


```
## 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 Pavlov"')
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

And if we wanted project that either scientist had spent more than three hours on, we would combine our tests:

```
In [37]: ex_str = """
SELECT * FROM Experiments
WHERE (Hours > 3) AND ((Scientist = "Mikhail Lomonosov")
```

```
OR (Scientist = "Ivan Pavlov"))
"""
c.execute(ex_str)
c.fetchall()
```

```
Out[37]: [(u'Mikhail Lomonosov', u'Antigravity', 4.0),
          (u'Ivan Pavlov', u'Teleportation', 9.0)]
```

The extra parentheses around the checks on the scientists' names ensure that the AND and OR are combined the way we want.

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. For example, we could rewrite our query as:

```
In [39]: ex_str = """
SELECT * FROM Experiments
WHERE (Hours > 3)
AND (Scientist IN ("Mikhail Lomonosov", "Ivan Pavlov"))
"""
c.execute(ex_str)
c.fetchall()
```

```
Out[39]: [(u'Mikhail Lomonosov', u'Antigravity', 4.0),
          (u'Ivan Pavlov', u'Teleportation', 9.0)]
```

WHERE can be used in a self join as well

```
In [31]: ## Get Scientists who have worked on two or more projects
ex_str = """
SELECT DISTINCT a.Scientist
FROM Experiments a JOIN Experiments b
WHERE (a.Scientist = b.Scientist) AND (a.Project != b.Project)
ORDER BY a.Scientist ASC
"""
c.execute(ex_str)
c.fetchall()
```

```
Out[31]: [(u'Ivan Pavlov',), (u'Mikhail Lomonosov',), (u'Sofia
          Kovalevskaya',)]
```

```
In [33]: ## To check for NULL using WHERE, use IS rather than =
# Note that NULL is represented by Python None
c.execute('SELECT * FROM Experiments WHERE Hours IS NULL')
c.fetchall()
```

```
Out[33]: [(u'Cheech Marin', u'Mind Travel', None)]
```

```
In [34]: ## To check for NULL using WHERE, use IS rather than =  
# Note that NULL is represented by Python None  
c.execute('SELECT * FROM Experiments WHERE Hours IS NOT NULL')  
c.fetchall()
```

```
Out[34]: [(u'Sofia Kovalevskaya', u'Antigravity', 6.5),  
(u'Sofia Kovalevskaya', u'Teleportation', 11.0),  
(u'Sofia Kovalevskaya', u'Teleportation', 5.0),  
(u'Mikhail Lomonosov', u'Antigravity', 4.0),  
(u'Mikhail Lomonosov', u'Time Travel', -2.0),  
(u'Dmitri Mendeleev', u'Antigravity', 9.0),  
(u'Ivan Pavlov', u'Teleportation', 9.0),  
(u'Ivan Pavlov', u'Time Travel', -7.0)]
```