (TABLE) — a DDL statement

3.1

```
CREATE TABLE Book (  
    BookID INTEGER PRIMARY KEY,  
    Title  VARCHAR(100) NOT NULL,  
    Author VARCHAR(50)  
);
```

- CREATE TABLE + fields; PRIMARY KEY; NOT NULL on Title

3.2

- UPDATE ... SET
- UPDATE Student SET Year = 13 WHERE StudentID = 5;

3.3

- both tables, joined on StudentID
- SELECT Student.Name, Grade.Mark FROM Student, Grade WHERE Student.StudentID = Grade.StudentID;

3.4

- DELETE FROM
- DELETE FROM Student WHERE Year = 13;

3.5

```
CREATE TABLE Grade (  
    StudentID INTEGER,  
    Mark      INTEGER,  
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID)  
);
```

- CREATE TABLE + fields; FOREIGN KEY; REFERENCES the correct table/field

3.6

- SELECT ... WHERE and ORDER BY
- SELECT Name FROM Student WHERE Year = 12 ORDER BY Name;

3.7

(a) one ship has **many** containers, so it is a one-to-many relationship

- ShipID is the **primary key** of SHIP, uniquely identifying each ship
- ShipID in CONTAINER is a **foreign key** referring to it, which links each container to its ship

(b) ALTER TABLE CONTAINER

- ADD LastInspected DATE;

(c)

```
SELECT COUNT(*)  
FROM CONTAINER, SHIP  
WHERE CONTAINER.ShipID = SHIP.ShipID  
      AND SHIP.ShipName = 'Aurora';
```

- SELECT COUNT; both tables; the join condition; the name condition, quoted

(d)

```
SELECT ShipName, SUM(Weight)  
FROM SHIP, CONTAINER  
WHERE SHIP.ShipID = CONTAINER.ShipID  
GROUP BY ShipName  
HAVING SUM(Weight) > 500  
ORDER BY SUM(Weight) DESC;
```

- SUM with GROUP BY; the join; HAVING — not WHERE, because the condition is on an aggregate; ORDER BY ... DESC

(e) the developer interface is the part of the DBMS used to **create and maintain** the database

- defining tables and data types, setting keys and relationships, and managing users and access rights
- the query processor only **runs** queries against a structure that already exists — it cannot create it