csvstudio.py

Keep it simple csv analysis tool

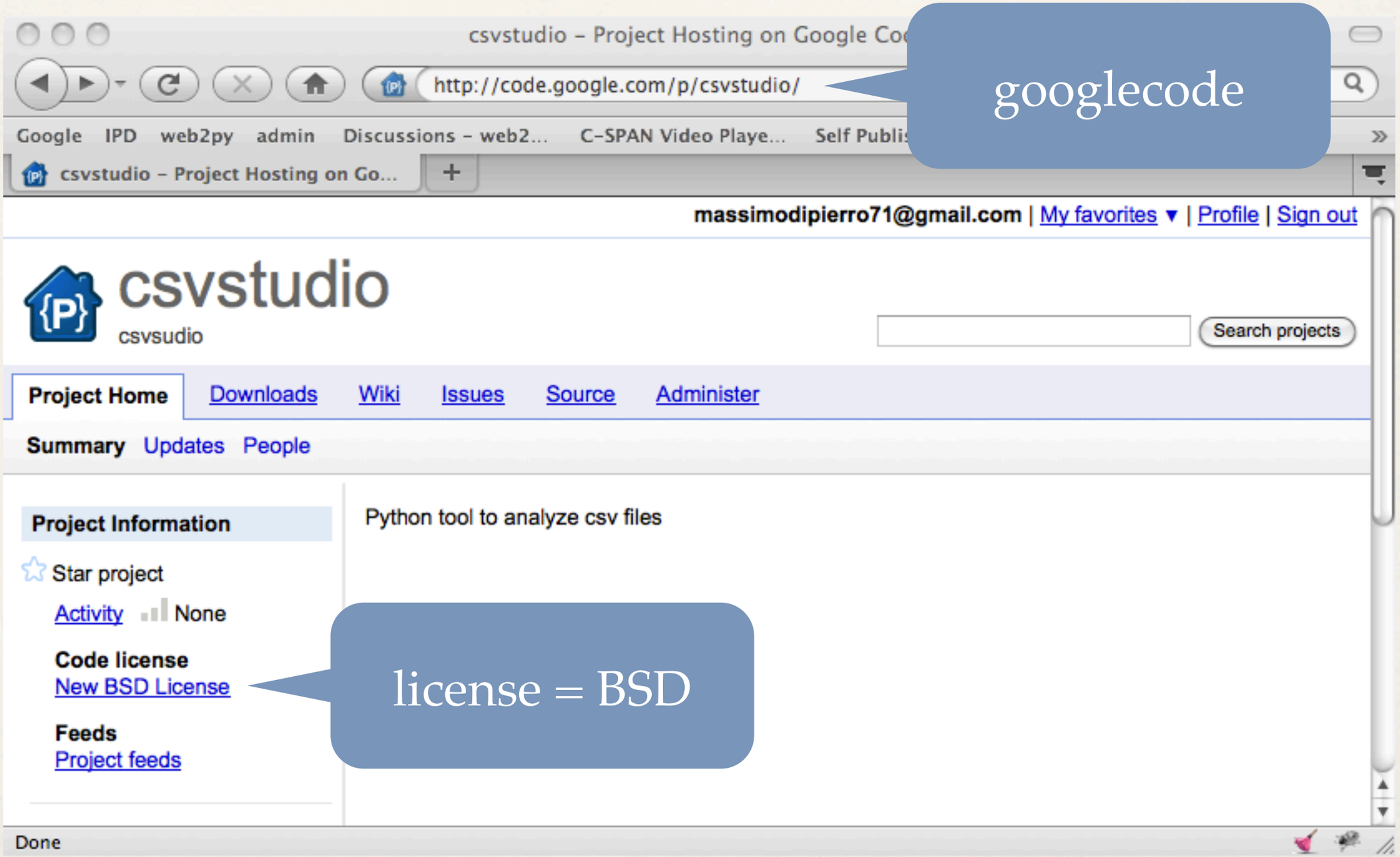
Why?

- Useful to handle large CSV files when you do not know what's inside

What does csvstudio.py do?

- Command line tool (one single file, no dependencies)
- It reads and parses CSV files
- Perform statistical analysis to discover content and dependent cols
- It can filter rows and cols using a simple but powerful syntax
- Can compute new columns, fix/normalize existing ones
- Can output text, html and other csv files
- Averything can be scripted

Where can I get it?



Show me what's in the file

```
$ python csvstudio.py -i input.csv -m 5
```

display only 5 rows

```
### 1
year=2000, code='X005', product_name='Cuckoo clock', price=114, quantity=15
### 2
year=2000, code='X005', product_name='Cuckoo clock', price=114, quantity=60
### 3
year=2000, code='X005', product_name='Cuckoo clock', price=106, quantity=38
### 4
year=2000, code='X005', product_name='Cuckoo clock', price=105, quantity=1

...

### 15000
year=2006, code='Y007', product_name='Standing clock', price=115, quantity=7
```

Do a statistical analysis

analysis

col names

values found in column year

code is equivalent to product_name

items have only 20 possible prices

```
$ python csvstudio.py -i input.csv -a

### Columns
columns = ['year', 'code', 'product_name', 'price', 'quantity']

### Column "year"
len(values) = 10
values = [2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009]

### Column "code"
len(values) = 2
values = ['X005', 'Y007']
code is equivalent to product_name

### Column "product_name"
len(values) = 2
values = ['Cuckoo clock', 'Standing clock']

### Column "price"
len(values) = 20
values = [100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115,
116, 117, 118, 119]
```

Filter rows

-q is a query

year=2003, record must contain X005

```
$ python csvstudio.py -i input.csv -q "year=2003 X005"

### 1
year=2003, code='X005', product_name='Cuck
### 2
year=2003, code='X005', product_name='Cuc
### 3
year=2003, code='X005', product_name='Cuc
### 4
year=2003, code='X005', product_name='Cuckoo clock', price=100, quantity=24

...

### 1000
year=2003, code='X005', product_name='Cuckoo clock', price=105, quantity=89
```

Filter cols

```
$ python csvstudio.py -i input.csv -q "year=2003 X005" -c price,quantity

### 1
price=105, quantity=4
### 2
price=108, quantity=16
### 3
price=105, quantity=83
### 4
price=100, quantity=24

...

### 1000
price=105, quantity=89
```

year=2003, record must contain X005

display only price and quantity

Compute

```
$ python csvstudio.py -i input.csv -q "year=2003 X005" -c price,quantity,revenue
                      -r "revenue=price*quantity" -t
```

compute a column revenue

add a row with totals

```
### 1
price=105, quantity=4, revenue=420
### 2
price=108, quantity=16, revenue=1728
### 3
price=105, quantity=83, revenue=8715
### 4
price=100, quantity=24, revenue=2400

...

### 1001
price=109360.0, quantity=51401.0, revenue=5628864.0
```

total revenues for cuckoo clocks in 2003

total number of cuckoo clocks sold in 2003

Fix/Normalize cells

```
$ python csvstudio.py -i input.csv
                      -r "product_name=product_name.strip().capitalize()"

### 1
year=2000, code='X005', product_name='Cuckoo clock', price=114, quantity=15,
product_name='Cuckoo clock'
### 2
year=2000, code='X005', product_name='Cuckoo clock', price=114, quantity=60,
product_name='Cuckoo clock'
### 3
year=2000, code='X005', product_name='Cuckoo clock', price=106, quantity=38,
product_name='Cuckoo clock'
### 4
year=2000, code='X005', product_name='Cuckoo clock', price=105, quantity=1,
product_name='Cuckoo clock'

...

### 15000
year=2006, code='Y007', product_name='Standing clock', price=115, quantity=7,
product_name='Standing clock'
```

Export

```
$ python csvstudio.py -i input.csv -q "year=2003 X005" -c price,quantity,revenue
                      -r "revenue=price*quantity" -o output.csv -f csv
```

to csv

```
$ python csvstudio.py -i input.csv -q "year=2003 X005" -c price,quantity,revenue
                      -r "revenue=price*quantity" -o output.html -f html
```

to html

```
$ python csvstudio.py -i input.csv -q "year=2003 X005" -c price,quantity,revenue
                      -r "revenue=price*quantity" -o output.text -f text
```

to text

Query Syntax (-q)

- AND conditions separated by space
- EACH condition can take the form "value" or "column$value" where $ is one of the supported operators =, <, >, <=, >=, #
- (# means "contains")
- EXAMPLE: `-q "2003 X005 price<50 product_name#clock"` means "all rows which contain 2003 and contain X005 and have a price value < 50 and a 'clock' in the column_name value"

Compute Value Syntax (-t, -r)

- -t sums the numerical columns for all selected rows (selected by -q)
- -r computes new columns
- formulas must be separated by semi-colon (;) and can contain math operators, functions defined in the math modules and in the random module. Each formula must have the form "new_col_name=value"
- EXAMPLE: `-r "revenue=price*quantity; discount=0.05*revenue"`
- Queries (-q) can be applied to computed fields (-r)