Database SELECT and WHERE Clause with sqlite3

Why should scientists care?

- Being able to select only certain records will speed up and clean up the retreval of information
- SELECT and WHERE are fundamental operations that will be used whenever you work with databases

In [1]:
```
# sqlite3 is a built-in python module for creating and using sql
# It does not require a separate server
import sqlite3
```

Setting up: Creating tables and inserting rows

In [20]:
```
# Create a database and a connection to it
conn = sqlite3.connect('/tmp/dbexample')

# A cursor is used to execute SQL statements in your database
c = conn.cursor()

# Create table
c.execute("create table Experiments (Scientist text, Project tex
```

Out[20]:
```
<sqlite3.Cursor at 0x2d80420>
```

In [22]:
```
### Insert rows into the table

## Make a list of tuples that will be inserted
rows_to_insert = [('Sofia Kovalevskaya', 'Antigravity', 6.5),
    ('Sofia Kovalevskaya', 'Teleportation', 11.0),
    ('Sofia Kovalevskaya', 'Teleportation', 5.0),
    ('Mikhail Lomonosov', 'Antigravity', 4.0),
    ('Mikhail Lomonosov', 'Time Travel', -2.0),
    ('Dmitri Mendeleev', 'Antigravity', 9.0),
    ('Ivan Pavlov', 'Teleportation', 9.0),
    ('Ivan Pavlov', 'Time Travel', -7.0),
    ('Cheech Marin', 'Mind Travel', None)]
```

```
## Insert all rows at once
c.executemany('INSERT into Experiments values (?, ?, ?)', rows_t

## This does the same thing as inserting rows one-by-one
#for row in rows_to_insert:
#    c.execute('insert into Experiments values (?, ?, ?)', row)
```

Out[22]: <sqlite3.Cursor at 0x2d80420>

In [23]:
```
# Save (commit) the changes
conn.commit()

# We can also close the cursor and connection if we are done wit
c.close()
conn.close()
```

Executing SQL statements: SELECT and WHERE

In [24]:
```
# Open connection and cursor back up
conn = sqlite3.connect('/tmp/dbexample')
c = conn.cursor()
```

The basic SELECT statement

In [25]:
```
# Classic SQL SELECT statement (all-caps is optional)
c.execute('SELECT Scientist, Hours FROM Experiments')

# To actually get the results you can call cursor.fetchall()
print c.fetchall()
```

```
[(u'Sofia Kovalevskaya', 6.5), (u'Sofia Kovalevskaya', 11.0),
(u'Sofia Kovalevskaya', 5.0), (u'Mikhail Lomonosov', 4.0),
(u'Mikhail Lomonosov', -2.0), (u'Dmitri Mendeleev', 9.0),
(u'Ivan Pavlov', 9.0), (u'Ivan Pavlov', -7.0), (u'Cheech
Marin', None)]
```

In [26]:
```
# This clears the cursor, so call the statement again to fetch a
c.execute('SELECT Scientist, Hours FROM Experiments')

# You can use the cursor as an iterator to get each row as a tup
for row in c:
    print row
```

```
(u'Sofia Kovalevskaya', 6.5)
(u'Sofia Kovalevskaya', 11.0)
(u'Sofia Kovalevskaya', 5.0)
(u'Mikhail Lomonosov', 4.0)
(u'Mikhail Lomonosov', -2.0)
(u'Dmitri Mendeleev', 9.0)
(u'Ivan Pavlov', 9.0)
(u'Ivan Pavlov', -7.0)
(u'Cheech Marin', None)
```

The WHERE Clause

We can use many other operators to filter our data. For example, we could ask for all of the experiments that were done by Ivan Pavlov:

In [27]:
```
c.execute('SELECT * FROM Experiments WHERE Scientist = "Ivan Pav
c.fetchall()
```

Out[27]:
```
[(u'Ivan Pavlov', u'Teleportation', 9.0),
 (u'Ivan Pavlov', u'Time Travel', -7.0)]
```

We can also make our WHERE conditions more sophisticated by combining tests with AND and OR. For example, suppose we want to know which project Mikhail Lomonosov spent more than three hours working on. We're only interested in rows that satisfy both criteria, so we combine the two tests with AND:

In [40]:
```
ex_str = """
SELECT * FROM Experiments
WHERE (Hours > 3) AND (Scientist = "Mikhail Lomonosov")
"""
c.execute(ex_str)
c.fetchall()
```

Out[40]:
```
[(u'Mikhail Lomonosov', u'Antigravity', 4.0)]
```

Variations

Use AND and OR operators to get records where either

- Mikhail Lomonosov worked more than 3 hours

OR

- Ivan Pavlov worked any number of hours

In [1]:
```
ex_str = """
FILL THIS IN YOURSELF
"""
c.execute(ex_str)
c.fetchall()
```

Instead of using OR to match one of several values, we can use the IN operator along with a list of values we would like to match. Use this to obtain records where

- Either Mikhail Lomonosov or Ivan Pavlov worked more than 3 hours

In [2]:
```
ex_str = """
FILL THIS IN YOURSELF
"""
c.execute(ex_str)
c.fetchall()
```

Comparisons with NULL should be done using IS rather than =.
Write a query to find all records where someone worked no hours

In [ ]:
```
# Note that NULL is represented by Python None
c.execute('FILL IN YOURSELF')
c.fetchall()
```

Non-null entries can be found using "IS NOT NULL"
Write a query to find all records where the hours worked were not null

In [ ]:
```
c.execute('FILL IN YOURSELF')
c.fetchall()
```