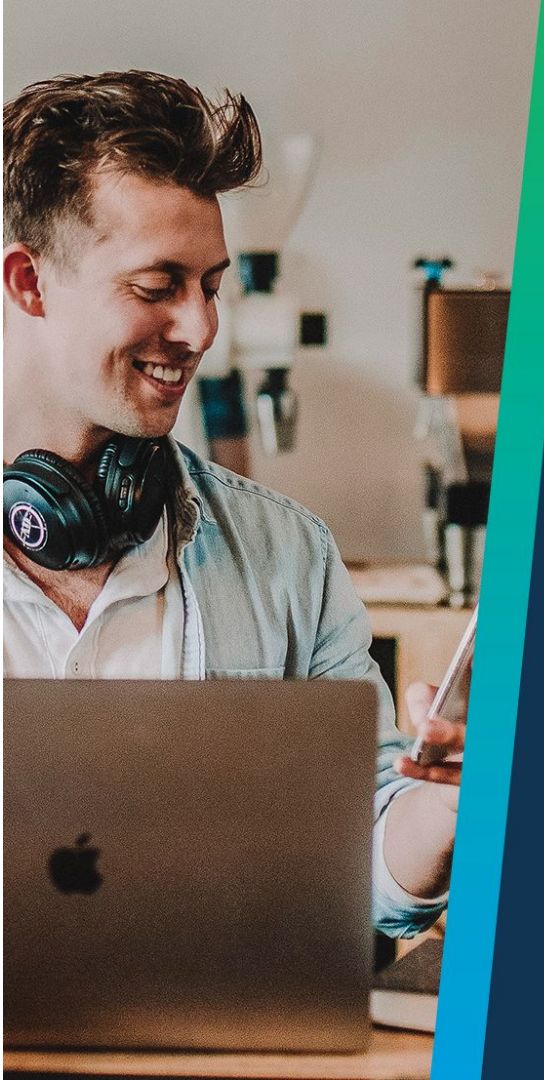


Data Science
Bootcamp

Hyperiondev

SQL Workshop

Lecture – Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
- No question is daft or silly - **ask them!**
- There are Q/A sessions midway and at the end of the session, should you wish to ask any follow-up questions.
- You can also submit questions here: http://hyperiondev.com/sbc4-se-questions
- For all non-academic questions, please submit a query: www.hyperiondev.com/support
- Report a safeguarding incident: http://hyperiondev.com/safeguardreporting
- We would love your feedback on lectures: https://hyperionde.wufoo.com/forms/zsgv4m40ui4i0g/

Objectives

1. What is SQL?
2. Why SQL?
3. SQL keywords
4. Creating and modifying tables
5. Retrieving data from tables
6. Deleting rows and tables

Github Repository - Lecture Examples

https://github.com/HyperionDevBootcamps/C4_SE_lecture_examples

Resources

https://opentextbc.ca/dbdesign01/

https://opentextbc.ca/dbdesign01

https://www.programiz.com/sql/online-compiler/

What is SQL?

It stands for Structured Query Language (SQL)

SQL is a database language that is composed of commands that enable users to create databases or table structures, perform various types of data manipulation and data administration as well as query the database to extract useful information.

Why SQL?

SQL is easy to learn since its vocabulary is relatively simple. Its basic command set has a vocabulary of fewer than 100 words. It is also a non-procedural language,which means that the user specifies what must be done and not how it should be done.

Users do not need to know the physical data storage format or the complex activities that take place when a SQL command is executed in order to issue a command.

Important Keywords

- **CREATE TABLE**: Creates a new table
- **NOT NULL**: Ensures that a column doesn't contain null values
- **UNIQUE**: Ensures that there are no repetitions
- **PRIMARY KEY**: Defines a primary key
- **DROP TABLE**: Deletes a table entirely

Important Keywords

- **INSERT**: Inserts rows into a table
- **SELECT**: Select attributes from a row
- **WHERE**: Restricts the selection of rows based on a conditional expression.
- **UPDATE**: Modifies an attribute's values in a table
- **ORDER BY**: Orders the selected rows by a specific column. Can specify ASCENDING or DESCENDING
- **DELETE**: Deletes one or more rows from a table

Special Operators

- These keywords are used in conditions.
- **BETWEEN**: Checks if a value is within range
- **IS NULL**: Checks if a value is null
- **LIKE**: Checks if a string matches a given pattern (regular expression)
- **IN**: Checks if a value is in a given list
- **EXISTS**: Checks if a query returns any rows
- **DISTINCT**: Limits to unique values

Creating Tables

To create new tables in SQL, you use the CREATE TABLE statement. Pass all the columns you want in the table, as well as their data types, as arguments to the CREATE TABLE function.

The syntax of the CREATE TABLE statement is shown below:

```
CREATE TABLE table_name (
        column1_name  datatype  constraint,
        column2_name  datatype  constraint,
        column3_name  datatype  constraint,
        ...
);
```

Example

To create a table called Employee, for example, that contains five columns (EmployeeID, LastName, FirstName, Address, and PhoneNumber), you would use the following SQL:

```
CREATE TABLE Employee (
    EmployeeID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Address varchar(255),
    PhoneNumber varchar(255),
);
```

The CREATE TABLE statement above will create an empty Employee table that will look like this:

EmployeeID	LastName	FirstName	Address	PhoneNumber

Inserting Values

There are two ways to write the INSERT INTO command:

1. You do not have to specify the column names where you intend to insert the data. This can be done if you are adding values for all of the columns of the table. However, you should ensure that the order of the values is in the same order as the columns in the table. The syntax will be as follows:

```
INSERT INTO table_name
        VALUES (value1, value2, value3,...);
```

2. The other way to insert data into a table is to specify both the column names and the values to be inserted. The syntax will be as follows:

```
INSERT INTO Employee (column1, column2, column3, ...)
        VALUES (value1, value2, value3,...);
```

Inserting Values

```
INSERT INTO Employee
        VALUES (1, 'Smith', 'John', '25 Oak Rd', '0837856767');
```

OR

```
INSERT INTO Employee (EmployeeID, LastName, FirstName, Address, PhoneNumber)
        VALUES (1234, 'Smith', 'John', '25 Oak Rd', '0837856767');
```

Retrieving Data

The SELECT statement is used to fetch data from a database. The data returned is stored in a result table, known as the result-set. The syntax of a SELECT statement is as follows:

```
SELECT column1, column2,...
FROM table_name;
```

```
SELECT FirtsName, LastName,...
FROM Employee;
```

If you want to select all the columns in the table, you use the following syntax:

```
SELECT * FROM table_name
```

Ordering Values

You can use the ORDER BY command to sort the results returned in ascending or descending order. The ORDER BY command sorts the records in ascending order by default. You need to use the DESC keyword to sort the records in descending order.

```
SELECT * FROM table_name
ORDER BY column1 DESC;
```

```
SELECT * FROM Employee
ORDER BY FirstName DESC;
```

Using WHERE, IN and BETWEEN keywords

```
SELECT * FROM Employee
WHERE FirstName='John';
```

```
SELECT * FROM albums
WHERE genre IN ('pop', 'soul');
```

```
SELECT * FROM albums
WHERE released BETWEEN 1975 AND 1985;
```

Updating Values

The UPDATE statement is used to modify the existing rows in a table.

To use the UPDATE statement you:

- Choose the table where the row you want to change is located.
- Set the new value(s) for the desired column(s).
- Select which of the rows you want to update using the WHERE statement. If you omit this, all rows in the table will change.

```
UPDATE table_name
SET column1 = value1,  column2 = value2, ...
WHERE condition;
```

Updating Values

CustomerID	CustomerName	Address	City
1	Maria Anderson	23 York St	New York
2	Jackson Peters	124 River Rd	Berlin
3	Thomas Hardy	455 Hanover Sq	London
4	Kelly Martins	55 Loop St	Cape Town

```
UPDATE Customer
SET Address = '78 Oak St', City= 'Los Angeles'
WHERE CustomerID = 1;
```

Deleting rows

Deleting a row is a simple process. All you need to do is select the right table androw that you want to remove. The DELETE statement is used to delete existing rows in a table.

```
DELETE FROM table_name
WHERE condition;
```

```
DELETE FROM Customer
WHERE CustomerID = 4;
```

Deleting tables

You can also delete all rows in a table without deleting the table:

```
DELETE * FROM table_name;
```

The DROP TABLE statement is used to remove every trace of a table in a database.

```
DROP TABLE table_name;
```

SQL vs SQLite

- SQLite is a form of SQL.
- SQL can be expressed in many ways:
  - SQLite
  - MySQL
  - PostgreSQL
- “UK” English vs. “US” English

SQLite

- Native to Python (Yay! No pip installations!)
- Self-Contained
  - Easy to port
- Serverless
  - Doesn’t require client-server architecture. Works directly with files.
- Transactional
  - Atomic, Consistent, Isolated and Durable (ACID). Ensures data integrity

Syntax

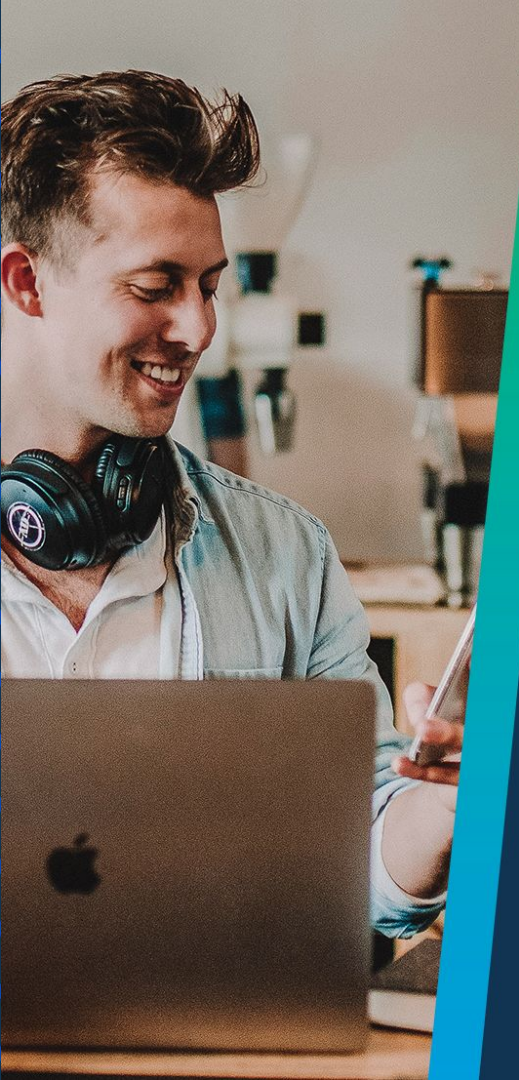
```
import sqlite3
db = sqlite3.connect('data/student_db')
cursor = db.cursor()
cursor.execute('''
    CREATE TABLE student(id INTEGER PRIMARY KEY, name TEXT,
                grade INTEGER)
''')
db.commit()
cursor.execute('''INSERT INTO student(name, grade)
                VALUES(?,?)''', (name1,grade1))
db.commit()
students_ = [(name1,grade1),(name2,grade2),(name3,grade3)]

cursor.executemany(''' INSERT INTO student(name, grade) VALUES(?,?)''', students_)
```

Hyperiondev

Q & A Section

Please use this time to ask any questions relating to the topic explained, should you have any



Hyperiondev

Thank you for joining us