Week 2 — Ingest and Explore the Dataset

Importing necessary libraries

In [1]:
```
# pip install dask[complete]

import pandas as pd
import numpy as np
import dask.dataframe as dd
import seaborn as sns
import matplotlib.pyplot as plt
```

Since the dataset is too large to be loaded at once, we will be use the chunksize function in pandas to initially explore the dataset

In [2]:
```
chunk_size = 100000
chunks = pd.read_csv('train_ver2.csv', chunksize = chunk_size)
first_chunk = next(chunks)

pd.set_option('display.max_columns', None)

first_chunk.head()
```

Out[2]:

	fecha_dato	ncodpers	ind_empleado	pais_residencia	sexo	age	fecha_alta	ind_nuevo	antiguedac
0	2015-01-28	1375586	N	ES	H	35	2015-01-12	0.0	6
1	2015-01-28	1050611	N	ES	V	23	2012-08-10	0.0	35
2	2015-01-28	1050612	N	ES	V	23	2012-08-10	0.0	35
3	2015-01-28	1050613	N	ES	H	22	2012-08-10	0.0	35
4	2015-01-28	1050614	N	ES	V	23	2012-08-10	0.0	35

The above output allows us to identify how the columns are organized and some examples of the cells we will see in the dataset. With this, we can start working on the data types for each column

In [3]:
```
col = ['fecha_dato']
dates = pd.read_csv('train_ver2.csv', usecols=col)
dates['fecha_dato'] = pd.to_datetime(dates['fecha_dato'], errors='coerce')

old = dates.min()
new = dates.max()

print(f'First date = {old} \n Last date = {new}')
```

```
First date = fecha_dato   2015-01-28
dtype: datetime64[ns]
 Last date = fecha_dato   2016-05-28
dtype: datetime64[ns]
```

In [4]:
```
monthly_count = dates.groupby([dates['fecha_dato'].dt.year, dates['fecha_dato'].dt.mor
monthly_count
```

Out[4]:
```
fecha_dato  fecha_dato
2015        1             625457
            2             627394
            3             629209
            4             630367
            5             631957
            6             632110
            7             829817
            8             843201
            9             865440
            10            892251
            11            906109
            12            912021
2016        1             916269
            2             920904
            3             925076
            4             928274
            5             931453
dtype: int64
```

Since the full dataset is over 2BG, we can't upload it in Jupyter, hence here we are exploring the dates range to determine where it will be our cuttoff. The data seems to be well distributed along the months so our cuttoff will be on June of 2016 and our final train dataset will have one year worth of records

Since the dataset is too large to ingest at once, we will use Dask dataframes to process the initial changes. Dask handles datasets larger than the available memory by partitioning the data and processing it in parallel across multiple processors or machines -it works like a pandas dataframe, but with parallel processing

We will export the data as objects so we don't get any dtypes errors for now

In [5]:
```
data = dd.read_csv('train_ver2.csv', assume_missing=True, dtype=object)
```

In [6]:
```
data['fecha_dato'] = dd.to_datetime(data['fecha_dato'], errors='coerce')

cutoff = pd.Timestamp('2015-06-01')

filtered_data = data[data['fecha_dato'] >= cutoff]

rename_col = {
    'fecha_dato': 'date',
    'ncodpers': 'customer_code',
    'ind_empleado': 'employee_index',
    'pais_residencia': 'country',
    'sexo': 'sex_H',
    'age': 'age',
    'fecha_alta': 'first_contract_date',
```

```
    'ind_nuevo': 'new_cust',
    'antiguedad': 'seniority_in_months',
    'indrel': 'primary_cust',
    'ult_fec_cli_1t': 'last_date_primary',
    'indrel_1mes': 'cust_type',
    'tiprel_1mes': 'cust_relationship',
    'indresi': 'residency_spain',
    'indext': 'birth_spain',
    'conyuemp': 'employee_spouse',
    'canal_entrada': 'join_channel',
    'indfall': 'deceased',
    'tipodom': 'address_type',
    'cod_prov': 'province_code',
    'nomprov': 'province_name',
    'ind_actividad_cliente': 'active_cust',
    'renta': 'income',
    'segmento': 'segment',
    'ind_ahor_fin_ult1': 'savings_acct',
    'ind_aval_fin_ult1': 'guarantees',
    'ind_cco_fin_ult1': 'current_acct',
    'ind_cder_fin_ult1': 'derivada_acct',
    'ind_cno_fin_ult1': 'payroll_acct',
    'ind_ctju_fin_ult1': 'junior_acct',
    'ind_ctma_fin_ult1': 'mas_particular_acct',
    'ind_ctop_fin_ult1': 'particular_acct',
    'ind_ctpp_fin_ult1': 'particular_plus_acct',
    'ind_deco_fin_ult1': 'short_term_depo',
    'ind_deme_fin_ult1': 'medium_term_depo',
    'ind_dela_fin_ult1': 'long_term_depo',
    'ind_ecue_fin_ult1': 'e_acct',
    'ind_fond_fin_ult1': 'funds',
    'ind_hip_fin_ult1': 'mortgage',
    'ind_plan_fin_ult1': 'pension',
    'ind_pres_fin_ult1': 'loans',
    'ind_reca_fin_ult1': 'taxes',
    'ind_tjcr_fin_ult1': 'credit_card',
    'ind_valo_fin_ult1': 'securities',
    'ind_viv_fin_ult1': 'home_acct',
    'ind_nomina_ult1': 'payroll_acct',
    'ind_nom_pens_ult1': 'pensions_2',
    'ind_recibo_ult1': 'direct_debt'
}

filtered_data = filtered_data.rename(columns=rename_col)
na_check = filtered_data.compute()
```

Here we are filtering out the early months of the dataset and changing to columns name from Spanish to English for best comprehension.

We then call compute(). Since dask uses a parallel processing, it performs what is called lazy operation, meaning that the changes are not applied to the whole dataset unless it is forced -by using compute(). We want to force it here so we can start seeing null values and other important characteristics of the dataset to strat the cleaning process, which is what we are doing on the next code by seeing what kind of values are on each column and how many null values each column has.

In [7]:
```
list_col = list(rename_col.values())

for clean in list_col:
    print (f"{clean} variables: {na_check[clean].unique()}")
    print(f"NA values: {na_check[clean].isna().sum()}")
```

```
date variables: <DatetimeArray>
['2015-06-28 00:00:00', '2015-07-28 00:00:00', '2015-08-28 00:00:00',
 '2015-09-28 00:00:00', '2015-10-28 00:00:00', '2015-11-28 00:00:00',
 '2015-12-28 00:00:00', '2016-01-28 00:00:00', '2016-02-28 00:00:00',
 '2016-03-28 00:00:00', '2016-04-28 00:00:00', '2016-05-28 00:00:00']
Length: 12, dtype: datetime64[ns]
NA values: 0
customer_code variables: ['  16132' '1063040' '1063041' ... '1173729' '1164094' '1550
586']
NA values: 0
employee_index variables: ['N' nan 'A' 'B' 'F' 'S']
NA values: 1861
country variables: ['ES' nan 'CL' 'NL' 'AT' 'CH' 'CA' 'IE' 'GB' 'AR' 'DE' 'DO' 'BE'
'MX' 'FR'
 'VE' 'QA' 'US' 'HN' 'EC' 'CR' 'CO' 'NI' 'BR' 'PT' 'MZ' 'AL' 'SE' 'IT'
 'PE' 'IN' 'PY' 'MA' 'PL' 'CN' 'FI' 'TW' 'GR' 'AE' 'PR' 'HK' 'RO' 'GT'
 'NO' 'BG' 'GA' 'RU' 'UA' 'SN' 'MR' 'EE' 'SV' 'CZ' 'IL' 'SA' 'CI' 'LU'
 'PA' 'ET' 'CM' 'BA' 'BO' 'HR' 'SG' 'BY' 'NG' 'CU' 'JP' 'SK' 'AU' 'MD'
 'TR' 'KE' 'UY' 'ZA' 'GE' 'DK' 'AD' 'GQ' 'EG' 'DZ' 'TH' 'PK' 'LY' 'TN'
 'TG' 'LB' 'KR' 'KH' 'GH' 'RS' 'KW' 'PH' 'VN' 'AO' 'MM' 'NZ' 'GI' 'LV'
 'SL' 'GN' 'GW' 'CG' 'ML' 'HU' 'MK' 'OM' 'LT' 'IS' 'CD' 'GM' 'KZ' 'CF'
 'BZ' 'ZW' 'DJ' 'JM' 'BM' 'MT']
NA values: 1861
sex_H variables: ['V' 'H' nan]
NA values: 1918
age variables: [' 48' ' 25' ' 24' ' 26' ' 23' ' 22' ' 29' ' 36' ' 32' ' 30' ' 28' ' 5
6'
 ' 27' ' 40' ' 34' ' 63' ' 53' ' 39' ' 60' ' 42' ' 31' ' 41' ' NA' ' 45'
 ' 37' ' 35' ' 57' ' 55' ' 51' ' 58' ' 46' ' 44' ' 50' ' 65' ' 47' ' 75'
 ' 38' ' 49' ' 43' ' 52' '  5' ' 18' ' 13' ' 11' ' 59' ' 33' ' 70' ' 69'
 ' 61' ' 82' ' 68' ' 54' ' 12' ' 67' ' 14' ' 71' ' 77' ' 92' '  6' ' 10'
 '  7' ' 84' ' 73' ' 62' ' 95' ' 17' ' 87' ' 15' ' 72' ' 64' ' 21' ' 66'
 ' 85' ' 83' ' 16' '  8' ' 20' ' 86' '  9' ' 19' ' 79' ' 74' ' 80' ' 96'
 ' 81' ' 89' ' 90' ' 78' ' 88' '100' ' 76' ' 91' ' 94' ' 93' ' 98' '  4'
 ' 97' '104' '106' '101' '103' ' 99' '  3' '  2' '102' '107' '111' '109'
 '105' '110' '112' '115' '108' '116' '113' '126' '117' '163' '127' '114'
 '164']
NA values: 0
first_contract_date variables: ['1995-03-08' '2012-09-19' nan ... '2016-05-25' '2016-
05-01' '2016-05-15']
NA values: 1861
new_cust variables: [' 0' nan ' 1']
NA values: 1861
seniority_in_months variables: ['    244' '     34' '     NA' '     25' '     33' '
22' '     10'
 '     21' '      9' '     17' '     12' '     20' '     30' '     18'
 '      2' '      5' '     24' '      6' '     27' '     19' '      8'
 '     13' '     32' '      7' '     11' '     28' '     15' '     35'
 '     16' '     23' '      3' '     26' '     31' '      1' '      4'
 '     29' '    157' '     14' '     36' '     40' '    139' '     46'
 '     38' '     45' '     44' '     43' '     41' '     39' '     47'
 '     42' '     37' '     49' '     50' '     48' '     51' '     56'
 '     54' '     55' '      0' '     53' '     52' '     57' '     58'
 '    209' '    165' '    164' '    105' '     81' '    129' '    109'
 '    128' '    108' '    156' '    121' '    136' '    150' '    142'
 '     64' '    122' '    125' '    146' '    138' '    101' '     69'
 '    163' '    116' '     96' '    117' '    107' '    137' '    145'
 '     61' '    162' '    160' '    102' '     88' '     65' '    114'
 '    113' '    161' '    217' '     77' '    154' '    152' '    126'
 '    159' '    166' '    104' '    119' '     94' '    149' '    103'
```

```
 '     82' '     76' '    151' '     70' '     86' '     79' '    135'
 '    169' '     60' '    118' '    134' '    120' '    110' '    148'
 '     78' '    141' '     66' '    140' '     99' '    147' '    100'
 '     95' '    133' '    124' '    127' '    193' '     80' '    132'
 '     83' '    123' '    231' '    158' '    143' '    187' '    111'
 '     85' '     98' '    170' '    106' '     84' '     63' '    155'
 '    189' '    175' '     87' '    177' '    115' '    112' '    232'
 '     97' '    144' '     93' '    203' '    131' '    172' '    190'
 '     72' '    176' '    153' '     89' '    174' '    194' '     71'
 '    173' '    212' '     68' '     59' '     74' '    130' '     73'
 '    183' '    180' '     62' '    216' '    179' '    178' '    168'
 '    184' '    171' '    167' '    198' '     92' '    199' '    206'
 '    235' '    213' '    208' '     75' '    195' '    201' '    186'
 '     67' '    188' '     90' '    207' '    185' '    192' '    182'
 '     91' '    215' '    211' '    181' '    196' '    219' '    205'
 '    202' '    200' '    214' '    191' '    227' '    218' '    225'
 '    224' '    226' '    242' '    210' '    223' '    237' '    222'
 '    204' '    233' '    220' '    228' '    197' '    221' '    241'
 '    229' '    240' '    234' '    243' '    230' '    238' '    246'
 '    236' '    239' '    245' '-999999' '    247' '    248' '    249'
 '    250' '    251' '    252' '    253' '    254' '    255' '    256']
NA values: 0
primary_cust variables: [' 1' nan '99']
NA values: 1861
last_date_primary variables: [nan '2015-07-13' '2015-07-29' '2015-07-30' '2015-07-23'
'2015-07-06'
 '2015-07-03' '2015-07-01' '2015-07-21' '2015-07-14' '2015-07-10'
 '2015-07-27' '2015-07-16' '2015-07-15' '2015-07-08' '2015-07-20'
 '2015-07-07' '2015-07-09' '2015-07-02' '2015-07-28' '2015-07-22'
 '2015-07-17' '2015-07-24' '2015-08-21' '2015-08-19' '2015-08-25'
 '2015-08-14' '2015-08-24' '2015-08-17' '2015-08-18' '2015-08-10'
 '2015-08-13' '2015-08-27' '2015-08-03' '2015-08-06' '2015-08-20'
 '2015-08-26' '2015-08-28' '2015-08-05' '2015-08-11' '2015-08-07'
 '2015-08-04' '2015-08-12' '2015-09-17' '2015-09-01' '2015-09-18'
 '2015-09-03' '2015-09-02' '2015-09-14' '2015-09-16' '2015-09-29'
 '2015-09-28' '2015-09-09' '2015-09-22' '2015-09-08' '2015-09-11'
 '2015-09-21' '2015-09-04' '2015-09-25' '2015-09-07' '2015-09-10'
 '2015-09-23' '2015-09-24' '2015-09-15' '2015-10-08' '2015-10-07'
 '2015-10-13' '2015-10-26' '2015-10-29' '2015-10-05' '2015-10-28'
 '2015-10-09' '2015-10-22' '2015-10-20' '2015-10-15' '2015-10-06'
 '2015-10-01' '2015-10-21' '2015-10-16' '2015-10-27' '2015-10-19'
 '2015-10-23' '2015-10-02' '2015-10-14' '2015-11-23' '2015-11-24'
 '2015-11-12' '2015-11-04' '2015-11-13' '2015-11-25' '2015-11-19'
 '2015-11-20' '2015-11-03' '2015-11-16' '2015-11-17' '2015-11-11'
 '2015-11-27' '2015-11-18' '2015-11-10' '2015-11-26' '2015-11-02'
 '2015-11-05' '2015-11-06' '2015-11-09' '2015-12-21' '2015-12-18'
 '2015-12-28' '2015-12-24' '2015-12-04' '2015-12-29' '2015-12-16'
 '2015-12-11' '2015-12-30' '2015-12-15' '2015-12-01' '2015-12-09'
 '2015-12-10' '2015-12-17' '2015-12-02' '2015-12-14' '2015-12-03'
 '2015-12-22' '2015-12-23' '2015-12-07' '2016-01-08' '2016-01-14'
 '2016-01-13' '2016-01-28' '2016-01-05' '2016-01-19' '2016-01-12'
 '2016-01-18' '2016-01-21' '2016-01-22' '2016-01-07' '2016-01-20'
 '2016-01-26' '2016-01-15' '2016-01-27' '2016-01-25' '2016-01-11'
 '2016-01-04' '2016-02-23' '2016-02-19' '2016-02-18' '2016-02-26'
 '2016-02-12' '2016-02-24' '2016-02-09' '2016-02-08' '2016-02-11'
 '2016-02-05' '2016-02-04' '2016-02-03' '2016-02-15' '2016-02-22'
 '2016-02-10' '2016-02-16' '2016-02-01' '2016-02-17' '2016-02-02'
 '2016-02-25' '2016-03-07' '2016-03-29' '2016-03-10' '2016-03-18'
 '2016-03-14' '2016-03-22' '2016-03-08' '2016-03-21' '2016-03-30'
 '2016-03-01' '2016-03-23' '2016-03-02' '2016-03-24' '2016-03-03'
```

```
 '2016-03-09' '2016-03-11' '2016-03-04' '2016-03-16' '2016-03-28'
 '2016-03-15' '2016-03-17' '2016-04-22' '2016-04-01' '2016-04-06'
 '2016-04-12' '2016-04-05' '2016-04-15' '2016-04-13' '2016-04-19'
 '2016-04-04' '2016-04-18' '2016-04-26' '2016-04-11' '2016-04-25'
 '2016-04-27' '2016-04-08' '2016-04-07' '2016-04-21' '2016-04-28'
 '2016-04-20' '2016-04-14' '2016-05-23' '2016-05-05' '2016-05-17'
 '2016-05-19' '2016-05-12' '2016-05-06' '2016-05-03' '2016-05-20'
 '2016-05-02' '2016-05-16' '2016-05-18' '2016-05-04' '2016-05-13'
 '2016-05-24' '2016-05-27' '2016-05-10' '2016-05-30' '2016-05-25'
 '2016-05-11' '2016-05-09' '2016-05-26']
NA values: 10482071
cust_type variables: ['1' nan '1.0' '3.0' '2.0' '3' '4.0' 'P' '4' '2']
NA values: 123908
cust_relationship variables: ['A' 'I' nan 'P' 'R' 'N']
NA values: 123908
residency_spain variables: ['S' nan 'N']
NA values: 1861
birth_spain variables: ['N' 'S' nan]
NA values: 1861
employee_spouse variables: [nan 'N' 'S']
NA values: 10501557
join_channel variables: ['KAT' 'KHE' 'KHD' nan 'KFC' 'KFA' 'KHC' 'KAZ' 'KHK' 'KHL' 'K
GN' 'RED'
 'KHN' 'KDH' 'KEH' 'KGC' 'KHM' 'KHO' 'KHF' 'KFK' 'KHA' 'KAF' 'K00' '013'
 'KAR' 'KFJ' 'KAG' 'KAA' 'KFF' 'KAI' 'KCC' 'KFG' 'KFP' 'KFD' 'KGX' 'KAH'
 'KAE' 'KFS' 'KAB' 'KFN' 'KAP' 'KFL' 'KFU' 'KGY' 'KAQ' 'KGV' 'KAJ' 'KAD'
 'KBG' 'KHQ' 'KAK' '007' 'KDR' 'KCA' 'KDT' 'KBO' 'KBQ' 'KAY' 'KCG' 'KBU'
 'KBZ' '004' 'KDO' 'KCK' 'KEC' 'KAC' 'KEU' 'KDE' 'KDY' 'KCH' 'KCI' 'KCL'
 'KDA' 'KES' 'KAS' 'KDX' 'KCM' 'KCN' 'KDQ' 'KCB' 'KDU' 'KAL' 'KAW' 'KEY'
 'KDZ' 'KCS' 'KCD' 'KCE' 'KEJ' 'KDC' 'KBL' 'KAO' 'KEA' 'KEW' 'KFT' 'KEV'
 'KBH' 'KEG' 'KEI' 'KEO' 'KBD' 'KDP' 'KBV' 'KCO' 'KBR' 'KCV' 'KBF' 'KCU'
 'KBX' 'KDD' 'KBW' 'KCF' 'KAN' 'KEZ' 'KAM' 'KDS' 'KBY' 'KEF' 'KBS' 'KDF'
 'KCP' 'KDB' 'KBP' 'KBE' 'KCT' 'KCX' 'KBN' 'KDV' 'KDG' 'KEB' 'KEL' 'KDW'
 'KBB' 'KBJ' 'KDM' 'KFH' 'KBM' 'KEN' 'KFI' 'KEQ' 'KAV' 'KFM' 'KAU' 'KED'
 'KEK' 'KFR' 'KFB' 'KFE' 'KGW' 'KFV' 'KGU' 'KDI' 'KEE' 'KCQ' 'KCR' 'KDN'
 'KEM' 'KCJ' 'KDL' '025' 'KHP' 'KHR' 'KHS']
NA values: 160057
deceased variables: ['N' nan 'S']
NA values: 1861
address_type variables: [' 1' nan]
NA values: 1862
province_code variables: ['28' '46' '23' '11' '36' '15' '33' '29' '21' '41' ' 2' '12'
'14' '50'
 '27' '30' ' 6' ' 7' '45' '24' ' 3' '25' '18' '32' ' 5' '37' '44' nan ' 8'
 '39' '10' '43' '34' '35' ' 9' '13' '22' '31' '38' '20' '52' ' 1' '19'
 '26' '42' ' 4' '17' '47' '16' '49' '48' '51' '40']
NA values: 49330
province_name variables: ['MADRID' 'VALENCIA' 'JAEN' 'CADIZ' 'PONTEVEDRA' 'CORUÑA, A'
'ASTURIAS'
 'MALAGA' 'HUELVA' 'SEVILLA' 'ALBACETE' 'CASTELLON' 'CORDOBA' 'ZARAGOZA'
 'LUGO' 'MURCIA' 'BADAJOZ' 'BALEARS, ILLES' 'TOLEDO' 'LEON' 'ALICANTE'
 'LERIDA' 'GRANADA' 'OURENSE' 'AVILA' 'SALAMANCA' 'TERUEL' nan 'BARCELONA'
 'CANTABRIA' 'CACERES' 'TARRAGONA' 'PALENCIA' 'PALMAS, LAS' 'BURGOS'
 'CIUDAD REAL' 'HUESCA' 'NAVARRA' 'SANTA CRUZ DE TENERIFE' 'GIPUZKOA'
 'MELILLA' 'ALAVA' 'GUADALAJARA' 'RIOJA, LA' 'SORIA' 'ALMERIA' 'GIRONA'
 'VALLADOLID' 'CUENCA' 'ZAMORA' 'BIZKAIA' 'CEUTA' 'SEGOVIA']
NA values: 49330
active_cust variables: [' 0' ' 1' nan]
NA values: 1861
income variables: ['160900.95' '74693.67' '35053.770000000004' ... '63867.66' '34341.
```

```
18'
 '89018.37']
NA values: 2240788
segment variables: ['02 - PARTICULARES' '03 - UNIVERSITARIO' nan '01 - TOP']
NA values: 163256
savings_acct variables: ['0' '1']
NA values: 0
guarantees variables: ['0' '1']
NA values: 0
current_acct variables: ['1' '0']
NA values: 0
derivada_acct variables: ['0' '1']
NA values: 0
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_14116\483060620.py in ?()
      1 list_col = list(rename_col.values())
      2 
      3 for clean in list_col:
----> 4     print (f"{clean} variables: {na_check[clean].unique()}")
      5     print(f"NA values: {na_check[clean].isna().sum()}")

c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core
\generic.py in ?(self, name)
   6295             and name not in self._accessors
   6296             and self._info_axis._can_hold_identifiers_and_holds_name(name)
   6297         ):
   6298             return self[name]
-> 6299         return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'unique'
```

In [8]:
```
filtered_data = filtered_data.drop(['province_code', 'address_type', 'employee_spouse'
```

After analyzing the previous output, we can see that the address_type column has only one value across the whole database, which is '1' (and null), so the column will be irrelevant to any future modeling. The column province_code has the same information as province_name, so we will drop the code one and keep the names. Lastly, the column employee_spouse has too many null values -over 10M, so we will drop it because it does not make sense to fill in those values since it is most of the database

In [9]:
```
filtered_data = filtered_data.loc[filtered_data['sex_H'].notnull()]

other = ['join_channel', 'province_name']
filtered_data[other] = filtered_data[other].fillna('other')

filtered_data['sex_H'] = filtered_data['sex_H'].map({'H': 1, 'V': 0}).fillna(0)

columns_to_dummy = ['residency_spain', 'birth_spain', 'deceased']
for col in columns_to_dummy:
    filtered_data[col] = filtered_data[col].map({'S': 1, 'N': 0}).fillna(0)

trim = ['customer_code', 'age', 'new_cust', 'seniority_in_months', 'primary_cust']
for col in trim:
    filtered_data[col] = filtered_data[col].astype(str).str.strip()
```

```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\dask_expr\_c
ollection.py:4192: UserWarning:
You did not provide metadata, so Dask is running your function on a small dataset to
guess output types. It is possible that Dask will guess incorrectly.
To provide an explicit output types or to silence this message, please provide the `m
eta=` keyword, as described in the map or apply function that you are using.
  Before: .apply(func)
  After:  .apply(func, meta=('sex_H', 'float64'))

  warnings.warn(meta_warning(meta))
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\dask_expr\_c
ollection.py:4192: UserWarning:
You did not provide metadata, so Dask is running your function on a small dataset to
guess output types. It is possible that Dask will guess incorrectly.
To provide an explicit output types or to silence this message, please provide the `m
eta=` keyword, as described in the map or apply function that you are using.
  Before: .apply(func)
  After:  .apply(func, meta=('residency_spain', 'float64'))

  warnings.warn(meta_warning(meta))
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\dask_expr\_c
ollection.py:4192: UserWarning:
You did not provide metadata, so Dask is running your function on a small dataset to
guess output types. It is possible that Dask will guess incorrectly.
To provide an explicit output types or to silence this message, please provide the `m
eta=` keyword, as described in the map or apply function that you are using.
  Before: .apply(func)
  After:  .apply(func, meta=('birth_spain', 'float64'))

  warnings.warn(meta_warning(meta))
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\dask_expr\_c
ollection.py:4192: UserWarning:
You did not provide metadata, so Dask is running your function on a small dataset to
guess output types. It is possible that Dask will guess incorrectly.
To provide an explicit output types or to silence this message, please provide the `m
eta=` keyword, as described in the map or apply function that you are using.
  Before: .apply(func)
  After:  .apply(func, meta=('deceased', 'float64'))

  warnings.warn(meta_warning(meta))
```

Above we are starting the cleaning process of the dataset. First we dropped the null values on the sex columns. We see the number 1861 repeating a lot across columns, so we will drop the null in sex and check if the other nulls will be drop. Since those nulls are across many columns, we concluded it would be best to drop them.

We filled null in columns join_channel and province_name with 'other' since they are string variables

We transformed the columns sex, residency_spain, birth_spain and deceased to dummy variables and filled na with 0

Lastly, we trimmed the cells on columns customer_code, age, new_cust, seniority_in_months, and primary_cust for cleanliness

In [11]:
```
products = ['savings_acct', 'guarantees', 'current_acct', 'derivada_acct', 'payroll_ac
            'junior_acct', 'mas_particular_acct', 'particular_acct', 'particular_plus_
            'short_term_depo', 'medium_term_depo', 'long_term_depo', 'e_acct', 'funds'
            'mortgage', 'pension', 'loans', 'taxes', 'credit_card', 'securities',
            'home_acct', 'payroll_acct', 'pensions_2', 'direct_debt']

filtered_data = filtered_data.fillna('0')
```

For now we will fill the rest of na values with '0' so we can keep cleaning the data. Later we will go back to these values and determine if the best approach is to fill it with '0'

In [12]:
```
dtype_mapping = {
    'customer_code': 'int',
    'employee_index': 'str',
    'country': 'str',
    'sex_H': 'str',
    'age': 'int',
    'first_contract_date': 'datetime64[ns]',
    'new_cust': 'int',
    'seniority_in_months': 'int',
    'primary_cust': 'int',
    'last_date_primary': 'object',
    'cust_type': 'object',
    'cust_relationship': 'str',
    'residency_spain': 'str',
    'birth_spain': 'str',
    'join_channel': 'str',
    'deceased': 'str',
    'province_name': 'str',
    'active_cust': 'int',
    'income': 'float',
    'segment': 'object',
    'savings_acct': 'int',
    'guarantees': 'int',
    'current_acct': 'int',
    'derivada_acct': 'int',
    'payroll_acct': 'int',
    'junior_acct': 'int',
    'mas_particular_acct': 'int',
    'particular_acct': 'int',
    'particular_plus_acct': 'int',
    'short_term_depo': 'int',
    'medium_term_depo': 'int',
    'long_term_depo': 'int',
    'e_acct': 'int',
    'funds': 'int',
    'mortgage': 'int',
    'pension': 'int',
    'loans': 'int',
    'taxes': 'int',
    'credit_card': 'int',
    'securities': 'int',
    'home_acct': 'int',
    'payroll_acct': 'int',
    'pensions_2': 'int',
    'direct_debt': 'int'
}
filtered_data = filtered_data.astype(dtype_mapping)
```

Now we will tranform the dtypes across the whole dataset and call compute() again to force all the above changes across the whole dataset

In [13]:
```
filtered_data = filtered_data.compute()
```

In [15]:
```
#Export data to a CSV file
filtered_data.to_csv('train_final.csv', index=False)
```

In [17]:
```
filtered_data.shape
```

Out[17]:
```
(10501007, 45)
```

In [18]:
```
filtered_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10501007 entries, 27973 to 391600
Data columns (total 45 columns):
 #   Column                 Dtype         
---  ------                 -----         
 0   date                   datetime64[ns]
 1   customer_code          int32         
 2   employee_index         object        
 3   country                object        
 4   sex_H                  object        
 5   age                    int32         
 6   first_contract_date    datetime64[ns]
 7   new_cust               int32         
 8   seniority_in_months    int32         
 9   primary_cust           int32         
 10  last_date_primary      object        
 11  cust_type              object        
 12  cust_relationship      object        
 13  residency_spain        object        
 14  birth_spain            object        
 15  join_channel           object        
 16  deceased               object        
 17  province_name          object        
 18  active_cust            int32         
 19  income                 float64       
 20  segment                object        
 21  savings_acct           int32         
 22  guarantees             int32         
 23  current_acct           int32         
 24  derivada_acct          int32         
 25  payroll_acct           int32         
 26  junior_acct            int32         
 27  mas_particular_acct    int32         
 28  particular_acct        int32         
 29  particular_plus_acct   int32         
 30  short_term_depo        int32         
 31  medium_term_depo       int32         
 32  long_term_depo         int32         
 33  e_acct                 int32         
 34  funds                  int32         
 35  mortgage               int32         
 36  pension                int32         
 37  loans                  int32         
 38  taxes                  int32         
 39  credit_card            int32         
 40  securities             int32         
 41  home_acct              int32         
 42  payroll_acct           int32         
 43  pensions_2             int32         
 44  direct_debt            int32         
dtypes: datetime64[ns](2), float64(1), int32(30), object(12)
memory usage: 2.4+ GB
```

End of Week 2