Data acquisition from Kaggle

Important Note: You must sign up for the competition here and download your kaggle.json from your Kaggle account page. See Steps 1-2 here for more information.

In [ ]:
```
from google.colab import files

# UPLOAD YOUR KAGGLE.JSON
# Only run this cell if you need to upload kaggle.json
files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kaggle.json to kaggle.json
```

Out[ ]:
```
{'kaggle.json': b'{"username":"catapultic","key":"bc709cc2cfed23022adc91952ba357c7"}'}
```

In [ ]:
```
# Kaggle credentials setup
!pip install kaggle
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-packages
(1.5.12)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-packag
es (from kaggle) (1.15.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages
(from kaggle) (2021.5.30)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (f
rom kaggle) (4.62.3)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages
(from kaggle) (1.24.3)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-
packages (from kaggle) (2.8.2)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-p
ackages (from kaggle) (5.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-package
s (from kaggle) (2.23.0)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.7/d
ist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-pac
kages (from requests->kaggle) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dis
t-packages (from requests->kaggle) (3.0.4)
```

In [ ]:
```
# Download Coupon Purchase Prediction data set
!kaggle competitions download -c coupon-purchase-prediction -p data
```

```
Warning: Looks like you're using an outdated API Version, please consider updati
ng (server 1.5.12 / client 1.5.4)
Downloading coupon_list_test.csv.zip to data
  0% 0.00/11.6k [00:00<?, ?B/s]
100% 11.6k/11.6k [00:00<00:00, 7.42MB/s]
Downloading coupon_area_train.csv.zip to data
  0% 0.00/832k [00:00<?, ?B/s]
```

```
100% 832k/832k [00:00<00:00, 53.7MB/s]
Downloading sample_submission.csv.zip to data
  0% 0.00/400k [00:00<?, ?B/s]
100% 400k/400k [00:00<00:00, 57.4MB/s]
Downloading coupon_area_test.csv.zip to data
  0% 0.00/14.0k [00:00<?, ?B/s]
100% 14.0k/14.0k [00:00<00:00, 28.7MB/s]
Downloading documentation.zip to data
  0% 0.00/21.6k [00:00<?, ?B/s]
100% 21.6k/21.6k [00:00<00:00, 19.6MB/s]
Downloading coupon_visit_train.csv.zip to data
 77% 65.0M/84.5M [00:03<00:02, 9.31MB/s]
100% 84.5M/84.5M [00:03<00:00, 23.2MB/s]
Downloading coupon_list_train.csv.zip to data
  0% 0.00/656k [00:00<?, ?B/s]
100% 656k/656k [00:00<00:00, 42.4MB/s]
Downloading user_list.csv.zip to data
  0% 0.00/627k [00:00<?, ?B/s]
100% 627k/627k [00:00<00:00, 88.5MB/s]
Downloading prefecture_locations.csv to data
  0% 0.00/2.00k [00:00<?, ?B/s]
100% 2.00k/2.00k [00:00<00:00, 1.73MB/s]
Downloading coupon_detail_train.csv.zip to data
 68% 5.00M/7.32M [00:01<00:00, 4.20MB/s]
100% 7.32M/7.32M [00:01<00:00, 6.06MB/s]
```

In [ ]:

```
# unzip and reorganize the zipped tables
# Master list of users
!unzip data/user_list.csv.zip -d data/

# Master list of coupons (train & test)
!unzip data/coupon_list_train.csv.zip -d data/
!unzip data/coupon_list_test.csv.zip -d data/

# Table containing physical areas where coupons are available (train & test)
!unzip data/coupon_area_train.csv.zip -d data/
!unzip data/coupon_area_test.csv.zip -d data/

# Purchase log of users buying coupons during the training period (train only)
!unzip data/coupon_detail_train.csv.zip -d data/

# Browsing log of users visiting coupons during the training period (train only)
!unzip data/coupon_visit_train.csv.zip -d data/
```

```
Archive:  data/user_list.csv.zip
  inflating: data/user_list.csv
Archive:  data/coupon_list_train.csv.zip
  inflating: data/coupon_list_train.csv
Archive:  data/coupon_list_test.csv.zip
  inflating: data/coupon_list_test.csv
Archive:  data/coupon_area_train.csv.zip
  inflating: data/coupon_area_train.csv
Archive:  data/coupon_area_test.csv.zip
  inflating: data/coupon_area_test.csv
Archive:  data/coupon_detail_train.csv.zip
  inflating: data/coupon_detail_train.csv
Archive:  data/coupon_visit_train.csv.zip
  inflating: data/coupon_visit_train.csv
```

In [ ]:

```
# Delete unused zip files
!rm -f data/*.zip
```

Translation of Japanese columns to English

Note: This does a full translation of the Japanese characters to English. It does not transliterate the Japanese place names to their English counterparts. We end up with the actual meaning of the Japanese names sometimes, like "Place which is by the water." That is okay - it is not important for training, they just help us explore the data and understand what we are looking at.

In [ ]:
```
# dependencies
%%capture
!pip install git+https://github.com/neuml/txtai#egg=txtai[pipeline]
!pip install pykakasi

# imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm
from txtai.pipeline import Translation
import pykakasi

translate = Translation()
kks = pykakasi.kakasi()
```

Translation helper functions

In [ ]:
```
# Lookup table of translations to save time
translations = {}

# Translates jp->en using txtai package (unless NaN)
def safe_translate(jp, transliterate=False):
  if pd.isna(jp) == False:
    if transliterate == True: # use pykakasi
      return ''.join([item['hepburn'].capitalize() for item in kks.convert(jp)])
    else:
      return translate(jp, 'en') # using txtai
  else:
    return jp

# Checks the translation dict first before translating
def lookup_or_translate(jp):
  if (jp not in translations):
    translations[jp] = safe_translate(jp) # pass transliterate=True to use kakas
  return translations[jp]

# Translates an entire column/list of data
def translate_list(data):
  translated = []
  for word in tqdm(data):
    t = lookup_or_translate(word)
    translated.append(t)
  return translated
```

In [ ]:
```
# Main loading function - takes a csv path, columns to translate,
# and returns a Pandas dataframe. Translates columns in-place.
def load_translate(csv_path, translate_columns=[]):
```

```
  df = pd.read_csv(csv_path)
  for c in translate_columns:
    df[f'{c}_en_t'] = translate_list(df[c])
  return df
```

In [ ]:

```
# Create lists of columns that need to be translated for each table
# Coupon Visit Training set does not require any translation

# User list table
user_cols = ['PREF_NAME']

# Coupon list train and test
c_list_cols = ['CAPSULE_TEXT', 'GENRE_NAME', 'large_area_name',
               'ken_name', 'small_area_name']

# Coupon detail
c_detail_cols = ['SMALL_AREA_NAME']

# Coupon area train and test
c_area_cols = ['SMALL_AREA_NAME', 'PREF_NAME']

# Prefecture locations
c_pref_cols = ['PREF_NAME', 'PREFECTUAL_OFFICE']
```

In [ ]:

```
# Perform the translations and load the data into DataFrames
df_users = load_translate('data/user_list.csv', user_cols)
df_area_train = load_translate('data/coupon_area_train.csv', c_area_cols)
df_area_test = load_translate('data/coupon_area_test.csv', c_area_cols)
df_c_list_train = load_translate('data/coupon_list_train.csv', c_list_cols)
df_c_list_test  = load_translate('data/coupon_list_test.csv', c_list_cols)
df_c_detail_train = load_translate('data/coupon_detail_train.csv', c_detail_cols
df_visit_train = load_translate('data/coupon_visit_train.csv')
df_locations = load_translate('data/prefecture_locations.csv', c_pref_cols)
```

In [ ]:

```
# Map JP-EN for prefecture names.
pref_names_jp = df_users.PREF_NAME.unique()
pref_names_en = ['N/A', 'Tokyo', 'Aichi Prefecture', 'Kanagawa Prefecture',
                 'Hiroshima Prefecture', 'Saitama Prefecture', 'Nara Prefecture',
                 'Ishikawa Prefecture', 'Osaka prefecture',
                 'Kumamoto Prefecture', 'Fukuoka Prefecture', 'Hokkaido', 'Kyoto'
                 'Akita', 'Chiba Prefecture', 'Nagasaki Prefecture',
                 'Hyogo Prefecture', 'Okinawa', 'Mie', 'Ibaraki Prefecture',
                 'Kagoshima Prefecture', 'Miyagi Prefecture', 'Shizuoka Prefectur
                 'Wakayama Prefecture', 'Nagano Prefecture', 'Okayama Prefecture'
                 'Tochigi Prefecture','Shiga Prefecture', 'Toyama Prefecture',
                 'Saga Prefecture', 'Miyazaki Prefecture', 'Iwate Prefecture',
                 'Niigata Prefecture', 'Oita Prefecture', 'Yamaguchi Prefecture',
                 'Gifu Prefecture','Gunma Prefecture', 'Fukushima Prefecture',
                 'Ehime Prefecture', 'Kagawa Prefecture', 'Yamanashi Prefecture',
                 'Kochi Prefecture', 'Shimane Prefecture', 'Tokushima Prefecture'
                 'Fukui Prefecture', 'Aomori Prefecture', 'Yamagata Prefecture',
                 'Tottori Prefecture']

print(f'Dictionary length - jp: {len(pref_names_jp)}, en: {len(pref_names_en)}')
pref_name_dict = {k:v for k, v in zip(pref_names_jp, pref_names_en)}

df_users['PREF_NAME_EN'] = df_users['PREF_NAME'].map(pref_name_dict)
```

```
df_users = df_users.drop(columns=['PREF_NAME', 'PREF_NAME_en_t'])
df_users
```

```
jp: 48, en: 48
```

Out[ ]:

	REG_DATE	SEX_ID	AGE	WITHDRAW_DATE	USER_ID_hash	PREF_N
0	2012-03-28 14:14:18	f	25	NaN	d9dca3cb44bab12ba313eaa681f663eb	
1	2011-05-18 00:41:48	f	34	NaN	560574a339f1b25e57b0221e486907ed	
2	2011-06-13 16:36:58	m	41	NaN	e66ae91b978b3229f8fd858c80615b73	Aichi P
3	2012-02-08 12:56:15	m	25	NaN	43fc18f32eafb05713ec02935e2c2825	
4	2011-05-22 23:43:56	m	62	NaN	dc6df8aa860f8db0d710ce9d4839840f	I P
...	...	...	...	...	...	
22868	2011-12-12 15:42:56	f	24	NaN	2f0a2f36a9f63b6ba2fa3a7e53bef906	
22869	2011-08-10 00:49:55	m	41	NaN	6ae7811a9c7c58546d6a1567ab098c21	
22870	2012-04-05 12:24:23	f	35	NaN	a417308c6a79ae0d86976401ec2e3b04	
22871	2011-02-20 10:34:22	f	59	NaN	4937ec1c86e71d901c4ccc0357cff0b1	
22872	2011-02-24 15:43:18	f	38	NaN	280f0cedda5c4b171ee6245889659571	P

22873 rows × 6 columns

In [ ]:

```
# Translate coupon capsule text
capsule_text_jp = df_c_list_train['CAPSULE_TEXT'].unique()
capsule_text_en = ['Restaurant', 'Hair salon', 'Spa', 'Relaxation', 'Beauty',
                   'Nail and eye salon', 'Delivery service', 'Lesson',
                   'Gift card', 'Other coupon', 'Leisure',
                   'Hotel', 'Japanese inn', 'Vacation rental', 'Lodge',
                   'Resort inn', 'Guest house', 'Japanese guest house',
                   'Public inn', 'Beauty', 'Event', 'Web service',
                   'Health / medical', 'Class', 'Correspondence course']



print(f'dictionary length - jp: {len(capsule_text_jp)}, en: {len(capsule_text_en
```

```
capsule_text_dict = {k:v for k, v in zip(capsule_text_jp, capsule_text_en)}
df_c_list_train['CAPSULE_TEXT_EN'] = df_c_list_train['CAPSULE_TEXT'].map(capsule
df_c_list_test['CAPSULE_TEXT_EN'] = df_c_list_test['CAPSULE_TEXT'].map(capsule_t
```

```
dictionary length - jp: 25, en: 25
```

Out[ ]:

	CAPSULE_TEXT	GENRE_NAME	PRICE_RATE	CATALOG_PRICE	DISCOUNT_PRICE	DISPFROM
0	グルメ	グルメ	52	5659	2690	2012-06-26 12:00:00
1	グルメ	グルメ	52	18000	8500	2012-06-27 12:00:00
2	グルメ	グルメ	51	7200	3480	2012-06-28 12:00:00
3	グルメ	グルメ	50	3300	1650	2012-06-24 12:00:00
4	グルメ	グルメ	56	3650	1600	2012-06-26 12:00:00

In [ ]:

```
# Genre name translation
genre_name_jp = df_c_list_train['GENRE_NAME'].unique()
genre_name_en = ['Gourmet dining', 'Hair salon', 'Spa', 'Relaxation', 'Beauty',
                 'Nail and eye salon', 'Delivery service', 'Class', 'Gift Card',
                 'Other coupons', 'Leisure', 'Hotels and inns', 'Health and medi

assert len(genre_name_jp) == len(genre_name_en)

genre_name_dict = {k:v for k, v in zip(genre_name_jp, genre_name_en)}
df_c_list_train['GENRE_NAME_EN'] = df_c_list_train['GENRE_NAME'].map(genre_name_
df_c_list_test['GENRE_NAME_EN'] = df_c_list_test['GENRE_NAME'].map(genre_name_di

df_c_list_train = df_c_list_train.drop(columns=['CAPSULE_TEXT', 'GENRE_NAME'])
df_c_list_test = df_c_list_test.drop(columns=['CAPSULE_TEXT', 'GENRE_NAME'])
```

In [ ]:

```
df_c_list = pd.concat([df_c_list_train, df_c_list_test])
large_area_name_jp = df_c_list['large_area_name'].unique()
large_area_name_en = ['Kanto', 'Kansai', 'Tokai', 'Hokkaido', 'Kyushu-Okinawa',
                      'Tohoku', 'Shikoku', 'China', "Hokushin'etsu"]

assert len(large_area_name_jp) == len(large_area_name_en)

large_area_dict = {k:v for k, v in zip(large_area_name_jp, large_area_name_en)}

df_c_list_train['LARGE_AREA_NAME_EN'] = df_c_list_train['large_area_name'].map(l
df_c_list_test['LARGE_AREA_NAME_EN'] = df_c_list_test['large_area_name'].map(lar
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-62-209c6e35250e> in <module>()
      4                       'Tohoku', 'Shikoku', 'China', "Hokushin'etsu"]
      5
```

```
----> 6 assert len(large_area_name_jp) == len(large_area_name_en)
      7
      8 large_area_dict = {k:v for k, v in zip(large_area_name_jp, large_area_na
me_en)}

AssertionError:
```

In [ ]:
```
df_c_list_test = df_c_list_test.drop(columns=['large_area_name', 'CAPSULE_TEXT_e
df_c_list_test.head()
```

Out[ ]:

	PRICE_RATE	CATALOG_PRICE	DISCOUNT_PRICE	DISPFROM	DISPEND	DISPPERIOD	VALIDFR
0	52	5659	2690	2012-06-26 12:00:00	2012-06-30 12:00:00	4	2012-07
1	52	18000	8500	2012-06-27 12:00:00	2012-07-04 12:00:00	7	2012-07
2	51	7200	3480	2012-06-28 12:00:00	2012-07-05 12:00:00	7	2012-07
3	50	3300	1650	2012-06-24 12:00:00	2012-06-29 12:00:00	5	2012-06
4	56	3650	1600	2012-06-26 12:00:00	2012-07-03 12:00:00	7	2012-07

In [ ]:
```
# Small area name translation
small_area_jp = df_c_list['small_area_name'].unique()
small_area_jp
small_area_en = ["Saitama", "Chiba", "Shinjuku, Takadanobaba Nakano - Kichijoji"
                 "Kyoto", "Ebisu / Meguro / Shinagawa",
                 "Ginza, Shinbashi, Tokyo, Ueno", "Aichi",
                 "Kawasaki, Shonan, Hakone, etc", 'Hokkaido', "Fukuoka", "Tochig
                 "Minami other", "Shibuya, Aoyama, Jiyugaoka",
                 "Ikebukuro Kagurazaka-Akabane", "Akasaka, Roppongi, Azabu",
                 "Yokohama", "Miyagi", "Fukushima", "Much", "Kochi",
                 "Tachikawa Machida, Hachioji other", "Hiroshima","Niigata",
                 "Okayama", "Ehime", "Kagawa", "Northern", "Tokushima", "Hyogo",
                 "Gifu", "Miyazaki", "Nagasaki", "Ishikawa", "Yamagata", "Shizuo
                 "Aomori", "Okinawa", "Akita", "Nagano", "Iwate", "Kumamoto",
                 "Yamaguchi", "Saga", "Nara", "Triple", "Gunma", "Wakayama",
                 "Yamanashi", "Tottori", "Kagoshima", "Fukui", "Shiga", "Toyama"
                 "Shimane", "Ibaraki"]

assert len(small_area_jp) == len(small_area_en)

small_area_dict = {k:v for k, v in zip(small_area_jp, small_area_en)}
df_c_list_train['SMALL_AREA_NAME_EN'] = df_c_list_train['small_area_name'].map(s
df_c_list_test['SMALL_AREA_NAME_EN'] = df_c_list_test['small_area_name'].map(sma

df_c_list_train.head()
```

Out[ ]:

	PRICE_RATE	CATALOG_PRICE	DISCOUNT_PRICE	DISPFROM	DISPEND	DISPPERIOD	VALIDFR

	PRICE_RATE	CATALOG_PRICE	DISCOUNT_PRICE	DISPFROM	DISPEND	DISPPERIOD	VALIDFR
0	50	3000	1500	2011-07-08 12:00:00	2011-07-09 12:00:00	1	2011-07
1	51	2080	1000	2011-07-01 12:00:00	2011-07-02 12:00:00	1	2011-07
2	50	7000	3500	2011-07-12 12:00:00	2011-07-15 12:00:00	3	2011-07
3	50	3000	1500	2011-07-09 12:00:00	2011-07-11 12:00:00	2	2011-07
4	50	2000	1000	2011-07-05 12:00:00	2011-07-06 12:00:00	1	2011-07

In [ ]:
```
df_c_list_train = df_c_list_train.drop(columns=['small_area_name', 'small_area_n
df_c_list_test = df_c_list_test.drop(columns=['small_area_name', 'small_area_nam
```

In [ ]:
```
df_c_list_test.head()
```

Out[ ]:

	PRICE_RATE	CATALOG_PRICE	DISCOUNT_PRICE	DISPFROM	DISPEND	DISPPERIOD	VALIDFR
0	52	5659	2690	2012-06-26 12:00:00	2012-06-30 12:00:00	4	2012-07
1	52	18000	8500	2012-06-27 12:00:00	2012-07-04 12:00:00	7	2012-07
2	51	7200	3480	2012-06-28 12:00:00	2012-07-05 12:00:00	7	2012-07
3	50	3300	1650	2012-06-24 12:00:00	2012-06-29 12:00:00	5	2012-06
4	56	3650	1600	2012-06-26 12:00:00	2012-07-03 12:00:00	7	2012-07

In [ ]:
```
# ken name
ken_jp = df_c_list['ken_name'].unique()
ken_en = []

for k in ken_jp:
  if k in pref_name_dict:
    ken_en.append(pref_name_dict[k])
  else:
    ken_en.append(k)

ken_en
```

```
assert len(ken_jp) == len(ken_en)

ken_dict = {k:v for k, v in zip(ken_jp, ken_en)}
df_c_list_train['KEN_NAME_EN'] = df_c_list_train['ken_name'].map(ken_dict)
df_c_list_test['KEN_NAME_EN'] = df_c_list_test['ken_name'].map(ken_dict)

df_c_list_train = df_c_list_train.drop(columns=['ken_name', 'ken_name_en_t'])
df_c_list_test = df_c_list_test.drop(columns=['ken_name', 'ken_name_en_t'])

df_c_list_train.head()
```

Out[ ]:

	PRICE_RATE	CATALOG_PRICE	DISCOUNT_PRICE	DISPFROM	DISPEND	DISPPERIOD	VALIDFR
0	50	3000	1500	2011-07-08 12:00:00	2011-07-09 12:00:00	1	2011-07
1	51	2080	1000	2011-07-01 12:00:00	2011-07-02 12:00:00	1	2011-07
2	50	7000	3500	2011-07-12 12:00:00	2011-07-15 12:00:00	3	2011-07
3	50	3000	1500	2011-07-09 12:00:00	2011-07-11 12:00:00	2	2011-07
4	50	2000	1000	2011-07-05 12:00:00	2011-07-06 12:00:00	1	2011-07

In [ ]:

```
df_c_detail_train['SMALL_AREA_NAME_EN'] = df_c_detail_train['SMALL_AREA_NAME'].m
print(df_c_detail_train['SMALL_AREA_NAME_EN'].unique())
```

```
['Hyogo' 'Ginza, Shinbashi, Tokyo, Ueno' 'Ebisu / Meguro / Shinagawa'
 'Shibuya, Aoyama, Jiyugaoka' 'Shinjuku, Takadanobaba Nakano - Kichijoji'
 'Gunma' 'Aichi' 'Yamagata' 'Akasaka, Roppongi, Azabu'
 'Kawasaki, Shonan, Hakone, etc' 'Saitama' 'Yokohama' 'Tochigi'
 'Hiroshima' 'Ikebukuro Kagurazaka-Akabane' 'Triple' 'Gifu' 'Shizuoka'
 'Northern' 'Minami other' 'Shiga' 'Kyoto' 'Hokkaido' 'Ishikawa' 'Nagano'
 'Chiba' 'Wakayama' 'Kagoshima' 'Saga' 'Nagasaki' 'Fukuoka' 'Much'
 'Miyazaki' 'Okinawa' 'Tachikawa Machida, Hachioji other' 'Iwate' 'Toyama'
 'Shimane' 'Yamaguchi' 'Nara' 'Fukushima' 'Aomori' 'Miyagi' 'Ibaraki'
 'Akita' 'Okayama' 'Ehime' 'Kumamoto' 'Kagawa' 'Tokushima' 'Kochi' 'Fukui'
 'Niigata' 'Tottori' 'Yamanashi']
```

In [ ]:

```
df_c_detail_train = df_c_detail_train.drop(columns=['SMALL_AREA_NAME', 'SMALL_AR
df_c_detail_train
```

Out[ ]:

	ITEM_COUNT	I_DATE	PURCHASEID_hash	USER
0	1	2012-03-28 15:06:06	c820a8882374a4e472f0984a8825893f	d9dca3cb44bab12ba313eaa6
1	1	2011-07-04 23:52:54	1b4eb2435421ede98c8931c42e8220ec	560574a339f1b25e57b0221e

	ITEM_COUNT	I_DATE	PURCHASEID_hash	USER
2	1	2011-07-16 00:52:49	36b5f9ba46c44b65587d0b16f2e4c77f	560574a339f1b25e57b0221e
3	1	2011-07-16 00:54:53	2f30f46937cc9004774e576914b2aa1a	560574a339f1b25e57b0221e
4	1	2011-07-16 00:55:52	4d000c64a55ac573d0ae1a8f03677f50	560574a339f1b25e57b0221e
...	...	...	...	
168991	1	2012-02-07 12:14:50	84b0c66349ae3c807f1d4601bfc0e8f6	280f0cedda5c4b171ee62458
168992	1	2012-02-28 15:43:21	f7b2b854457ae6ece44be04c32520064	280f0cedda5c4b171ee62458
168993	1	2012-03-19 12:11:16	e12f28eb208f5466dede7a7cb2fc566b	280f0cedda5c4b171ee62458
168994	2	2012-04-12 12:27:34	bcade77b186543a4820b3a6e3c06ad2f	280f0cedda5c4b171ee62458
168995	1	2012-05-09 12:12:26	fec51967a2f8135aa929cf2b5cc8722c	280f0cedda5c4b171ee62458

168996 rows × 6 columns

In [ ]:
```
df_area_test['SMALL_AREA_NAME_EN'] = df_area_test['SMALL_AREA_NAME'].map(small_a
df_area_test['PREF_NAME_EN'] = df_area_test['PREF_NAME'].map(pref_name_dict)
df_area_test
```

Out[ ]:

	SMALL_AREA_NAME	PREF_NAME	COUPON_ID_hash	SMALL_AREA_NAMI
0	京都	京都府	c76ea297ebd3a5a4d3bf9f75269f66fa	
1	ミナミ他	大阪府	c76ea297ebd3a5a4d3bf9f75269f66fa	Min
2	銀座・新橋・東京・上野	東京都	dd74dc95ca294afa02db40a543ae1763	Ginza, New Bridge,
3	川崎・湘南・箱根他	神奈川県	c65b550cbef918796ad53b1d5b7165c1	I'll be right back. I'll b back.
4	埼玉	埼玉県	c65b550cbef918796ad53b1d5b7165c1	
...	...	...	...	
2160	ミナミ他	大阪府	f9c657ce7ca80b3766ced3a9a3c709bb	Min
2161	福井	福井県	f9c657ce7ca80b3766ced3a9a3c709bb	
2162	鳥取	鳥取県	f9c657ce7ca80b3766ced3a9a3c709bb	Birdc

	SMALL_AREA_NAME	PREF_NAME	COUPON_ID_hash	SMALL_AREA_NAMI
2163	滋賀	滋賀県	f9c657ce7ca80b3766ced3a9a3c709bb	
2164	香川	香川県	f9c657ce7ca80b3766ced3a9a3c709bb	Kar

2165 rows × 7 columns

In [ ]:
```
df_area_test = df_area_test.drop(columns=['SMALL_AREA_NAME', 'SMALL_AREA_NAME_en
df_area_test.head()
```

Out[ ]:

	COUPON_ID_hash	SMALL_AREA_NAME_EN	PREF_NAME_EN
0	c76ea297ebd3a5a4d3bf9f75269f66fa	Kyoto	Kyoto
1	c76ea297ebd3a5a4d3bf9f75269f66fa	Minami other	Osaka prefecture
2	dd74dc95ca294afa02db40a543ae1763	Ginza, Shinbashi, Tokyo, Ueno	Tokyo
3	c65b550cbef918796ad53b1d5b7165c1	Kawasaki, Shonan, Hakone, etc	Kanagawa Prefecture
4	c65b550cbef918796ad53b1d5b7165c1	Saitama	Saitama Prefecture

In [ ]:
```
df_visit_train.head()
```

Out[ ]:

	PURCHASE_FLG	I_DATE	PAGE_SERIAL	REFERRER_hash	V
0	0	2012-03-28 14:15:00	7	7d3892e54acb559ae36c459978489330	34c48f84026e(
1	0	2012-03-28 14:17:28	9	7d3892e54acb559ae36c459978489330	34c48f84026e(
2	0	2012-03-28 14:20:05	16	7d3892e54acb559ae36c459978489330	17c450c3b470c
3	0	2012-03-28 14:23:16	18	7d3892e54acb559ae36c459978489330	91a15e6a95d0
4	0	2012-03-28 14:26:25	20	7d3892e54acb559ae36c459978489330	96fcbc8f6e45

In [ ]:
```
df_locations['PREF_NAME_EN'] = df_locations['PREF_NAME'].map(pref_name_dict)

po_jp = df_locations['PREFECTUAL_OFFICE'].unique()
```

In [ ]:
```
po_jp
```

Out[ ]:
```
array(['札幌市', '青森市', '盛岡市', '仙台市', '秋田市', '山形市', '福島市', '水戸市',
'宇都宮市',
```

```
       '前橋市', 'さいたま市', '千葉市', '新宿区', '横浜市', '新潟市', '富山市', '金沢
市', '福井市',
       '甲府市', '長野市', '岐阜市', '静岡市', '名古屋市', '津市', '大津市', '京都市',
'大阪市',
       '神戸市', '奈良市', '和歌山市', '鳥取市', '松江市', '岡山市', '広島市', '山口市',
'徳島市',
       '高松市', '松山市', '高知市', '福岡市', '佐賀市', '長崎市', '熊本市', '大分市',
'宮崎市',
       '鹿児島市', '那覇市'], dtype=object)
```

In [ ]:

```
po_en = ['Sapporo', 'Aomori City', 'Morioka City', 'Sendai City', 'Akita City',
         'Maebashi', 'Saitama City', 'Chiba', 'Shinjuku ward', 'Yokohama City',
         'Kofu City', 'Nagano City', 'Gifu City', 'Shizuoka City', 'Nagoya City'
         'Kobe City', 'Nara City', 'Wakayama City', 'Tottori City', 'Matsue', 'O
         'Takamatsu City', 'Matsuyama City', 'Kochi City', 'Fukuoka City', 'Saga
         'Kagoshima City', 'Naha City']

assert len(po_jp) == len(po_en)

pref_office_dict = {k:v for k, v in zip(po_jp, po_en)}
df_locations['PREFECTUAL_OFFICE_EN'] = df_locations['PREFECTUAL_OFFICE'].map(pre
df_locations
```

Out[ ]:

	PREF_NAME	PREFECTUAL_OFFICE	LATITUDE	LONGITUDE	PREF_NAME_en_t	PREFECTUAL_(
0	北海道	札幌市	43.063968	141.347899	Hokkaido	Y(
1	青森県	青森市	40.824623	140.740593	Aomori	
2	岩手県	盛岡市	39.703531	141.152667	Iwatea	And yet there
3	宮城県	仙台市	38.268839	140.872103	Miyagi	
4	秋田県	秋田市	39.718600	140.102334	Akita	
5	山形県	山形市	38.240437	140.363634	Hierarchy	N
6	福島県	福島市	37.750299	140.467521	Fukushima	Fu
7	茨城県	水戸市	36.341813	140.446793	Zhengji prefecture	(For fully forma
8	栃木県	宇都宮市	36.565725	139.883565	Tsai	
9	群馬県	前橋市	36.391208	139.060156	Cycling prefectures	Fror
10	埼玉県	さいたま市	35.857428	139.648933	Zheng-yang	I'm sorry, I'm so
11	千葉県	千葉市	35.605058	140.123308	Cypriot	
12	東京都	新宿区	35.689521	139.691704	Tokyo City	
13	神奈川県	横浜市	35.447753	139.642514	Kanagawa	Y(

	PREF_NAME	PREFECTUAL_OFFICE	LATITUDE	LONGITUDE	PREF_NAME_en_t	PREFECTUAL_(
14	新潟県	新潟市	37.902418	139.023221	Nungga prefecture	
15	富山県	富山市	36.695290	137.211338	Toshiyama	To
16	石川県	金沢市	36.594682	136.625573	Isegawa	
17	福井県	福井市	36.065219	136.221642	Fukui	
18	山梨県	甲府市	35.664158	138.568449	Yamanashi	The city o
19	長野県	長野市	36.651289	138.181224	Nagano	
20	岐阜県	岐阜市	35.391227	136.722291	Zheng Zhou	
21	静岡県	静岡市	34.976978	138.383054	Shizuoka	S
22	愛知県	名古屋市	35.180188	136.906565	PHILIPPIA	
23	三重県	津市	34.730283	136.508591	Tribunal	I'm sorry, I'm so
24	滋賀県	大津市	35.004531	135.868590	Kaji prefecture	
25	京都府	京都市	35.021004	135.755608	Kyoto.	
26	大阪府	大阪市	34.686316	135.519711	Osaka.	
27	兵庫県	神戸市	34.691279	135.183025	(For fully formatted text, see publication)	
28	奈良県	奈良市	34.685333	135.832744	Nara	
29	和歌山県	和歌山市	34.226034	135.167506	Kunihiyama	(For fully forma pul
30	鳥取県	鳥取市	35.503869	134.237672	Torigo	
31	島根県	松江市	35.472297	133.050499	island root	
32	岡山県	岡山市	34.661772	133.934675	Okayama	(
33	広島県	広島市	34.396560	132.459622	Hiroshima	H
34	山口県	山口市	34.186121	131.470500	Yamaguchi	Ya
35	徳島県	徳島市	34.065770	134.559303	Tokushima	To
36	香川県	高松市	34.340149	134.043444	Kagawa	
37	愛媛県	松山市	33.841660	132.765362	PHILIPPIA	Ma
38	高知県	高知市	33.559705	133.531080	HUDU	T

	PREF_NAME	PREFECTUAL_OFFICE	LATITUDE	LONGITUDE	PREF_NAME_en_t	PREFECTUAL_(
39	福岡県	福岡市	33.606785	130.418314	Zhao	
40	佐賀県	佐賀市	33.249367	130.298822	Saga prefecture	
41	長崎県	長崎市	32.744839	129.873756	Nagasaki	I
42	熊本県	熊本市	32.789828	130.741667	Kumamoto	B
43	大分県	大分市	33.238194	131.612591	Biggest prefectures	
44	宮崎県	宮崎市	31.911090	131.423855	Miyazaki	
45	鹿児島県	鹿児島市	31.560148	130.557981	Kagoshima	Ka
46	沖縄県	那覇市	26.212401	127.680932	Okinawa	What's the mat

In [ ]:
```
df_locations = df_locations.drop(columns=['PREF_NAME', 'PREFECTUAL_OFFICE', 'PRE
df_locations
```

Out[ ]:

	LATITUDE	LONGITUDE	PREF_NAME_EN	PREFECTUAL_OFFICE_EN
0	43.063968	141.347899	Hokkaido	Sapporo
1	40.824623	140.740593	Aomori Prefecture	Aomori City
2	39.703531	141.152667	Iwate Prefecture	Morioka City
3	38.268839	140.872103	Miyagi Prefecture	Sendai City
4	39.718600	140.102334	Akita	Akita City
5	38.240437	140.363634	Yamagata Prefecture	Yamagata City
6	37.750299	140.467521	Fukushima Prefecture	Fukushima City
7	36.341813	140.446793	Ibaraki Prefecture	Mito City
8	36.565725	139.883565	Tochigi Prefecture	Utsunomiya City
9	36.391208	139.060156	Gunma Prefecture	Maebashi
10	35.857428	139.648933	Saitama Prefecture	Saitama City
11	35.605058	140.123308	Chiba Prefecture	Chiba
12	35.689521	139.691704	Tokyo	Shinjuku ward
13	35.447753	139.642514	Kanagawa Prefecture	Yokohama City
14	37.902418	139.023221	Niigata Prefecture	Niigata City
15	36.695290	137.211338	Toyama Prefecture	Toyama City
16	36.594682	136.625573	Ishikawa Prefecture	Kanazawa
17	36.065219	136.221642	Fukui Prefecture	Fukui City
18	35.664158	138.568449	Yamanashi Prefecture	Kofu City

	LATITUDE	LONGITUDE	PREF_NAME_EN	PREFECTUAL_OFFICE_EN
19	36.651289	138.181224	Nagano Prefecture	Nagano City
20	35.391227	136.722291	Gifu Prefecture	Gifu City
21	34.976978	138.383054	Shizuoka Prefecture	Shizuoka City
22	35.180188	136.906565	Aichi Prefecture	Nagoya City
23	34.730283	136.508591	Mie	Tsu City
24	35.004531	135.868590	Shiga Prefecture	Otsu City
25	35.021004	135.755608	Kyoto	Kyoto City
26	34.686316	135.519711	Osaka prefecture	Osaka City
27	34.691279	135.183025	Hyogo Prefecture	Kobe City
28	34.685333	135.832744	Nara Prefecture	Nara City
29	34.226034	135.167506	Wakayama Prefecture	Wakayama City
30	35.503869	134.237672	Tottori Prefecture	Tottori City
31	35.472297	133.050499	Shimane Prefecture	Matsue
32	34.661772	133.934675	Okayama Prefecture	Okayama City
33	34.396560	132.459622	Hiroshima Prefecture	Hiroshima City
34	34.186121	131.470500	Yamaguchi Prefecture	Yamaguchi City
35	34.065770	134.559303	Tokushima Prefecture	Tokushima City
36	34.340149	134.043444	Kagawa Prefecture	Takamatsu City
37	33.841660	132.765362	Ehime Prefecture	Matsuyama City
38	33.559705	133.531080	Kochi Prefecture	Kochi City
39	33.606785	130.418314	Fukuoka Prefecture	Fukuoka City
40	33.249367	130.298822	Saga Prefecture	Saga City
41	32.744839	129.873756	Nagasaki Prefecture	Nagasaki City
42	32.789828	130.741667	Kumamoto Prefecture	Kumamoto City
43	33.238194	131.612591	Oita Prefecture	Oita City
44	31.911090	131.423855	Miyazaki Prefecture	Miyazaki City
45	31.560148	130.557981	Kagoshima Prefecture	Kagoshima City
46	26.212401	127.680932	Okinawa	Naha City

In [ ]:
```
# Save CSV files to translated output.
!mkdir data_translated
dir = 'data_translated'
```

In [ ]:
```
# df_users.to_csv(f'{dir}/user_list.csv')
# df_area_test.to_csv(f'{dir}/coupon_area_test.csv')
# df_area_train.to_csv(f'{dir}/coupon_area_train.csv')
# df_c_detail_train.to_csv(f'{dir}/coupon_detail_train.csv')
```

```
# df_c_list_test.to_csv(f'{dir}/coupon_list_test.csv')
# df_c_list_train.to_csv(f'{dir}/coupon_list_train.csv')
# df_visit_train.to_csv(f'{dir}/coupon_visit_train.csv')
# df_locations.to_csv(f'{dir}/prefecture_locations.csv')

!zip -r translated_data.zip data_translated/
```

```
adding: data_translated/ (stored 0%)
adding: data_translated/coupon_visit_train.csv (deflated 77%)
adding: data_translated/coupon_list_train.csv (deflated 79%)
adding: data_translated/prefecture_locations.csv (deflated 57%)
adding: data_translated/coupon_area_test.csv (deflated 86%)
adding: data_translated/coupon_detail_train.csv (deflated 64%)
adding: data_translated/coupon_area_train.csv (deflated 87%)
adding: data_translated/user_list.csv (deflated 57%)
adding: data_translated/coupon_list_test.csv (deflated 78%)
```

In [ ]: