Temasek Junior College

2024 JC2 H2 Computing

Database 5 – Structured Query Language (SQL)

Section	3	Data and Information
Unit	3.3	Databases and Data Management
Objectives	3.3.8	Write SQL statements and use a programming language to work with SQL databases

1 Introduction

Structured Query Language (SQL) is the standard language for working with databases and is implemented within most database management systems (DBMS).

SQL is a declarative language. Programs are specified in terms of what they must do. This is a direct contrast to imperative programming, where programs specify how the computation should be performed by defining a sequence of steps that must be followed.

SQL commands are broken down into five groups:

- **Data definition language (DDL)** provides commands that allow you to define and modify a database and its components (e.g. tables, relationships), such as SQL CREATE statements.
- **Data query language (DQL)** provides commands that allow you to retrieve data from a database, such as SQL SELECT statements.
- **Data manipulation language (DML)** provides commands that allow you to insert, update, and delete data from a database, such as SQL INSERT INTO, UPDATE and DELETE FROM statements.
- **Data control language (DCL)** provides commands that allow you to grant and revoke access permissions for a database and its contents.
- **Data transaction language (DTL)** provides commands that allow you to manage transactions.

SQL statements are made up of key words (such as CREATE, SELECT, INSERT, UPDATE). These are reserved words so must not be used for identifiers (e.g. names of tables or attributes).

It is conventional to write SQL statements with the key words in capital letters. This helps with readability. However, SQL is **not** case sensitive.

Each SQL statement is terminated by a semicolon. Depending on the environment in which you execute the statement, this may or may not need to be specified. However, it is good practice to always include it.

It is conventional to write longer SQL statements over multiple lines. Again, this can help with readability.

Recall that a database is a set of data stored on one or more computers or servers.

Take as an example, a librarian has keyed in the book and borrower details given in the previous lessons.

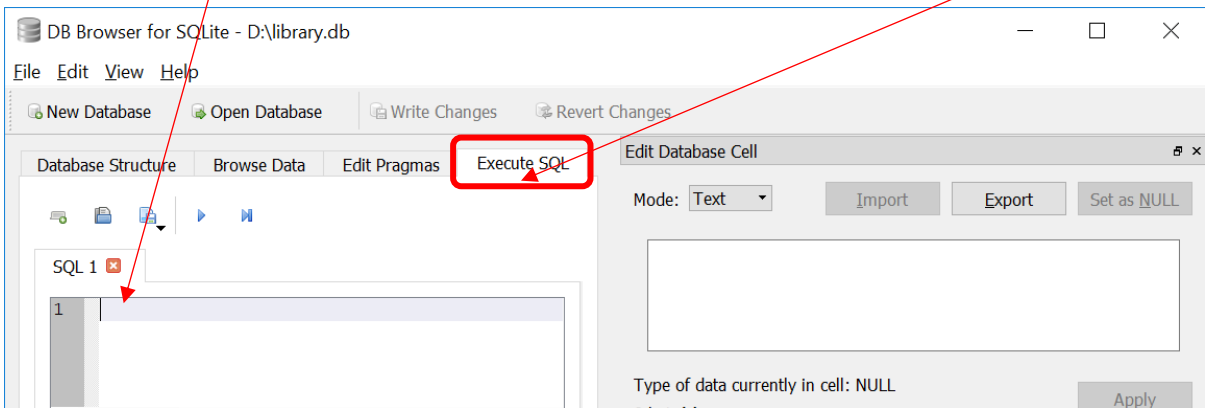
The book details are stored in Book table, with the publisher details stored in Publisher table, the borrower's details in Borrower table and the loans made in Loan table. The intention is to be able to retrieve, edit or delete the data easily.

This can be done through Data Manipulation Language (DML) statements like SELECT, INSERT, UPDATE and DELETE.

In this lesson, we will learn the SQL syntax to run the various queries in the database.

Entering SQL into DB Browser for SQLite

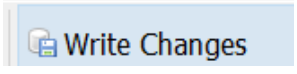
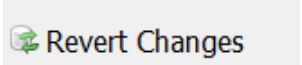
To enter SQL into DB Browser, after loading the database, click on the **Execute SQL** tab. There is a text area for you to type in your SQL commands.



You can also click on the buttons above the text area.

	Creates a new tab for entering SQL.
	Loads SQL file
	Saves SQL entered as a text file (You may save it as .sql)
	Execute all SQL statements in that tab.
	Execute the current line only.

After executing SQL commands or making changes, you can click the following:

	Save changes made into the file
	Undo changes made to the database, i.e. reload from file.

Let us now look at the SQL statements that you can enter.

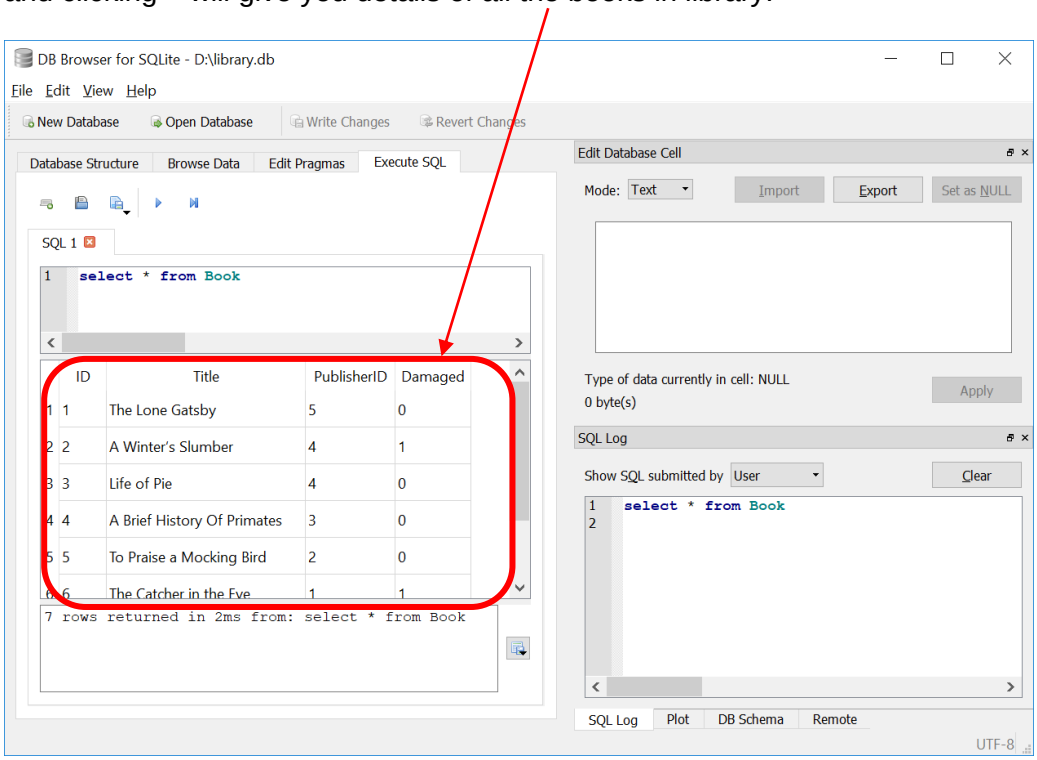
The database used in the following statements uses library.db.
You can load the database into SQLite.

SELECT

The SELECT statement allows the user to retrieve data from the database.

To select all fields, use `*`.

For example, typing
`SELECT * FROM Book`
and clicking ▶ will give you details of all the books in library.



Conditions may be added using WHERE.

For example, `SELECT * from Book WHERE Damaged = 1`
The statement will return all the damaged books.

You can also find NULL values, for example books with no publishers (no value on the PublisherID field), using the IS operator.
`SELECT * from Book WHERE PublisherID IS NULL`

You can also search for terms which are not NULL, for example:
`SELECT * from Book WHERE PublisherID IS NOT NULL`
This statement returns all records where there is a publisher.

You can insert more than one condition using AND and OR binary operators.
For example,
`SELECT * FROM Book WHERE Title = 'Life of Pie' AND Damaged = 0`
This statement returns the books with Title 'Life of Pie' **and** are not damaged.

What if you only need the book titles?
In that case, include the fields you want in the SELECT statement. For example,
`SELECT Title FROM Book`

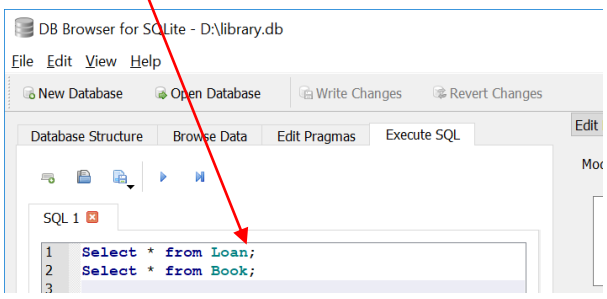
This statement will give you all the Title of the books in the Book table.

Title
The Lone Gatsby
A Winter's Slumber
Life of Pie
A Brief History Of Primates
To Praise a Mocking Bird
The Catcher in the Eye
H2 Computing Ten Year Series

Try out the SQL statements yourself!

Do you know?

You can execute multiple SQL statements, but each statement must end with a colon ;



Quiz

1. Which of the following SQL statements show all data inside the Publisher table? (Tick the statements)

	`Select all FROM Publisher`
	`Select * FROM Publisher`
	`Select ID, Title FROM Publisher`
	`Select ID, Title, PublisherID, Damaged FROM Publisher`

2. Write down the SQL statement to show all the Names of the publishers inside the Publisher table.

   .....

3. What does the following SQL statement do?
   `SELECT Title FROM Book WHERE PublisherID=1 AND Damaged=0`

   .....

   .....

4. Write down the SQL statement to show all the Titles of the books which have PublisherID 1 or 2.

   .....

   .....

Another example is to look at all the loans.

To list all loans, the SQL statement `SELECT * FROM Loan` is used.

ID	BorrowerID	BookID	Date Borrowed
1	3	2	20180220
2	3	1	20171215
3	2	3	20171231
4	1	5	20180111

The list can be ordered by the BookID (in **asc**ending order) instead by using the SQL statement `SELECT * FROM Loan ORDER BY BookID ASC`

The result is as follows.

ID	BorrowerID	BookID	Date Borrowed
2	3	1	20171215
1	3	2	20180220
3	2	3	20171231
4	1	5	20180111

The list can be ordered by the BookID (in **desc**ending order) by using the SQL statement `SELECT * FROM Loan ORDER BY BorrowerID DESC`

The result is as follows.

ID	BorrowerID	BookID	Date Borrowed
4	1	5	20180111
3	2	3	20171231
1	3	2	20180220
2	3	1	20171215

What happens if the following SQL statement is executed?
`SELECT * FROM Loan ORDER BY BookID`

By default, it will order by BookID in ascending order.
Try out the SQL statements using DBViewer in SQLite.

Quiz

1. Write down the SQL statement to select all the records in Loan table arranged in ascending order of BookID with BorrowerID = 1.

……

What if you want to find out the total number of loans?

You can use function COUNT to get the answer. The SQL can be the following:
```
SELECT COUNT(*) FROM Loan
```

SQL JOIN

A SQL JOIN allows the combination of data from two sets of data (i.e. two tables). There are different types of joins. We look at three: **cross join**, **inner join** and **left outer join**.

Cross join returns the Cartesian product of rows from the tables in the join. It combines each row in the first table with each row in the second table.

For example, the following SQL statement
```
SELECT * FROM Book, Publisher
```
produces the following table.

ID	Title	PublisherID	Damaged	ID	Name
1	The Lone Gatsby	5	0	1	NPH
1	The Lone Gatsby	5	0	2	Unpop
1	The Lone Gatsby	5	0	3	Appleson
1	The Lone Gatsby	5	0	4	Squirrel
1	The Lone Gatsby	5	0	5	Yellow Flame
2	A Winter's Slumber	4	1	1	NPH
2	A Winter's Slumber	4	1	2	Unpop
2	A Winter's Slumber	4	1	3	Appleson
2	A Winter's Slumber	4	1	4	Squirrel
2	A Winter's Slumber	4	1	5	Yellow Flame
3	Life of Pie	4	0	1	NPH
3	Life of Pie	4	0	2	Unpop
3	Life of Pie	4	0	3	Appleson
3	Life of Pie	4	0	4	Squirrel
3	Life of Pie	4	0	5	Yellow Flame
4	A Brief History Of Primates	3	0	1	NPH
4	A Brief History Of Primates	3	0	2	Unpop
4	A Brief History Of Primates	3	0	3	Appleson
4	A Brief History Of Primates	3	0	4	Squirrel
4	A Brief History Of Primates	3	0	5	Yellow Flame
5	To Praise a Mocking Bird	2	0	1	NPH
5	To Praise a Mocking Bird	2	0	2	Unpop
5	To Praise a Mocking Bird	2	0	3	Appleson
5	To Praise a Mocking Bird	2	0	4	Squirrel
5	To Praise a Mocking Bird	2	0	5	Yellow Flame
6	The Catcher in the Eye	1	1	1	NPH
6	The Catcher in the Eye	1	1	2	Unpop
6	The Catcher in the Eye	1	1	3	Appleson
6	The Catcher in the Eye	1	1	4	Squirrel
6	The Catcher in the Eye	1	1	5	Yellow Flame

123	H2 Computing Ten Year Series	NULL	0	1	NPH
123	H2 Computing Ten Year Series	NULL	0	2	Unpop
123	H2 Computing Ten Year Series	NULL	0	3	Appleson
123	H2 Computing Ten Year Series	NULL	0	4	Squirrel
123	H2 Computing Ten Year Series	NULL	0	5	Yellow Flame

Notice that there are two columns of ID. The first ID column is from Book table, while the second ID column is from the Publisher table.

Think:
How can the attribute names in Book and Publisher be renamed to remove the confusion over the ID columns?

The table above is a big table, with a lot of necessary rows. It is more useful to have the publisher of the title of each book. To do that, we can add a condition.

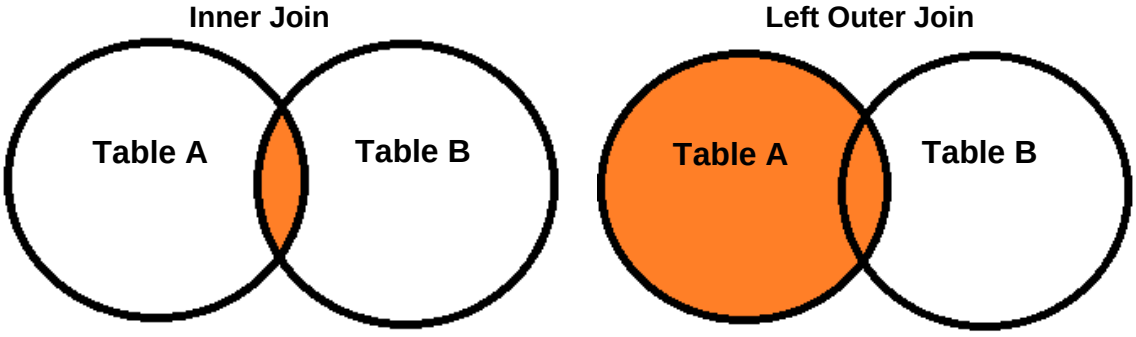
```
SELECT * FROM Book, Publisher
WHERE Book.PublisherID = Publisher.ID
```

The SQL statement above gives the following results:

ID	Title	PublisherID	Damaged	ID	Name
1	The Lone Gatsby	5	0	5	Yellow Flame
2	A Winter's Slumber	4	1	4	Squirrel
3	Life of Pie	4	0	4	Squirrel
4	A Brief History Of Primates	3	0	3	Appleson
5	To Praise a Mocking Bird	2	0	2	Unpop
6	The Catcher in the Eye	1	1	1	NPH

This table is more meaningful as it links the titles to the publishers. However, notice that the title ‘H2 Computing Ten Year Series’ is not included as it has no PublisherID.

This is an example of inner join. Inner join can also be explicitly defined, like left outer join, which we will learn next.



Inner join selects all records from Table A and Table B which meet the join condition.
Left outer join selects all records from Table A, along with records from Table B which meet the join condition.

```
SELECT * FROM Book
INNER JOIN Publisher
ON Book.PublisherID = Publisher.ID
```

For example, the SQL statement above gives the following results:

ID	Title	PublisherID	Damaged	ID	Name
1	The Lone Gatsby	5	0	5	Yellow Flame
2	A Winter's Slumber	4	1	4	Squirrel
3	Life of Pie	4	0	4	Squirrel
4	A Brief History Of Primates	3	0	3	Appleson
5	To Praise a Mocking Bird	2	0	2	Unpop
6	The Catcher in the Eye	1	1	1	NPH

The results include all books who have publishers.
The title ‘H2 Computing Ten Year Series’ is not included as it has no publisher.

Notice that there are two columns of ID. The first ID column is from Book table, while the second ID column is from the Publisher table.

However, if the inner join is changed to left outer join as below:

```
SELECT * FROM Book
LEFT OUTER JOIN Publisher
ON Book.PublisherID = Publisher.ID
```

The results are different. This is because all records in Book are included in the left outer join, even books who have no publishers, unlike the inner join.

ID	Title	PublisherID	Damaged	ID	Name
1	The Lone Gatsby	5	0	5	Yellow Flame
2	A Winter's Slumber	4	1	4	Squirrel
3	Life of Pie	4	0	4	Squirrel
4	A Brief History Of Primates	3	0	3	Appleson
5	To Praise a Mocking Bird	2	0	2	Unpop
6	The Catcher in the Eye	1	1	1	NPH
123	H2 Computing Ten Year Series	NULL	0	NULL	NULL

Quiz

1. Write down the SQL statement to join all the books that are not damaged with their Publishers.

   ……

   ……

UPDATE

The UPDATE command allows the editing of data values in a database. One or more records may be updated at the same time.

For example, the statement

```
UPDATE Book Set Title = 'Book: ' || Title
```

will update the values of Title in the Book table such that each book title now start with 'Book: '. Note the use of `||` for string concatenation (adding of two strings).

Consider another example. All the books borrowed by Kumara (BorrowerID 3) should have been borrowed by Sarah (BorrowerID 2). Then, the following SQL statement can be carried out to update the Loan table.

```
UPDATE Loan Set BorrowerID = 2 WHERE BorrowerID = 3
```

Quiz

1. All the books borrowed by Sarah (BorrowerID 2) should have been borrowed by Kumara (BorrowerID 3). Write down the SQL statement to update the Loan table.

   ……

   ……

INSERT INTO

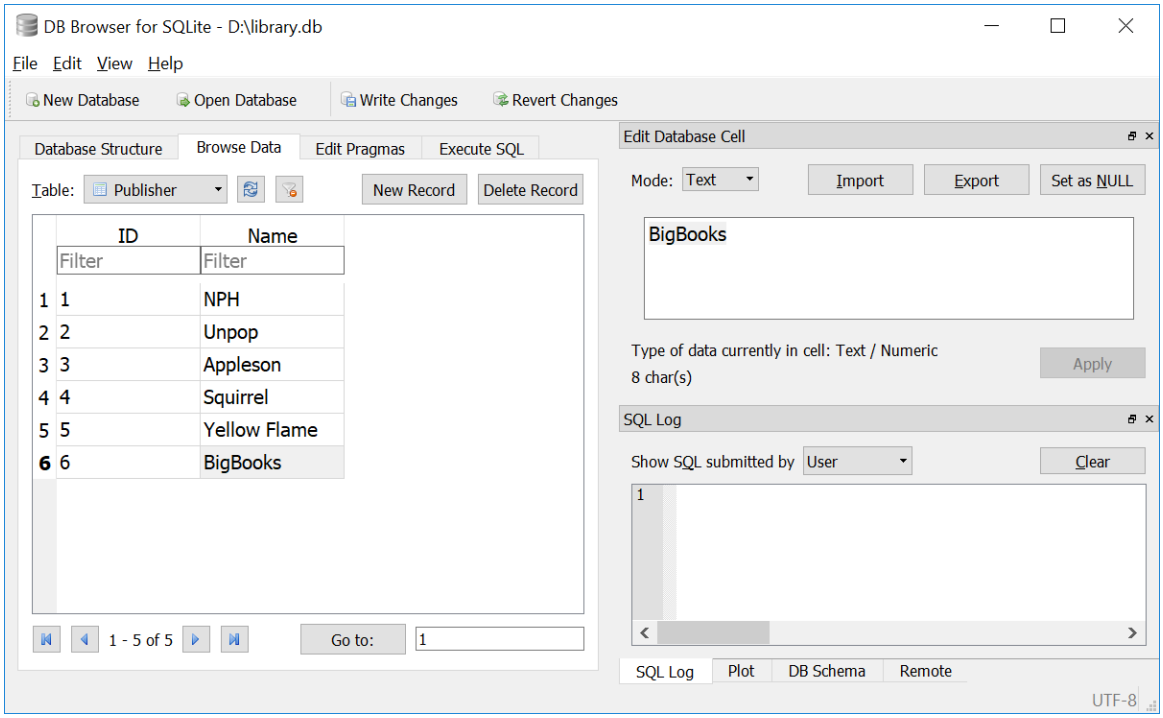
The INSERT INTO command is used to insert a new record in a table.

For example, to insert a new publisher, we can use the following SQL:

```
INSERT INTO Publisher(Name) VALUES ('BigBooks')
```

This will insert the publisher 'BigBooks' from the Publisher table.

You can check by selecting Publisher table under 'Browse Data' tab.



Notice that the ID is automatically given! Do you know how the database does it?
(Hint: Answer is found later…)

DELETE

The DELETE command is used to delete existing records in a table.

For example,

```
DELETE FROM Books WHERE Title = 'Life of Pie'
```

This will delete the book 'Life of Pie' from the Books table.
Please note that the text is case-sensitive, which means that the following 2 SQL statements are different.

```
DELETE FROM Book WHERE Title =  'Life of Pie'
DELETE FROM Book WHERE Title =  'Life of pie'.
```

Be careful when using the DELETE statement. For example:

```
DELETE FROM Loan
```

This will delete all entries from the Loans table!

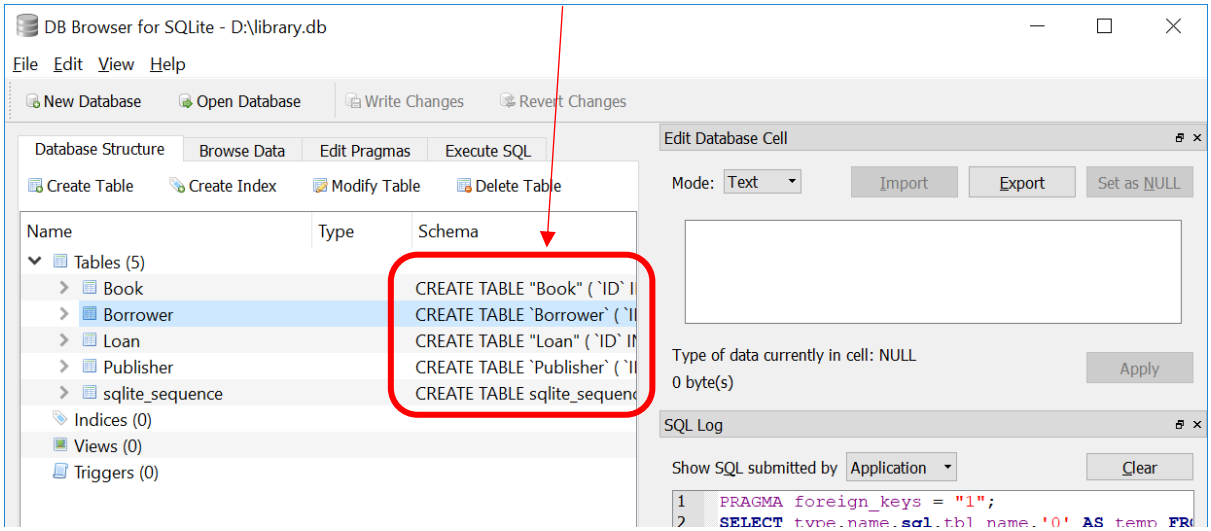
Quiz

1. Which of the following SQL statements are valid? (Tick the statements)

	`DELETE FROM Book WHERE Title = 'H2 Computing Ten Year Series'`
	`DELETE FROM Book WHERE Title = *`
	`DELETE FROM Book WHERE`
	`DELETE FROM Book`

CREATE TABLE

The CREATE TABLE command allows you to a table. You can see the SQL code for the various tables under Database Structure in DB Browser.



For example, to create the Borrower table, you can key in:

```
CREATE TABLE 'Borrower' (
	'ID'  INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
	'FirstName' TEXT NOT NULL,
	'Surname'  TEXT NOT NULL,
	'Contact'  INTEGER NOT NULL
)
```

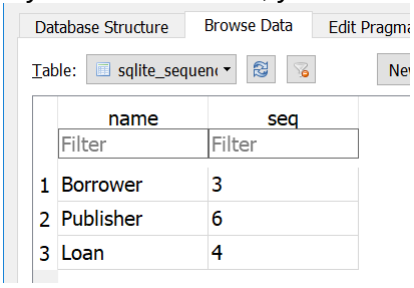
Let's look at the SQL statement carefully.
INTEGER and TEXT are data types, 'ID', 'FirstName', 'Surname' and 'Contact' are field names. The ID field is the primary key of the table Borrower, while Autoincrement means the ID value is automatically given by the database. NOT NULL means that the fields do not accept Null values.

Thus, the syntax for creating tables in SQL is

CREATE TABLE *table_name*(
	column1_name COLUMN1_TYPE COLUMN1_CONSTRAINTS,
	column2_name COLUMN2_TYPE COLUMN2_CONSTRAINTS,
…
	PRIMARY KEY (*column1_name, column2_name,…*),
	FOREIGN KEY (*column_name*) **REFERENCES** *table_name*(*column_name*)
)

Also, note that there is an additional table sqlite_sequence above.

If you view the table, you will see the following.



This table is used by SQLite so as to keep track of the next number to give for tables with AUTOINCREMENT. It is generated automatically when you have an AUTOINCREMENT field used in SQLite.

Let us look at the Book table next. The SQL code to create the table is:

```
CREATE TABLE 'Book' (
      'ID' INTEGER NOT NULL,
      'Title' TEXT NOT NULL,
      'PublisherID' INTEGER,
      'Damaged'  INTEGER NOT NULL,
      FOREIGN KEY('PublisherID') REFERENCES 'Publisher'('ID'),
      PRIMARY KEY('ID')
)
```

Notice the line starting with FOREIGN KEY. It defines the foreign key, which is the field linking to the primary key of another table. For this example, it is linking the PublisherID field in the Book table to the ID field in the Publisher table.

Quiz

5. Key in SQL statement to create a table Testing with fields name, tag_no and remarks. The fields name and remarks should accept text, while tag_no is an integer automatically incremented. The primary key is tag_no. Name field should not be NULL.

   ……

   ……

   ……

   ……

Tip!
You can click 'Revert Changes' in DB Browser to remove the recent changes made.

DROP TABLE

The DROP TABLE command deletes the entire table.
For example, to remove the Loan table, you can key in:
`DROP TABLE Loan`

What is the difference between `DELETE FROM Loan` and `DROP TABLE Loan`?

With `DELETE FROM Loan`, you delete all entries from the Loan table, but the Loan table remains there. With `DROP TABLE Loan`, the Loan table will be removed. You cannot insert any entries into Loan table anymore.

Quiz

6. Try keying `DROP TABLE Publisher` to remove the table containing the publishers. Is it possible? Why?

   ……

   ……

Operators

You have seen some operators being used in the examples earlier. These operators are often used in SELECT statements, but can be used in other statements (like the UPDATE statement example shown earlier). The following are comparison operators, logical operators and arithmetic operators that you need to know.

A. Comparison Operators

Comparison Operator	Description
=	Checks if the values of two operands are equal or not, if yes then the condition becomes true.
!=	Checks if the values of two operands are equal or not, if the values are not equal, then the condition becomes true.
<>	Checks if the values of two operands are equal or not, if the values are not equal, then the condition becomes true.
>	Checks if the values of the left operand is greater than the value of the right operand, if yes then the condition becomes true.
<	Checks if the values of the left operand is less than the value of the right operand, if yes then the condition becomes true.
>=	Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes then the condition becomes true.
<=	Checks if the value of the left operand is less than or equal to the value of the right operand, if yes then the condition becomes true.

B. Logical Operators

Logical Operator	Description
AND	The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.
OR	The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.
IS	The value exists. For example, IS NULL looks for NULL values.
IS NOT	The value does not exist.
||	String concatenation

C. Arithmetic Operators

Arithmetic Operator	Name	Description
+	Addition	Adds values on either side of the operator
-	Subtraction	Subtracts the right-hand operand from the left-hand operand
*****	Multiplication	Multiplies values on either side of the operator
/	Division	Divides the left-hand operand by the right-hand operand
%	Modulus	Divides the left-hand operand by the right-hand operand and returns the remainder

Functions

Aggregate functions help to count / calculate results from the database.

Function	Description
MIN	Minimum value
MAX	Maximum value
SUM	Sum of all values
COUNT	Number of values

Exercise

Open the file poly.db.
It contains information from Temasek Polytechnic Full-Time Enrolment Figures Breakdown, Annual dataset accessed on 23 March 2018 from Temasek Polytechnic which is made available under the terms of the Singapore Open Data Licence version 1.0 (http://www.data.gov.sg)
The database provides the enrolment in different polytechnic courses in Temasek Polytechnic (TP) by year and gender.

Try to give the SQL statement to generate the following.

1. List the courses in TP that has more than 250 males in 2017.

   ……

   ……

2. List the courses in TP that has no more than 50 females in 2016.

   ……

   ……

3. List out all fields in the table for courses not in School of Informatics & IT in 2016.

   ……

   ……

4. List the largest number of female(s) in a course in 2017.

   ……

   ……

5. List the smallest number of male(s) in a course in 2016.

   ……

   ……

6. Rename all the diplomas called ‘Diploma in Cyber & Digital Security’ to ‘Diploma in Cyber and Digital Security’.

   ……

   ……

7. Update the table to add 5 students to each gender and each course in School of Design in 2017.

……

……