**TOOLS & PROCEDURES**

1. **Load**

Given an input file, VendorMaster.txt for example. The following command shall create a data table named **vendors** and fill it with raw data from the input file.

ruby load.rb --host=localhost --username=postgres --password=postgres

--database=mydb

--input=/home/admin/VendorMaster.txt

--format=csv

--table=vendors

1. **Sanitize**

Certain input files contain invalid UTF-8 bytecodes, use the sanitize.rb script to remove all these characters from an input file before loading. Note that removal is done in-place, be sure to back up your original file if needed. Example:

ruby sanitize.rb /tmp/vendors.txt

1. **Verify**

Verification will then be made on vendors table. Normally, the followings are checked for certain fields

* Uniqueness
* Integrity
* Data type: numeric, text, enum, etc.
* Data format: text size, text values, enum, inclusion, exclusion, etc.

Look at an example:

ruby validate.rb --host=localhost --username=postgres --password=postgres

--database=mydb

--table=vendors

--log-to=errors

--unique=vendor\_code

--unique=vendor\_name

--not-null=vendor\_code

--not-null=vendor\_name

--match="vendor\_code/^V.\*/"

--match="vendor\_name/[a-zA-Z]+/"

--not-match="status/(failed|inactive)/"

The above script will check every record of **vendors** table:

* Uniqueness of **vendor\_code** and **vendor\_name**
* Value for **vendor\_code** and **vendor\_name** must be present (not null)
* Value of **vendor\_code** must start with **V** prefix (V001, V020… etc.)
* Value of **vendor\_name** must contain alphabet characters only
* Value of **status** must not be **failed** nor **inactive**

Records that do not pass the verification above will be marked accordingly (to the **errors** field as specified by the **--log-to=errors** switch)

1. **Other Processing**

**For simple processing with SQL query.** Some further processing may be required for certain type of input. If this is the case, pass your custom SQL queries to **patch.rb** script

ruby patch.rb --host=localhost --username=postgres --database=db

--query="ALTER TABLE vendors ADD COLUMN tmp text; UPDATE vendors

SET tmp = vendor\_number"

**For complex data manipulation.** Data manipulating using SQL query itself is sometimes a bit overhead. Then, we can take advantage of Ruby which is more robust for text processing. Use the eval.rb script to execute process every single record of the data table using Ruby syntax

ruby eval.rb --host=localhost --username=postgres --database=db --table=vendors

--eval="item.uom = item.uom.split(‘;’).first"

1. **Export**

After all records of **vendors** table have been checked, we dump it to a text file which is in turn used as input for the other application

ruby export.rb --host=localhost --username=postgres --database=db

--table=items

--output=/tmp/data.csv --format=csv --delim="\t"

--select="id, name AS vendor\_name, age" --where="active = 1"

**NOTES**

1. **Script organization**

Mỗi organization sẽ có một **PROFILE** (thực ra là 1 bash file, **cambell.sh** for example) chứa toàn bộ script cần chạy cho organization đó (cũng như ghi chú của những phần xử lý manual)

Advantages

* Có một nơi chứa **all-in-one** processing script, khi cần generate data cho một organization nào đó thì **chỉ cần run script** này, không cần phải nhớ những thao tác thủ công (bỏ input vào folder blabla, copy/paste nội dung file, mở PostgreSQL, copy/paste những đoạn script sau….)
* Dùng PROFILE để có được overview (đã làm gì trên organization này)
* Dùng PROFILE để report cho BA hay khách hàng
* Dùng chung script để chỉ phải sửa một lần (once and forever) cho một lỗi
* Tránh mistake do copy/paste/replace

1. **Some notes regarding how we move forward**

Please make sure:

* Contribute vào script chung để mọi người cùng dùng (và test giùm)
* Khi viết script mới hoặc feature mới cho script cần cân nhắc input/output để reusable. Mà thường đã viết vào script chung thì mình cũng đã bị forced viết theo kiểu reusable của wrapper, không muốn cũng không được ^.^

1. **Sample PROFILE**

# load data

ruby load.rb ...

# validate

ruby validate.rb ...

# other processing

ruby patch.rb ...

ruby patch.rb ...

ruby patch.rb ...

ruby eval.rb ...

ruby eval.rb ...

# export

ruby export.rb ...

ruby export.rb ...

**DOCUMENTATION**

 