```
# Here I'll show you how to work with SQLite databases
# in Python

# A database makes it easy for you to organize your
# data for storage and fast searching

# I show how to install SQLite and use it in a previous video

# You need the SQLite module to use it
import sqlite3
import sys
import csv

# connect() will open an SQLite database, or if it
# doesn't exist it will create it
# The file appears in the same directory as this
# Python file
db_conn = sqlite3.connect('test.db')
print("Database Created")

# A cursor is used to traverse the records of a result
the_cursor = db_conn.cursor()

def print_db():
    # To retrieve data from a table use SELECT followed
    # by the items to retrieve and the table to
    # retrieve from
    try:
        result = the_cursor.execute("SELECT id, f_name, l_name, age, address, salary, hire_date
FROM employees")

        # You receive a list of lists that hold the result
        for row in result:
            print("id :", row[0])
            print("f_name :", row[1])
            print("l_name :", row[2])
            print("age :", row[3])
            print("address :", row[4])
            print("salary :", row[5])
            print("hire_date :", row[6])
    except sqlite3.OperationalError:
        print("The table doesn't exist")
    except:
        print("Couldn't retrieve data from database")


# execute() executes a SQL command
# We organize our data in tables by defining their
# name and the data type for the data

# We define the table name
# A primary key is a unique value that differentiates
# each row of data in our table
# The primary key will auto increment each time we
```

```
# add a new Employee
# If a piece of data is marked as NOT NULL, that means
# it must have a value to be valid

# NULL is NULL and stands in for no value
# INTEGER is an integer
# TEXT is a string of variable length
# REAL is a float
# BLOB is used to store binary data

# You can delete a table if it exists like this
# db_conn.execute("DROP TABLE IF EXISTS Employees")
# db_conn.commit()
try:
    db_conn.execute("CREATE TABLE employees(id INTEGER PRIMARY KEY
AUTOINCREMENT NOT NULL, f_name TEXT NOT NULL, l_name TEXT NOT NULL, age INT
NOT NULL, address TEXT, salary REAL, hire_date TEXT);")
    db_conn.commit()
    print("Table Created")
except sqlite3.OperationalError as e:
    print("Table couldn't be created :", str(e))

# To insert data into a table we use INSERT INTO
# followed by the table name and the item name
# and the data to assign to those items
db_conn.execute("INSERT INTO employees(f_name, l_name, age, address, salary, hire_date)
VALUES ('Derek', 'Banas', 43, '123 Main St', 500000, date('now'));")
db_conn.commit()
print("Employee Entered")

# Print out all the data in the database
print_db()

# You can update a value in a table by referencing
# something unique like the ID or anything else
# with the UPDATE command
try:
    db_conn.execute("UPDATE employees SET address = '121 Main St' WHERE ID = 1")
    db_conn.commit()
except sqlite3.OperationalError:
    print("Database couldn't be updated")

print_db()

# Delete matching data from the database by
# referencing the table name and something unique
try:
    db_conn.execute("DELETE FROM employees WHERE ID = 1")
    db_conn.commit()
except sqlite3.OperationalError:
    print("Data couldn't be deleted")

print_db()
```

```
# Undo the last commit()
db_conn.rollback()

print_db()

# You can add a new column to a table with ALTER
try:
    db_conn.execute("ALTER TABLE employees ADD COLUMN 'image' BLOB DEFAULT NULL")
    db_conn.commit()
except sqlite3.OperationalError:
    print("Table couldn't be altered")

# Retrieve table column names
the_cursor.execute("PRAGMA TABLE_INFO(employees)")

# fetchall() returns all remaining rows of a query result
# as a list
row_names = [nameTuple[1] for nameTuple in the_cursor.fetchall()]
print(row_names)

# Get the total number of rows
the_cursor.execute('SELECT COUNT(*) FROM employees')
num_of_rows = the_cursor.fetchall()
print("Total Rows :", num_of_rows[0][0])

# Get SQLite version
the_cursor.execute("SELECT SQLITE_VERSION()")

# fetchone() returns one result
print("SQLITE VERSION :", the_cursor.fetchone())

# Use the dictionary cursor to retrieve data in a dictionary
with db_conn:
    db_conn.row_factory = sqlite3.Row
    the_cursor = db_conn.cursor()
    the_cursor.execute("SELECT * FROM employees")
    rows = the_cursor.fetchall()
    for row in rows:
        print("{} {}".format(row["f_name"], row["l_name"]))

# Write data to File
with open('dump.sql', 'w') as f:
    # iterdump() returns an iterator to dump the database
    # in SQL format

    for line in db_conn.iterdump():
        f.write("%s\n" % line)

# Close the database connection
db_conn.close()
```