# Usage Translator Coding Exercise

## Objective

As a software developer candidate, you are required to complete a coding exercise, and will be evaluated on the following criteria:

Functionality**:** Does it satisfy the functional requirements as stated without any defects?

Maintainability: How easily is it for fellow developers to understand and modify the code?

Performance: What are the time and space complexity?

Scalability: Is the solution easily extensible?

Testability: How easy will it be to perform unit and functional testing on the code.

## User Story

As provider of email and web hosting services, amongst others, we collect account usage data.

We need to enter usage data into our databases for billing processing and others.

### Inputs – the process will be provided with two input files.

Sample\_report.csv – a CSV file containing usage info to be entered into the database according to the acceptance criteria below.

typemap.json – A JSON file used to map PartNumber value from the CSV file to a ‘product’ value to be inserted in the database.

## Acceptance Criteria

Read the attached Sample\_Report.csv file and output two SQL INSERT statements for the following normalized tables. You can assume storage size is sufficient for the purpose. You can also assume the tables are empty.

* 1. **The ‘chargeable’ table will have columns**
     1. id: int auto-increment
     2. partnerID: int
     3. product: varchar
     4. partnerPurchasedPlanID: varchar
     5. plan: varchar
     6. usage: int

*The implementation of the code for the* ***‘chargeable’*** *table should:*

* 1. Log an error and skip entries:
     + - * without ‘PartNumber’
         * with non-positive ‘itemCount’
  2. Skip any entries where the value of PartnerID matches a configurable list of ‘PartnerID’ [Note: for the purpose of this exercise the list of PartnerIDs to skip contains just 26392]
  3. Map ‘PartNumber’ in the csv to the ‘product’ column in the ‘chargeable’ table based on the map in the attached typemap.json file. For example the PartNumber ADS000010U0R will be mapped to product value ‘core.chargeable.adsync’ for the insert.
  4. Map ‘accountGuid’ to ‘partnerPurchasedPlanID’ as alphanumeric string of length 32 and should strip any non-alphanumeric characters before insert.
  5. Map ‘itemCount’ in csv as ‘usage’ in the table subject to a unit reduction rule which for the purpose of this exercise is as follows,
     + - * EA000001GB0O: 1000
         * PMQ00005GB0R: 5000
         * SSX006NR: 1000
         * SPQ00001MB0R: 2000

*For example, an itemCount of 5000 for the PartNumber ‘PMQ00005GB0R’ would be inserted as usage = 5000/5000 = 1*

*Insert the quantity as is if there is no unit reduction rule.*

* 1. Output stats of running totals over ‘itemCount’ for each of the products in a success log.

1. **The ‘domains’ table will have columns,**
2. id: int auto-increment
3. partnerPurchasedPlanID: varchar
4. domain: varchar

*The implementation of the code for the* ***‘domains’*** *table should:*

1. Record the Domain associated with the partnerPurchasedPlanID in the table
2. Ensure only distinct domain names are recorded in the ‘domains’ table

## Deliverables

* **An archive of the code you have developed to solve this problem**

*(using language of your choice \*but Java preferred\*)*

* **Instructions and/or documentation for how to execute the code**

*(i.e. environment, third-party libraries, etc). You may use any non-GPL open source library that is readily available on the internet.*

* **A printed output of the code’s response. This should really just look like two INSERT statements.**

*(We will use this if we cannot get the program to run in our environment)*

## **Additional Criteria for Evaluation** *(Not necessary, but GREAT if you can)*

As a bonus feature code will also be evaluated to consider how the resulting input statements are secured from exploitation. (*One example to consider is SQL injection.)*