A Database Join on Text Files from the Unix Command Line

Let's perform a simple database join from the UNIX command line.

We start with two files, `household-ppp.tsv`, a tab-separated file of median household incomes by country, and `college-degrees-perc.tsv`, the percent of people with Bachelor or higher degrees in the age 25-64.

To look at the contents of a file, we use the `cat` command.

(The `!` in this worksheet means that the rest of the line is a UNIX command.)

In [76]:
```
!cat household-ppp.tsv
```

```
1       Luxembourg      34407
2       United States   31111
3       Norway  31011
4       Iceland 28166
5       Australia       26915
6       Switzerland     26844
7       Canada  25363
8       United Kingdom  25168
9       Ireland 24677
10      Austria 24114
11      Netherlands     24024
12      Sweden  22889
13      Denmark 22461
14      Belgium 21532
15      Germany 21241
16      Finland 20875
17      New Zealand     20679
18      France  19615
19      Japan   19432
20      South Korea     19179
```

These are spaced out in an odd way; the reason is that there isn't space characters but tab characters. We can make these visible with the `-vT` option to `cat`.

In [77]:
```
!cat -vT household-ppp.tsv
```

```
1^ILuxembourg^I34407
2^IUnited States^I31111
3^INorway^I31011
4^IIceland^I28166
5^IAustralia^I26915
6^ISwitzerland^I26844
7^ICanada^I25363
8^IUnited Kingdom^I25168
9^IIreland^I24677
10^IAustria^I24114
11^INetherlands^I24024
12^ISweden^I22889
13^IDenmark^I22461
14^IBelgium^I21532
15^IGermany^I21241
16^IFinland^I20875
17^INew Zealand^I20679
18^IFrance^I19615
19^IJapan^I19432
20^ISouth Korea^I19179
```

We can also format this more nicely by putting it throught the `column` command. There are many such commands available at the command line. You won't remember them all, but if you need something, instead of writing something yourself, take a minute or two looking in the man pages or on Google.

The scary looking expression with the printf is just saying "put a TAB" character here. We only need it here in the notebook because there's no easy way of typing a literal tab and because the notebook uses /bin/sh. On the command line, you just type a literal TAB with Control-V TAB, or in Bash you can write `$'\t'`.

In [21]:
```
!cat household-ppp.tsv | column -t -s "`printf '\t'`"
```

```
1   Luxembourg      34407
2   United States   31111
3   Norway          31011
4   Iceland         28166
5   Australia       26915
6   Switzerland     26844
7   Canada          25363
8   United Kingdom  25168
9   Ireland         24677
10  Austria         24114
11  Netherlands     24024
12  Sweden          22889
13  Denmark         22461
14  Belgium         21532
15  Germany         21241
16  Finland         20875
17  New Zealand     20679
18  France          19615
19  Japan           19432
20  South Korea     19179
```

Let's look at the second table. We use `head` instead of `cat`; `head` shows us just the initial contents of a file. Another useful command might be `more` or `less`.

In [78]:
```
!head college-degrees-perc.tsv
```

```
1       Norway  32
2       United States   31
3       Netherlands     29
4       Iceland 26
5       Denmark 25
6       New Zealand     25
7       Canada  25
8       South Korea     24
9       Australia       24
10      Sweden  23
```

Some people prefer using a more general tool called `sed` (stream editor). The following command says copy and edit files (with no editing to be done), and whe you hit line 7, then quit.

In [79]:
```
!sed 7q college-degrees-perc.tsv
```

```
1       Norway  32
2       United States   31
3       Netherlands     29
4       Iceland 26
5       Denmark 25
6       New Zealand     25
7       Canada  25
```

There are other kinds of edits we can make with `sed`, for example.

This command say:

- substitute `United States` with USA (the strings can be regular expressions)
- substitute `United Kingdom` with UK
- translate all spaces to underscores
- delete any line not containing either US or UK

In fact, `sed` really is a little programming language with one letter commands. The commands themselves are used in many apps in similar forms (e.g., `ed`, `vi`, etc.)

- s/old/new/g - text substitution with regular expressions
- /target/ - string search
- a - add line
- d - delete line
- y/old/new/ - replace characters
- ! - negation
- ; - statement separator

You don't have to remember all these commands, but you should know that they are there. And a few standard `sed` commands are useful to know.

In [80]:
```
!sed 's/United States/USA/;s/United Kingdom/UK/;y/ /_/;/US\|UK/!d' college-degrees-perc.tsv
```

```
2       USA     31
11      UK      23
```

In order to perform a relational join, we need to sort on the fields we are joining on. The sort command has many options. You can find out about them from the manual page.

In [81]:
```
!man sort | sed 30q
```

```
SORT(1)                          User Commands                          SORT(1)



NAME
       sort - sort lines of text files

SYNOPSIS
       sort [OPTION]... [FILE]...
       sort [OPTION]... --files0-from=F

DESCRIPTION
       Write sorted concatenation of all FILE(s) to standard output.

       Mandatory  arguments  to  long  options  are  mandatory for short options too.
       Ordering options:

       -b, --ignore-leading-blanks
              ignore leading blanks

       -d, --dictionary-order
              consider only blanks and alphanumeric characters

       -f, --ignore-case
              fold lower case to upper case characters

       -g, --general-numeric-sort
              compare according to general numerical value

       -i, --ignore-nonprinting
```

You can also find out about them from the `--help` argument.

In [82]:
```
!sort --help
```

```
Usage: sort [OPTION]... [FILE]...
  or:  sort [OPTION]... --files0-from=F
Write sorted concatenation of all FILE(s) to standard output.

Mandatory arguments to long options are mandatory for short options too.
Ordering options:

  -b, --ignore-leading-blanks  ignore leading blanks
  -d, --dictionary-order      consider only blanks and alphanumeric characters
  -f, --ignore-case           fold lower case to upper case characters
  -g, --general-numeric-sort  compare according to general numerical value
  -i, --ignore-nonprinting    consider only printable characters
  -M, --month-sort            compare (unknown) < `JAN' < ... < `DEC'
  -h, --human-numeric-sort    compare human readable numbers (e.g., 2K 1G)
  -n, --numeric-sort          compare according to string numerical value
  -R, --random-sort           sort by random hash of keys
      --random-source=FILE    get random bytes from FILE
  -r, --reverse               reverse the result of comparisons
      --sort=WORD             sort according to WORD:
                                general-numeric -g, human-numeric -h, month -M,
                                numeric -n, random -R, version -V
  -V, --version-sort          natural sort of (version) numbers within text

Other options:

      --batch-size=NMERGE   merge at most NMERGE inputs at once;
                            for more use temp files
  -c, --check, --check=diagnose-first  check for sorted input; do not sort
  -C, --check=quiet, --check=silent  like -c, but do not report first bad line
      --compress-program=PROG  compress temporaries with PROG;
                              decompress them with PROG -d
      --debug               annotate the part of the line used to sort,
                              and warn about questionable usage to stderr
      --files0-from=F       read input from the files specified by
                            NUL-terminated names in file F;
                            If F is - then read names from standard input
  -k, --key=POS1[,POS2]     start a key at POS1 (origin 1), end it at POS2
                            (default end of line).  See POS syntax below
  -m, --merge               merge already sorted files; do not sort
  -o, --output=FILE         write result to FILE instead of standard output
  -s, --stable              stabilize sort by disabling last-resort comparison
  -S, --buffer-size=SIZE    use SIZE for main memory buffer
  -t, --field-separator=SEP  use SEP instead of non-blank to blank transition
  -T, --temporary-directory=DIR  use DIR for temporaries, not $TMPDIR or /tmp;
                              multiple options specify multiple directories
      --parallel=N          change the number of sorts run concurrently to N
  -u, --unique              with -c, check for strict ordering;
                              without -c, output only the first of an equal run
  -z, --zero-terminated     end lines with 0 byte, not newline
      --help     display this help and exit
      --version  output version information and exit

POS is F[.C][OPTS], where F is the field number and C the character position
in the field; both are origin 1.  If neither -t nor -b is in effect, characters
in a field are counted from the beginning of the preceding whitespace.  OPTS is
one or more single-letter ordering options, which override global ordering
options for that key.  If no key is given, use the entire line as the key.

SIZE may be followed by the following multiplicative suffixes:
% 1% of memory, b 1, K 1024 (default), and so on for M, G, T, P, E, Z, Y.

With no FILE, or when FILE is -, read standard input.

*** WARNING ***
The locale specified by the environment affects sort order.
Set LC_ALL=C to get the traditional sort order that uses
native byte values.

Report sort bugs to bug-coreutils@gnu.org
GNU coreutils home page: <http://www.gnu.org/software/coreutils/>
General help using GNU software: <http://www.gnu.org/gethelp/>
Report sort translation bugs to <http://translationproject.org/team/>
```

We want to join on the country name. So let's sort on that field.

In [83]:
```
!sort -k 2 household-ppp.tsv
```

```
5       Australia       26915
10      Austria 24114
14      Belgium 21532
7       Canada  25363
13      Denmark 22461
16      Finland 20875
18      France  19615
15      Germany 21241
4       Iceland 28166
9       Ireland 24677
19      Japan   19432
1       Luxembourg      34407
11      Netherlands     24024
17      New Zealand     20679
3       Norway  31011
20      South Korea     19179
12      Sweden  22889
6       Switzerland     26844
8       United Kingdom  25168
2       United States   31111
```

That seems to have worked. So let's do that for both files now. So, let's sort both files and then join them.

In [4]:
```
!sort -k 2 household-ppp.tsv > A
!sort -k 2 college-degrees-perc.tsv > B
!join -t "`printf '\t'`" -j 2 A B
```

```
Australia       5       26915   9       24
Belgium 14      21532   24      14
Canada  7       25363   7       25
Denmark 13      22461   5       25
Finland 16      20875   13      21
France  18      19615   21      16
Germany 15      21241   20      16
Iceland 4       28166   4       26
Ireland 9       24677   14      21
Japan   19      19432   12      23
Luxembourg      1       34407   19      18
Netherlands     11      24024   3       29
New Zealand     17      20679   6       25
Norway  3       31011   1       32
South Korea     20      19179   8       24
Sweden  12      22889   10      23
Switzerland     6       26844   15      21
United Kingdom  8       25168   11      23
United States   2       31111   2       31
```

Each table contained a rank that we aren't interested in. So we need to cut out fields 1, 3, and 5. But the output of join is space separated and cut wants a tab separated input, so we use the `tr` command to translate spaced to tabs. We could also have used `sed "y/ /\t/"`.

In [5]:
```
!join -t "`printf '\t'`" -j 2 A B | cut -f 1,3,5 | sort -t "`printf '\t'`" -r -k 2 | tee table.tsv
```

```
Luxembourg      34407   18
United States   31111   31
Norway  31011   32
Iceland 28166   26
Australia       26915   24
Switzerland     26844   21
Canada  25363   25
United Kingdom  25168   23
Ireland 24677   21
Netherlands     24024   29
Sweden  22889   23
Denmark 22461   25
Belgium 21532   14
Germany 21241   16
Finland 20875   21
New Zealand     20679   25
France  19615   16
Japan   19432   23
South Korea     19179   24
```

Note that with the `tee` command, we simultaneously saved the output in a file and displayed it on standard output.

Let's now reformat this output a little more nicely, using the `column` command.

Note the use of "(...)" at the shell to combine the output from three commands.

In [18]:
```
!(echo "COUNTRY\tFAMILY-INCOME-PPP\tPERC-UNIVERSITY"; cat table.tsv) | column -t -s "`printf '\t'`"
```

```
COUNTRY         FAMILY-INCOME-PPP  PERC-UNIVERSITY
Luxembourg      34407              18
United States   31111              31
Norway          31011              32
Iceland         28166              26
Australia       26915              24
Switzerland     26844              21
Canada          25363              25
United Kingdom  25168              23
Ireland         24677              21
Netherlands     24024              29
Sweden          22889              23
Denmark         22461              25
Belgium         21532              14
Germany         21241              16
Finland         20875              21
New Zealand     20679              25
France          19615              16
Japan           19432              23
South Korea     19179              24
```

This may seem like an awful lot of work to join two small tables. Why not just write a Python script or load the data into a spreadsheet?

- The commands on arbitrarily large files, even those that don't fit in memory.
- These utilities handle Unicode, locales, and search order correctly.
- These utilities know how to handle disk caching etc. correctly, and they can use multiple cores.
- If you're skilled in both Python and UNIX, this will still be a lot faster to write on the command line.

In short, you'd have a hard time beating them for large problems with any code you are likely to be able to write.

NB: Your *locale* is defined by your environment variable LC_ALL and affects things like sort order.

In [85]: