# Week 13 — Bring it all together

## Loading and cleaning the data

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
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import category_encoders as ce
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from datetime import datetime
from sklearn.decomposition import PCA
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import roc_auc_score, f1_score, confusion_matrix
from sklearn.linear_model import LogisticRegression
from collections import defaultdict
from joblib import Parallel, delayed
from lime.lime_tabular import LimeTabularExplainer
import lime
import lime.lime_tabular
import random
```

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)
```

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]:

```
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 [8]:

```
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 [9]:
```
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 [10]:
```
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 [11]:
```
train = filtered_data.compute()
```

In [12]:
```
pd.set_option('display.max_columns', None)
train['total_products'] = train[products].sum(axis=1)
```

# NA values treatment

In [13]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10501007 entries, 27973 to 391600
Data columns (total 46 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
 45  total_products        int64
dtypes: datetime64[ns](2), float64(1), int32(30), int64(1), object(12)
memory usage: 2.5+ GB
```

In [14]:
```
#Changing dates columns to datetime type

dates = ['date', 'first_contract_date']
train[dates] = train[dates].apply(pd.to_datetime)
```

In [15]:
```
pd.set_option('display.max_rows', None)

count_col = ['primary_cust', 'last_date_primary', 'deceased', 'seniority_in_months', '

for col in count_col:
    count = train[col].value_counts()
    percentage = (count/count.sum()*100).round(2)
    products_bought = train.groupby(col)['total_products'].sum()
    summary = pd.DataFrame({'Count': count, 'Percentage':percentage, 'Products Owned':
    print(summary)
```

```
              Count  Percentage  Products Owned
primary_cust
1          10480153        99.8        15183312
99            20854         0.2            1928
                  Count  Percentage  Products Owned
last_date_primary
0              10480153       99.80        15183312
2015-07-01          142        0.00              44
2015-07-02           63        0.00              17
2015-07-03           95        0.00              34
2015-07-06          138        0.00              38
2015-07-07          112        0.00              58
2015-07-08           65        0.00              27
2015-07-09          148        0.00              43
2015-07-10          117        0.00              59
2015-07-13           81        0.00              37
2015-07-14           72        0.00              26
2015-07-15           91        0.00              47
2015-07-16           57        0.00              20
2015-07-17          111        0.00              70
2015-07-20          110        0.00              46
2015-07-21          130        0.00              52
2015-07-22          102        0.00              55
2015-07-23           78        0.00              31
2015-07-24           97        0.00              52
2015-07-27           83        0.00              29
2015-07-28          115        0.00              54
2015-07-29          104        0.00              43
2015-07-30           96        0.00              34
2015-08-03           96        0.00               0
2015-08-04           69        0.00               0
2015-08-05           65        0.00               3
2015-08-06           41        0.00               1
2015-08-07           59        0.00               3
2015-08-10           82        0.00               3
2015-08-11           81        0.00               4
2015-08-12           62        0.00               3
2015-08-13           78        0.00               2
2015-08-14           51        0.00               2
2015-08-17           75        0.00               1
2015-08-18           59        0.00               6
2015-08-19           44        0.00               6
2015-08-20           59        0.00               5
2015-08-21           61        0.00               0
2015-08-24           61        0.00               2
2015-08-25           63        0.00               1
2015-08-26           68        0.00               5
2015-08-27           85        0.00               9
2015-08-28           83        0.00               3
2015-09-01           97        0.00               3
2015-09-02           71        0.00               6
2015-09-03           61        0.00               2
2015-09-04           83        0.00              11
2015-09-07           79        0.00               4
2015-09-08          107        0.00              11
2015-09-09           70        0.00               0
2015-09-10           65        0.00               3
2015-09-11           79        0.00               6
2015-09-14          114        0.00               6
2015-09-15           82        0.00               9
```

| | | | |
|---|---|---|---|
| 2015-09-16 | 82 | 0.00 | 2 |
| 2015-09-17 | 90 | 0.00 | 9 |
| 2015-09-18 | 111 | 0.00 | 8 |
| 2015-09-21 | 57 | 0.00 | 5 |
| 2015-09-22 | 108 | 0.00 | 8 |
| 2015-09-23 | 84 | 0.00 | 3 |
| 2015-09-24 | 84 | 0.00 | 0 |
| 2015-09-25 | 73 | 0.00 | 4 |
| 2015-09-28 | 70 | 0.00 | 17 |
| 2015-09-29 | 66 | 0.00 | 7 |
| 2015-10-01 | 115 | 0.00 | 2 |
| 2015-10-02 | 94 | 0.00 | 5 |
| 2015-10-05 | 129 | 0.00 | 2 |
| 2015-10-06 | 88 | 0.00 | 0 |
| 2015-10-07 | 112 | 0.00 | 10 |
| 2015-10-08 | 87 | 0.00 | 6 |
| 2015-10-09 | 91 | 0.00 | 9 |
| 2015-10-13 | 102 | 0.00 | 7 |
| 2015-10-14 | 89 | 0.00 | 5 |
| 2015-10-15 | 113 | 0.00 | 4 |
| 2015-10-16 | 82 | 0.00 | 9 |
| 2015-10-19 | 110 | 0.00 | 2 |
| 2015-10-20 | 93 | 0.00 | 2 |
| 2015-10-21 | 77 | 0.00 | 3 |
| 2015-10-22 | 93 | 0.00 | 0 |
| 2015-10-23 | 90 | 0.00 | 1 |
| 2015-10-26 | 131 | 0.00 | 3 |
| 2015-10-27 | 108 | 0.00 | 11 |
| 2015-10-28 | 125 | 0.00 | 16 |
| 2015-10-29 | 83 | 0.00 | 5 |
| 2015-11-02 | 128 | 0.00 | 14 |
| 2015-11-03 | 84 | 0.00 | 3 |
| 2015-11-04 | 100 | 0.00 | 6 |
| 2015-11-05 | 66 | 0.00 | 0 |
| 2015-11-06 | 54 | 0.00 | 0 |
| 2015-11-09 | 96 | 0.00 | 1 |
| 2015-11-10 | 89 | 0.00 | 2 |
| 2015-11-11 | 93 | 0.00 | 2 |
| 2015-11-12 | 76 | 0.00 | 16 |
| 2015-11-13 | 90 | 0.00 | 3 |
| 2015-11-16 | 109 | 0.00 | 6 |
| 2015-11-17 | 80 | 0.00 | 5 |
| 2015-11-18 | 110 | 0.00 | 8 |
| 2015-11-19 | 85 | 0.00 | 4 |
| 2015-11-20 | 103 | 0.00 | 2 |
| 2015-11-23 | 94 | 0.00 | 2 |
| 2015-11-24 | 109 | 0.00 | 9 |
| 2015-11-25 | 86 | 0.00 | 17 |
| 2015-11-26 | 61 | 0.00 | 3 |
| 2015-11-27 | 91 | 0.00 | 11 |
| 2015-12-01 | 104 | 0.00 | 6 |
| 2015-12-02 | 79 | 0.00 | 2 |
| 2015-12-03 | 80 | 0.00 | 4 |
| 2015-12-04 | 68 | 0.00 | 5 |
| 2015-12-07 | 64 | 0.00 | 3 |
| 2015-12-09 | 90 | 0.00 | 1 |
| 2015-12-10 | 72 | 0.00 | 12 |
| 2015-12-11 | 88 | 0.00 | 3 |
| 2015-12-14 | 100 | 0.00 | 5 |
| 2015-12-15 | 76 | 0.00 | 3 |

| | | | |
|---|---|---|---|
| 2015-12-16 | 158 | 0.00 | 7 |
| 2015-12-17 | 172 | 0.00 | 2 |
| 2015-12-18 | 139 | 0.00 | 9 |
| 2015-12-21 | 206 | 0.00 | 11 |
| 2015-12-22 | 71 | 0.00 | 1 |
| 2015-12-23 | 27 | 0.00 | 2 |
| 2015-12-24 | 763 | 0.01 | 6 |
| 2015-12-28 | 521 | 0.00 | 8 |
| 2015-12-29 | 99 | 0.00 | 8 |
| 2015-12-30 | 96 | 0.00 | 4 |
| 2016-01-04 | 34 | 0.00 | 0 |
| 2016-01-05 | 167 | 0.00 | 4 |
| 2016-01-07 | 108 | 0.00 | 2 |
| 2016-01-08 | 107 | 0.00 | 10 |
| 2016-01-11 | 90 | 0.00 | 0 |
| 2016-01-12 | 78 | 0.00 | 7 |
| 2016-01-13 | 122 | 0.00 | 12 |
| 2016-01-14 | 89 | 0.00 | 3 |
| 2016-01-15 | 105 | 0.00 | 6 |
| 2016-01-18 | 93 | 0.00 | 7 |
| 2016-01-19 | 169 | 0.00 | 6 |
| 2016-01-20 | 74 | 0.00 | 3 |
| 2016-01-21 | 96 | 0.00 | 6 |
| 2016-01-22 | 111 | 0.00 | 3 |
| 2016-01-25 | 94 | 0.00 | 2 |
| 2016-01-26 | 109 | 0.00 | 7 |
| 2016-01-27 | 112 | 0.00 | 9 |
| 2016-01-28 | 99 | 0.00 | 9 |
| 2016-02-01 | 121 | 0.00 | 2 |
| 2016-02-02 | 96 | 0.00 | 5 |
| 2016-02-03 | 65 | 0.00 | 9 |
| 2016-02-04 | 74 | 0.00 | 10 |
| 2016-02-05 | 67 | 0.00 | 3 |
| 2016-02-08 | 107 | 0.00 | 4 |
| 2016-02-09 | 96 | 0.00 | 2 |
| 2016-02-10 | 79 | 0.00 | 10 |
| 2016-02-11 | 96 | 0.00 | 17 |
| 2016-02-12 | 93 | 0.00 | 18 |
| 2016-02-15 | 129 | 0.00 | 4 |
| 2016-02-16 | 91 | 0.00 | 4 |
| 2016-02-17 | 73 | 0.00 | 16 |
| 2016-02-18 | 64 | 0.00 | 4 |
| 2016-02-19 | 70 | 0.00 | 3 |
| 2016-02-22 | 100 | 0.00 | 4 |
| 2016-02-23 | 85 | 0.00 | 5 |
| 2016-02-24 | 88 | 0.00 | 11 |
| 2016-02-25 | 65 | 0.00 | 7 |
| 2016-02-26 | 101 | 0.00 | 8 |
| 2016-03-01 | 98 | 0.00 | 3 |
| 2016-03-02 | 84 | 0.00 | 0 |
| 2016-03-03 | 59 | 0.00 | 3 |
| 2016-03-04 | 89 | 0.00 | 5 |
| 2016-03-07 | 55 | 0.00 | 5 |
| 2016-03-08 | 72 | 0.00 | 1 |
| 2016-03-09 | 99 | 0.00 | 3 |
| 2016-03-10 | 82 | 0.00 | 10 |
| 2016-03-11 | 84 | 0.00 | 5 |
| 2016-03-14 | 97 | 0.00 | 6 |
| 2016-03-15 | 96 | 0.00 | 13 |
| 2016-03-16 | 76 | 0.00 | 6 |

```
2016-03-17            66        0.00                5
2016-03-18            77        0.00                6
2016-03-21            74        0.00                9
2016-03-22            69        0.00                4
2016-03-23            46        0.00                1
2016-03-24            49        0.00                3
2016-03-28            83        0.00                8
2016-03-29            74        0.00               11
2016-03-30            79        0.00                4
2016-04-01           132        0.00                4
2016-04-04            86        0.00                2
2016-04-05            89        0.00                2
2016-04-06            71        0.00                2
2016-04-07            58        0.00                0
2016-04-08            87        0.00                7
2016-04-11           101        0.00                3
2016-04-12            96        0.00                6
2016-04-13            82        0.00                2
2016-04-14            57        0.00                3
2016-04-15            88        0.00                4
2016-04-18            78        0.00                7
2016-04-19            88        0.00                2
2016-04-20            63        0.00                2
2016-04-21            76        0.00                3
2016-04-22            62        0.00                4
2016-04-25            77        0.00                1
2016-04-26            79        0.00                2
2016-04-27            74        0.00                0
2016-04-28            44        0.00                2
2016-05-02           128        0.00               12
2016-05-03            65        0.00                2
2016-05-04            83        0.00                5
2016-05-05            61        0.00                2
2016-05-06            99        0.00                4
2016-05-09            77        0.00                0
2016-05-10            78        0.00                9
2016-05-11            74        0.00                3
2016-05-12            73        0.00                6
2016-05-13            55        0.00                3
2016-05-16            89        0.00                4
2016-05-17            84        0.00                0
2016-05-18            92        0.00               11
2016-05-19           111        0.00                1
2016-05-20            84        0.00                6
2016-05-23            83        0.00                3
2016-05-24           124        0.00               13
2016-05-25            75        0.00                3
2016-05-26           128        0.00                5
2016-05-27           109        0.00                5
2016-05-30            98        0.00                3
             Count  Percentage  Products Owned
deceased
0         10474146       99.74        15154952
1            26861        0.26           30288
                      Count  Percentage  Products Owned
seniority_in_months
-999999                  28        0.00             173
 0                   134357        1.28           81977
 1                   132477        1.26          116322
 2                   128049        1.22          121988
```

| | | | |
|---|---|---|---|
| 3 | 128348 | 1.22 | 125068 |
| 4 | 120189 | 1.14 | 118461 |
| 5 | 125917 | 1.20 | 123711 |
| 6 | 114032 | 1.09 | 113076 |
| 7 | 112509 | 1.07 | 111993 |
| 8 | 107713 | 1.03 | 110579 |
| 9 | 105700 | 1.01 | 108015 |
| 10 | 121312 | 1.16 | 122779 |
| 11 | 95961 | 0.91 | 98046 |
| 12 | 149217 | 1.42 | 152003 |
| 13 | 110316 | 1.05 | 113314 |
| 14 | 115248 | 1.10 | 118015 |
| 15 | 110974 | 1.06 | 114746 |
| 16 | 113837 | 1.08 | 122250 |
| 17 | 112308 | 1.07 | 119066 |
| 18 | 107755 | 1.03 | 120180 |
| 19 | 99779 | 0.95 | 113503 |
| 20 | 98493 | 0.94 | 114129 |
| 21 | 115992 | 1.10 | 132966 |
| 22 | 103688 | 0.99 | 115446 |
| 23 | 109845 | 1.05 | 123607 |
| 24 | 114855 | 1.09 | 128260 |
| 25 | 100606 | 0.96 | 112778 |
| 26 | 105067 | 1.00 | 119014 |
| 27 | 98761 | 0.94 | 110583 |
| 28 | 97947 | 0.93 | 108623 |
| 29 | 94570 | 0.90 | 105597 |
| 30 | 91540 | 0.87 | 104116 |
| 31 | 87694 | 0.84 | 101541 |
| 32 | 83742 | 0.80 | 100037 |
| 33 | 95070 | 0.91 | 111565 |
| 34 | 88816 | 0.85 | 104911 |
| 35 | 90189 | 0.86 | 104846 |
| 36 | 103372 | 0.98 | 118865 |
| 37 | 92019 | 0.88 | 106671 |
| 38 | 95849 | 0.91 | 111690 |
| 39 | 90396 | 0.86 | 104765 |
| 40 | 94975 | 0.90 | 109068 |
| 41 | 92266 | 0.88 | 106534 |
| 42 | 85560 | 0.81 | 99886 |
| 43 | 93260 | 0.89 | 108551 |
| 44 | 90608 | 0.86 | 105577 |
| 45 | 95837 | 0.91 | 110918 |
| 46 | 90374 | 0.86 | 104599 |
| 47 | 83273 | 0.79 | 95488 |
| 48 | 87697 | 0.84 | 99366 |
| 49 | 81310 | 0.77 | 90667 |
| 50 | 81877 | 0.78 | 90542 |
| 51 | 75641 | 0.72 | 83595 |
| 52 | 80410 | 0.77 | 88300 |
| 53 | 78248 | 0.75 | 85370 |
| 54 | 67857 | 0.65 | 74965 |
| 55 | 57137 | 0.54 | 64327 |
| 56 | 42483 | 0.40 | 48686 |
| 57 | 30535 | 0.29 | 36846 |
| 58 | 23325 | 0.22 | 29897 |
| 59 | 18519 | 0.18 | 24882 |
| 60 | 18589 | 0.18 | 25053 |
| 61 | 20235 | 0.19 | 28479 |
| 62 | 22311 | 0.21 | 31582 |

| | | | |
|---|---|---|---|
| 63 | 24035 | 0.23 | 35113 |
| 64 | 25026 | 0.24 | 37564 |
| 65 | 23366 | 0.22 | 35804 |
| 66 | 23877 | 0.23 | 37479 |
| 67 | 23289 | 0.22 | 37337 |
| 68 | 22430 | 0.21 | 38440 |
| 69 | 22089 | 0.21 | 37973 |
| 70 | 20651 | 0.20 | 36243 |
| 71 | 19851 | 0.19 | 35488 |
| 72 | 18972 | 0.18 | 35240 |
| 73 | 16070 | 0.15 | 31453 |
| 74 | 13331 | 0.13 | 27433 |
| 75 | 11721 | 0.11 | 24285 |
| 76 | 12367 | 0.12 | 24814 |
| 77 | 14122 | 0.13 | 26437 |
| 78 | 16120 | 0.15 | 28748 |
| 79 | 17066 | 0.16 | 29085 |
| 80 | 19928 | 0.19 | 33636 |
| 81 | 27153 | 0.26 | 42995 |
| 82 | 26567 | 0.25 | 39847 |
| 83 | 28216 | 0.27 | 40405 |
| 84 | 30605 | 0.29 | 42774 |
| 85 | 30923 | 0.29 | 42611 |
| 86 | 33366 | 0.32 | 44435 |
| 87 | 33912 | 0.32 | 43678 |
| 88 | 34946 | 0.33 | 42555 |
| 89 | 34669 | 0.33 | 41404 |
| 90 | 34391 | 0.33 | 40700 |
| 91 | 32990 | 0.31 | 37119 |
| 92 | 30092 | 0.29 | 32686 |
| 93 | 32329 | 0.31 | 34731 |
| 94 | 30613 | 0.29 | 32837 |
| 95 | 31754 | 0.30 | 34199 |
| 96 | 33503 | 0.32 | 36463 |
| 97 | 32987 | 0.31 | 36191 |
| 98 | 33943 | 0.32 | 37536 |
| 99 | 31364 | 0.30 | 36075 |
| 100 | 31312 | 0.30 | 36690 |
| 101 | 31940 | 0.30 | 37828 |
| 102 | 34352 | 0.33 | 41527 |
| 103 | 33307 | 0.32 | 41804 |
| 104 | 33722 | 0.32 | 43894 |
| 105 | 35477 | 0.34 | 47725 |
| 106 | 32983 | 0.31 | 44369 |
| 107 | 34338 | 0.33 | 46587 |
| 108 | 34511 | 0.33 | 47806 |
| 109 | 33252 | 0.32 | 46724 |
| 110 | 36763 | 0.35 | 52030 |
| 111 | 35059 | 0.33 | 49947 |
| 112 | 35845 | 0.34 | 51794 |
| 113 | 33510 | 0.32 | 50053 |
| 114 | 34692 | 0.33 | 52448 |
| 115 | 33684 | 0.32 | 51963 |
| 116 | 31886 | 0.30 | 49951 |
| 117 | 36185 | 0.34 | 57659 |
| 118 | 34030 | 0.32 | 55238 |
| 119 | 34252 | 0.33 | 56383 |
| 120 | 33865 | 0.32 | 56710 |
| 121 | 31267 | 0.30 | 53219 |
| 122 | 31519 | 0.30 | 54724 |

| | | | |
|---|---|---|---|
| 123 | 31063 | 0.30 | 54828 |
| 124 | 32338 | 0.31 | 57399 |
| 125 | 32208 | 0.31 | 58029 |
| 126 | 32269 | 0.31 | 58805 |
| 127 | 32473 | 0.31 | 59557 |
| 128 | 29976 | 0.29 | 56365 |
| 129 | 29509 | 0.28 | 55015 |
| 130 | 28768 | 0.27 | 53983 |
| 131 | 27157 | 0.26 | 51241 |
| 132 | 28615 | 0.27 | 53606 |
| 133 | 28906 | 0.28 | 54328 |
| 134 | 31085 | 0.30 | 58177 |
| 135 | 28451 | 0.27 | 52765 |
| 136 | 30146 | 0.29 | 56736 |
| 137 | 29556 | 0.28 | 55062 |
| 138 | 29931 | 0.29 | 55915 |
| 139 | 30079 | 0.29 | 56607 |
| 140 | 29953 | 0.29 | 56680 |
| 141 | 30348 | 0.29 | 57298 |
| 142 | 30189 | 0.29 | 56765 |
| 143 | 29914 | 0.28 | 55922 |
| 144 | 30203 | 0.29 | 56514 |
| 145 | 28389 | 0.27 | 54244 |
| 146 | 29192 | 0.28 | 56880 |
| 147 | 27382 | 0.26 | 53382 |
| 148 | 27150 | 0.26 | 53796 |
| 149 | 25470 | 0.24 | 51473 |
| 150 | 26667 | 0.25 | 53886 |
| 151 | 27286 | 0.26 | 55566 |
| 152 | 26150 | 0.25 | 54770 |
| 153 | 27156 | 0.26 | 57283 |
| 154 | 25813 | 0.25 | 54536 |
| 155 | 24703 | 0.24 | 52207 |
| 156 | 29772 | 0.28 | 62329 |
| 157 | 28565 | 0.27 | 59058 |
| 158 | 29615 | 0.28 | 61403 |
| 159 | 33162 | 0.32 | 67825 |
| 160 | 35951 | 0.34 | 72419 |
| 161 | 37581 | 0.36 | 74462 |
| 162 | 42914 | 0.41 | 85902 |
| 163 | 43631 | 0.42 | 86304 |
| 164 | 46647 | 0.44 | 90975 |
| 165 | 52452 | 0.50 | 101572 |
| 166 | 51356 | 0.49 | 100128 |
| 167 | 46816 | 0.45 | 90710 |
| 168 | 50527 | 0.48 | 99020 |
| 169 | 50698 | 0.48 | 99628 |
| 170 | 50209 | 0.48 | 98689 |
| 171 | 47537 | 0.45 | 93851 |
| 172 | 48847 | 0.47 | 98613 |
| 173 | 43700 | 0.42 | 87924 |
| 174 | 42631 | 0.41 | 87811 |
| 175 | 38755 | 0.37 | 81405 |
| 176 | 35140 | 0.33 | 76629 |
| 177 | 35750 | 0.34 | 79771 |
| 178 | 34457 | 0.33 | 76997 |
| 179 | 32634 | 0.31 | 73477 |
| 180 | 33582 | 0.32 | 76695 |
| 181 | 28600 | 0.27 | 66528 |
| 182 | 28840 | 0.27 | 68492 |

| | | | |
|---|---|---|---|
| 183 | 25310 | 0.24 | 62131 |
| 184 | 26273 | 0.25 | 65260 |
| 185 | 25488 | 0.24 | 63722 |
| 186 | 24352 | 0.23 | 60973 |
| 187 | 23605 | 0.22 | 58832 |
| 188 | 22642 | 0.22 | 56207 |
| 189 | 22337 | 0.21 | 56357 |
| 190 | 20292 | 0.19 | 51181 |
| 191 | 19287 | 0.18 | 48781 |
| 192 | 19366 | 0.18 | 48895 |
| 193 | 19869 | 0.19 | 51452 |
| 194 | 20261 | 0.19 | 52468 |
| 195 | 18705 | 0.18 | 49615 |
| 196 | 17680 | 0.17 | 47779 |
| 197 | 16993 | 0.16 | 46622 |
| 198 | 17586 | 0.17 | 48122 |
| 199 | 18342 | 0.17 | 51064 |
| 200 | 17627 | 0.17 | 49049 |
| 201 | 18539 | 0.18 | 51508 |
| 202 | 17789 | 0.17 | 49215 |
| 203 | 16843 | 0.16 | 46210 |
| 204 | 16329 | 0.16 | 44543 |
| 205 | 16346 | 0.16 | 44697 |
| 206 | 17183 | 0.16 | 47186 |
| 207 | 16146 | 0.15 | 44498 |
| 208 | 17219 | 0.16 | 46870 |
| 209 | 16647 | 0.16 | 46004 |
| 210 | 15686 | 0.15 | 42298 |
| 211 | 16630 | 0.16 | 44712 |
| 212 | 15878 | 0.15 | 42485 |
| 213 | 15637 | 0.15 | 41310 |
| 214 | 15534 | 0.15 | 41376 |
| 215 | 15098 | 0.14 | 40048 |
| 216 | 14908 | 0.14 | 39225 |
| 217 | 14439 | 0.14 | 37596 |
| 218 | 13876 | 0.13 | 35589 |
| 219 | 12341 | 0.12 | 31643 |
| 220 | 12460 | 0.12 | 32061 |
| 221 | 11783 | 0.11 | 30059 |
| 222 | 10694 | 0.10 | 26716 |
| 223 | 10979 | 0.10 | 27441 |
| 224 | 10241 | 0.10 | 24997 |
| 225 | 10871 | 0.10 | 26060 |
| 226 | 9914 | 0.09 | 23724 |
| 227 | 9295 | 0.09 | 22195 |
| 228 | 9523 | 0.09 | 23029 |
| 229 | 9118 | 0.09 | 21646 |
| 230 | 8979 | 0.09 | 21126 |
| 231 | 9218 | 0.09 | 21590 |
| 232 | 8585 | 0.08 | 20016 |
| 233 | 7646 | 0.07 | 17678 |
| 234 | 7839 | 0.07 | 18501 |
| 235 | 7999 | 0.08 | 18630 |
| 236 | 7135 | 0.07 | 16643 |
| 237 | 8632 | 0.08 | 20004 |
| 238 | 8271 | 0.08 | 19242 |
| 239 | 7329 | 0.07 | 17083 |
| 240 | 7490 | 0.07 | 17878 |
| 241 | 7302 | 0.07 | 17696 |
| 242 | 6467 | 0.06 | 15687 |

| | | | |
|---|---|---|---|
| 243 | 5919 | 0.06 | 14406 |
| 244 | 5553 | 0.05 | 13744 |
| 245 | 4618 | 0.04 | 11569 |
| 246 | 4170 | 0.04 | 10778 |
| 247 | 3516 | 0.03 | 9130 |
| 248 | 2271 | 0.02 | 6169 |
| 249 | 1777 | 0.02 | 5132 |
| 250 | 1512 | 0.01 | 4415 |
| 251 | 1071 | 0.01 | 3195 |
| 252 | 676 | 0.01 | 2204 |
| 253 | 416 | 0.00 | 1504 |
| 254 | 261 | 0.00 | 992 |
| 255 | 179 | 0.00 | 748 |
| 256 | 102 | 0.00 | 453 |

| | Count | Percentage | Products Owned |
|---|---|---|---|
| province_name | | | |
| ALAVA | 29200 | 0.28 | 40388 |
| ALBACETE | 87764 | 0.84 | 99725 |
| ALICANTE | 245070 | 2.33 | 309253 |
| ALMERIA | 47225 | 0.45 | 64018 |
| ASTURIAS | 204362 | 1.95 | 263752 |
| AVILA | 29799 | 0.28 | 37045 |
| BADAJOZ | 146712 | 1.40 | 161279 |
| BALEARS, ILLES | 99314 | 0.95 | 130005 |
| BARCELONA | 1003756 | 9.56 | 1232054 |
| BIZKAIA | 143195 | 1.36 | 194627 |
| BURGOS | 75113 | 0.72 | 94985 |
| CACERES | 97769 | 0.93 | 108450 |
| CADIZ | 223411 | 2.13 | 291240 |
| CANTABRIA | 120634 | 1.15 | 172272 |
| CASTELLON | 80360 | 0.77 | 101618 |
| CEUTA | 5625 | 0.05 | 8560 |
| CIUDAD REAL | 91585 | 0.87 | 109976 |
| CORDOBA | 110061 | 1.05 | 136763 |
| CORUÑA, A | 327217 | 3.12 | 380436 |
| CUENCA | 43382 | 0.41 | 49994 |
| GIPUZKOA | 55402 | 0.53 | 76597 |
| GIRONA | 71236 | 0.68 | 78678 |
| GRANADA | 138206 | 1.32 | 171753 |
| GUADALAJARA | 51584 | 0.49 | 74208 |
| HUELVA | 91738 | 0.87 | 107627 |
| HUESCA | 31147 | 0.30 | 38012 |
| JAEN | 49857 | 0.47 | 69255 |
| LEON | 64259 | 0.61 | 85635 |
| LERIDA | 61791 | 0.59 | 62619 |
| LUGO | 64531 | 0.61 | 71615 |
| MADRID | 3393023 | 32.31 | 6253191 |
| MALAGA | 278115 | 2.65 | 368903 |
| MELILLA | 7259 | 0.07 | 10959 |
| MURCIA | 305733 | 2.91 | 334259 |
| NAVARRA | 68567 | 0.65 | 84739 |
| OURENSE | 63638 | 0.61 | 71735 |
| PALENCIA | 37833 | 0.36 | 44971 |
| PALMAS, LAS | 181645 | 1.73 | 259049 |
| PONTEVEDRA | 213724 | 2.04 | 249444 |
| RIOJA, LA | 66041 | 0.63 | 81762 |
| SALAMANCA | 125293 | 1.19 | 147729 |
| SANTA CRUZ DE TENERIFE | 55740 | 0.53 | 83804 |
| SEGOVIA | 32669 | 0.31 | 45053 |
| SEVILLA | 459492 | 4.38 | 657771 |

```
SORIA                    13848      0.13       18076
TARRAGONA                80871      0.77       95511
TERUEL                   17237      0.16       20330
TOLEDO                  141582      1.35      183010
VALENCIA                534921      5.09      685129
VALLADOLID              182831      1.74      230665
ZAMORA                   38903      0.37       44246
ZARAGOZA                263268      2.51      327414
other                    47469      0.45       65051
```

#### Analyzing columns: Last Date Primary and Primary Customer

In [16]:
```
print(train['last_date_primary'].value_counts())
print(train['primary_cust'].value_counts())
```

```
last_date_primary
0             10480153
2015-12-24         763
2015-12-28         521
2015-12-21         206
2015-12-17         172
2016-01-19         169
2016-01-05         167
2015-12-16         158
2015-07-09         148
2015-07-01         142
2015-12-18         139
2015-07-06         138
2016-04-01         132
2015-10-26         131
2015-07-21         130
2016-02-15         129
2015-10-05         129
2016-05-02         128
2015-11-02         128
2016-05-26         128
2015-10-28         125
2016-05-24         124
2016-01-13         122
2016-02-01         121
2015-07-10         117
2015-10-01         115
2015-07-28         115
2015-09-14         114
2015-10-15         113
2016-01-27         112
2015-07-07         112
2015-10-07         112
2016-05-19         111
2015-09-18         111
2015-07-17         111
2016-01-22         111
2015-11-18         110
2015-07-20         110
2015-10-19         110
2015-11-16         109
2015-11-24         109
2016-01-26         109
2016-05-27         109
2015-10-27         108
2016-01-07         108
2015-09-22         108
2015-09-08         107
2016-02-08         107
2016-01-08         107
2016-01-15         105
2015-07-29         104
2015-12-01         104
2015-11-20         103
2015-10-13         102
2015-07-22         102
2016-04-11         101
2016-02-26         101
2016-02-22         100
2015-11-04         100
```

```
2015-12-14         100
2016-03-09          99
2016-05-06          99
2016-01-28          99
2015-12-29          99
2016-05-30          98
2016-03-01          98
2016-03-14          97
2015-07-24          97
2015-09-01          97
2015-11-09          96
2016-02-11          96
2016-01-21          96
2016-02-09          96
2016-03-15          96
2015-07-30          96
2016-04-12          96
2016-02-02          96
2015-12-30          96
2015-08-03          96
2015-07-03          95
2015-10-02          94
2015-11-23          94
2016-01-25          94
2016-02-12          93
2015-10-20          93
2015-11-11          93
2015-10-22          93
2016-01-18          93
2016-05-18          92
2016-02-16          91
2015-11-27          91
2015-07-15          91
2015-10-09          91
2015-09-17          90
2015-11-13          90
2016-01-11          90
2015-10-23          90
2015-12-09          90
2015-10-14          89
2015-11-10          89
2016-03-04          89
2016-01-14          89
2016-05-16          89
2016-04-05          89
2016-04-19          88
2016-04-15          88
2015-12-11          88
2016-02-24          88
2015-10-06          88
2016-04-08          87
2015-10-08          87
2015-11-25          86
2016-04-04          86
2016-02-23          85
2015-08-27          85
2015-11-19          85
2016-05-17          84
2015-09-23          84
2015-09-24          84
```

```
2016-03-02           84
2016-03-11           84
2015-11-03           84
2016-05-20           84
2015-10-29           83
2016-03-28           83
2015-09-04           83
2015-08-28           83
2016-05-23           83
2016-05-04           83
2015-07-27           83
2015-09-16           82
2015-08-10           82
2016-03-10           82
2016-04-13           82
2015-09-15           82
2015-10-16           82
2015-08-11           81
2015-07-13           81
2015-11-17           80
2015-12-03           80
2015-09-11           79
2016-02-10           79
2016-03-30           79
2015-12-02           79
2015-09-07           79
2016-04-26           79
2016-04-18           78
2015-07-23           78
2016-01-12           78
2016-05-10           78
2015-08-13           78
2016-03-18           77
2015-10-21           77
2016-04-25           77
2016-05-09           77
2015-12-15           76
2016-04-21           76
2015-11-12           76
2016-03-16           76
2015-08-17           75
2016-05-25           75
2016-04-27           74
2016-03-21           74
2016-05-11           74
2016-03-29           74
2016-01-20           74
2016-02-04           74
2016-02-17           73
2016-05-12           73
2015-09-25           73
2015-07-14           72
2015-12-10           72
2016-03-08           72
2016-04-06           71
2015-12-22           71
2015-09-02           71
2015-09-28           70
2016-02-19           70
2015-09-09           70
```

```
2016-03-22         69
2015-08-04         69
2015-12-04         68
2015-08-26         68
2016-02-05         67
2015-09-29         66
2015-11-05         66
2016-03-17         66
2015-09-10         65
2016-05-03         65
2015-07-08         65
2016-02-25         65
2015-08-05         65
2016-02-03         65
2015-12-07         64
2016-02-18         64
2015-07-02         63
2016-04-20         63
2015-08-25         63
2016-04-22         62
2015-08-12         62
2015-08-24         61
2015-08-21         61
2015-11-26         61
2015-09-03         61
2016-05-05         61
2015-08-20         59
2015-08-07         59
2015-08-18         59
2016-03-03         59
2016-04-07         58
2015-07-16         57
2016-04-14         57
2015-09-21         57
2016-05-13         55
2016-03-07         55
2015-11-06         54
2015-08-14         51
2016-03-24         49
2016-03-23         46
2016-04-28         44
2015-08-19         44
2015-08-06         41
2016-01-04         34
2015-12-23         27
Name: count, dtype: int64
primary_cust
1     10480153
99       20854
Name: count, dtype: int64
```

In [17]:

```
non_primary = train[train['primary_cust'] == 99]
non_primary['total_products'].sum()
```

Out[17]:

```
1928
```

0 dates on last_date_primary mean they are still primary customers

We will drop the column primary customer and keep the last date as primary customer since we can have all the information from one column -customers that do not have a date are still

primary
We will keep the non-primary customers since they still own products of the bank

**Analyzing column: Deceased**

In [18]:
```
train.groupby('deceased')[products].sum()
```

Out[18]:

| deceased | savings_acct | guarantees | current_acct | derivada_acct | payroll_acct | payroll_acct | junior_acct |
|---|---|---|---|---|---|---|---|
| **0** | 958 | 214 | 6488365 | 3821 | 539367 | 539367 | 90875 |
| **1** | 0 | 0 | 12798 | 12 | 59 | 59 | 0 |

Deceased clients have a few products, and they make up 0.2% of the database. Deceased clients are not going to buy any more products, so we will drop the rows of deceased clients and drop the column

**Analyzing column: Seniority in Months**
We will drop the rows with value -999999

**Analyzing column: Province Name**
We will drop the rows where province name = others

**Analyzing column: Age**
We will drop columns where age is 0 and over 100. We believe these clients are not going to be valuable on our model

**Analyzing column: Income**

In [19]:
```
(train['income'] == 0).sum()
```

Out[19]:
```
2238903
```

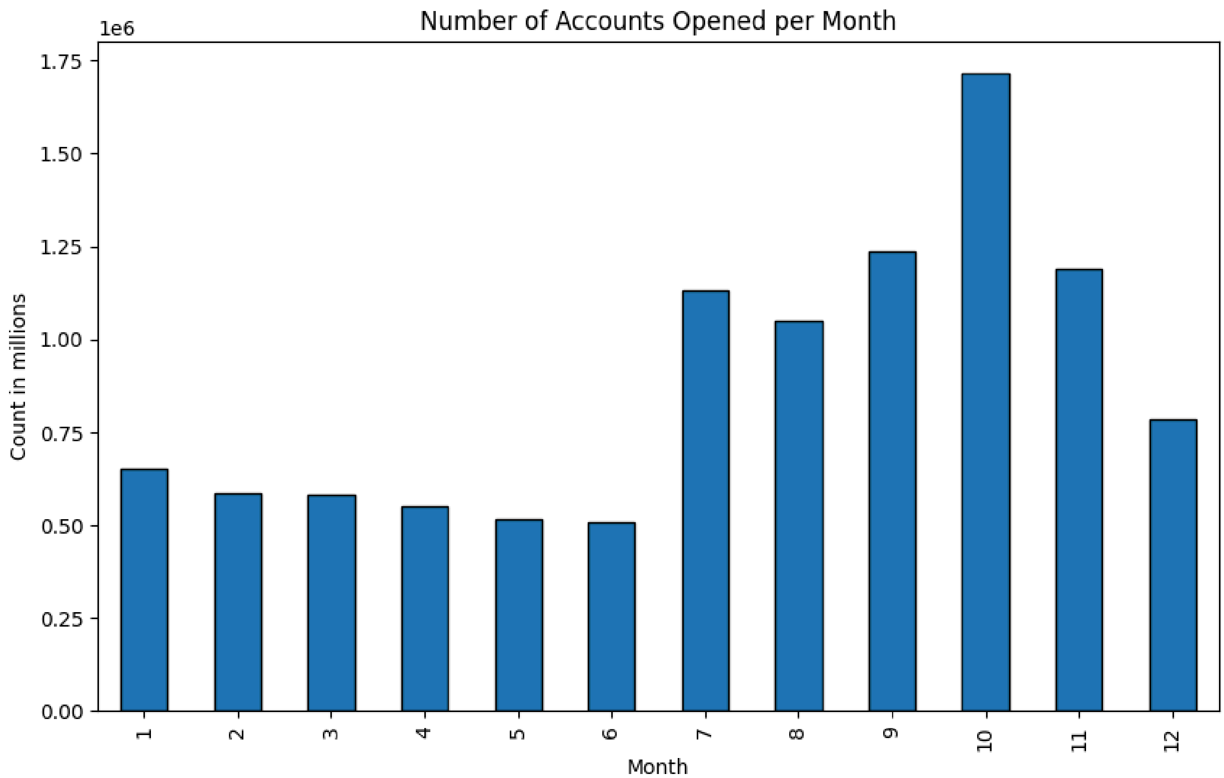
**Analyzing columns: Seniority in Months and First Contract Date**
We had the impression these two columns were giving us the same information and we decided to check the correlation. Turns out it is highly correlated, so we'll keep the column seniority in months.
Before we delete the first contract date column, we want to extract any information we might need from it.

In [20]:
```
correlation = train[['seniority_in_months', 'first_contract_date']].corr()
print(correlation)
```

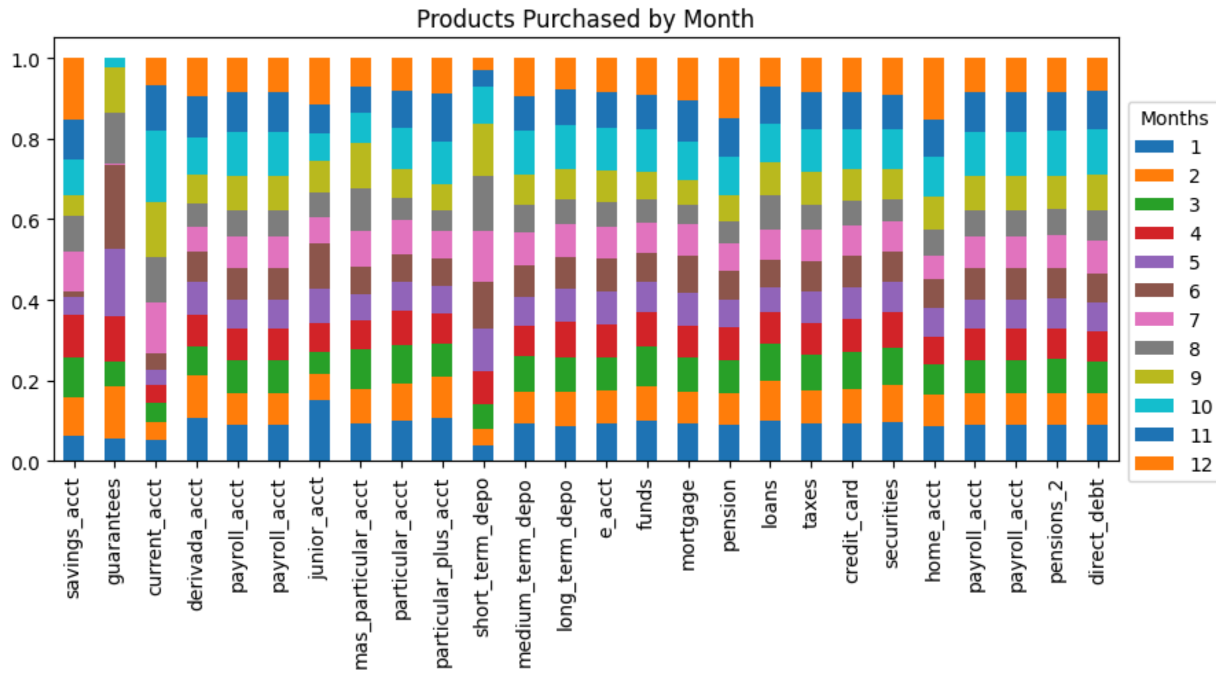
```
                     seniority_in_months  first_contract_date
seniority_in_months              1.00000             -0.03798
first_contract_date             -0.03798              1.00000
```

In [21]:
```
month_count = train['first_contract_date'].dt.month.value_counts().sort_index()

month_count.plot(kind='bar', figsize=(10, 6), edgecolor='black')
plt.title('Number of Accounts Opened per Month')
plt.xlabel('Month')
plt.ylabel('Count in millions')
```

Out[21]:
```
Text(0, 0.5, 'Count in millions')
```




In [22]:
```
months = train['first_contract_date'].dt.month
dummy = train.groupby(months)[products].sum()
dummy = (dummy/dummy.sum()).T
ax = dummy.plot(kind='bar', stacked=True, figsize=(10,4))
plt.legend(loc='center left', title='Months', bbox_to_anchor=(1, .4))
plt.title('Products Purchased by Month')
plt.show()
```




When we plot contract dates by month to see if there is any seasonality, we can conclude that the last half of the year has significantly more new contracts than the first half. This can maybe be explained by raises, bonus, new people getting hired after summer, and other factors
We don't see any patterns on product purchases for a specific month. Looks like the increase on purchases in the second half of the year is an overall increase

# Splitting the dataset into train and validation

We will use the 80/20 ratio

In [23]:
```
train, temp = train_test_split(train, test_size=0.3, random_state=42)

val_set, test_set = train_test_split(temp, test_size=0.5, random_state=42)
```

In [24]:
```
map_dtypes = {
    "date": "datetime64[ns]",
    "customer_code": "int32",
    "employee_index": "object",
    "country": "object",
    "sex_H": "object",
    "age": "int32",
    "first_contract_date": "datetime64[ns]",
    "new_cust": "int32",
    "seniority_in_months": "int32",
    "primary_cust": "int32",
    "last_date_primary": "object",
    "cust_type": "object",
    "cust_relationship": "object",
    "residency_spain": "object",
    "birth_spain": "object",
    "join_channel": "object",
    "deceased": "object",
    "province_name": "object",
```

```
    "active_cust": "int32",
    "income": "float64",
    "segment": "object",
    "savings_acct": "int32",
    "guarantees": "int32",
    "current_acct": "int32",
    "derivada_acct": "int32",
    "payroll_acct": "int32",
    "junior_acct": "int32",
    "mas_particular_acct": "int32",
    "particular_acct": "int32",
    "particular_plus_acct": "int32",
    "short_term_depo": "int32",
    "medium_term_depo": "int32",
    "long_term_depo": "int32",
    "e_acct": "int32",
    "funds": "int32",
    "mortgage": "int32",
    "pension": "int32",
    "loans": "int32",
    "taxes": "int32",
    "credit_card": "int32",
    "securities": "int32",
    "home_acct": "int32",
    "payroll_acct": "int32",
    "pensions_2": "int32",
    "direct_debt": "int32",
    "total_products": "int64"
}

test_set = test_set.astype(map_dtypes)
val_set = val_set.astype(map_dtypes)
```

# EDA

In [25]:

```
train.describe().round(2)
```

Out[25]:

| | date | customer_code | age | first_contract_date | new_cust | seniority_in_mon |
|---|---|---|---|---|---|---|
| **count** | 7350704 | 7350704.00 | 7350704.00 | 7350704 | 7350704.00 | 735070 |
| **mean** | 2015-12-20 06:48:44.553838080 | 850982.41 | 40.10 | 2009-04-12 15:48:38.383001088 | 0.07 | 76 |
| **min** | 2015-06-28 00:00:00 | 15889.00 | 2.00 | 1995-01-16 00:00:00 | 0.00 | -99999 |
| **25%** | 2015-09-28 00:00:00 | 464328.00 | 24.00 | 2004-07-06 00:00:00 | 0.00 | 22 |
| **50%** | 2015-12-28 00:00:00 | 943521.00 | 39.00 | 2011-09-26 00:00:00 | 0.00 | 50 |
| **75%** | 2016-03-28 00:00:00 | 1220509.00 | 50.00 | 2013-11-22 00:00:00 | 0.00 | 133 |
| **max** | 2016-05-28 00:00:00 | 1553687.00 | 164.00 | 2016-05-31 00:00:00 | 1.00 | 256 |
| **std** | NaN | 437214.98 | 17.23 | NaN | 0.26 | 138 |

From the descriptive statistics table we can see:

- There are more man than women in the dataset
- The average and median age in the dataset is 40 years old
- There is a good range of seniority in the dataset, ranging from 0 to 256 months, or 21.3 years
- Most of the clients in the dataset have their primary residency and birth place in Spain

  ### Counting values, percentage and product bought for variables on columns employee_index, country, primary_customer, residency_spain, and deceased

In [26]:

```
pd.set_option('display.max_rows', None)

count_col = ['employee_index', 'country', 'residency_spain', 'birth_spain', 'primary_c

for col in count_col:
    count = train[col].value_counts()
    percentage = (count/count.sum()*100).round(2)
    products_bought = train.groupby(col)['total_products'].sum()
    summary = pd.DataFrame({'Count': count, 'Percentage':percentage, 'Products Owned':
    print(summary)
```

| employee_index | Count | Percentage | Products Owned |
|---|---|---|---|
| A | 1231 | 0.02 | 8614 |
| B | 1811 | 0.02 | 7018 |
| F | 1288 | 0.02 | 5008 |
| N | 7346369 | 99.94 | 10613093 |
| S | 5 | 0.00 | 50 |

| country | Count | Percentage | Products Owned |
|---|---|---|---|
| AD | 55 | 0.00 | 150 |
| AE | 121 | 0.00 | 278 |
| AL | 7 | 0.00 | 7 |
| AO | 32 | 0.00 | 101 |
| AR | 2441 | 0.03 | 3467 |
| AT | 221 | 0.00 | 413 |
| AU | 224 | 0.00 | 421 |
| BA | 16 | 0.00 | 6 |
| BE | 750 | 0.01 | 1164 |
| BG | 237 | 0.00 | 190 |
| BM | 6 | 0.00 | 6 |
| BO | 780 | 0.01 | 706 |
| BR | 1131 | 0.02 | 1235 |
| BY | 49 | 0.00 | 59 |
| BZ | 8 | 0.00 | 8 |
| CA | 230 | 0.00 | 361 |
| CD | 6 | 0.00 | 6 |
| CF | 8 | 0.00 | 8 |
| CG | 14 | 0.00 | 24 |
| CH | 1018 | 0.01 | 2279 |
| CI | 27 | 0.00 | 54 |
| CL | 502 | 0.01 | 642 |
| CM | 45 | 0.00 | 84 |
| CN | 268 | 0.00 | 418 |
| CO | 1755 | 0.02 | 1697 |
| CR | 79 | 0.00 | 123 |
| CU | 377 | 0.01 | 328 |
| CZ | 46 | 0.00 | 30 |
| DE | 2287 | 0.03 | 3948 |
| DJ | 9 | 0.00 | 0 |
| DK | 118 | 0.00 | 147 |
| DO | 202 | 0.00 | 210 |
| DZ | 45 | 0.00 | 37 |
| EC | 1084 | 0.01 | 880 |
| EE | 23 | 0.00 | 18 |
| EG | 39 | 0.00 | 79 |
| ES | 7317736 | 99.55 | 10588843 |
| ET | 17 | 0.00 | 20 |
| FI | 172 | 0.00 | 215 |
| FR | 2562 | 0.03 | 3582 |
| GA | 21 | 0.00 | 62 |
| GB | 2337 | 0.03 | 4210 |
| GE | 7 | 0.00 | 0 |
| GH | 7 | 0.00 | 0 |
| GI | 7 | 0.00 | 14 |
| GM | 8 | 0.00 | 0 |
| GN | 21 | 0.00 | 21 |
| GQ | 57 | 0.00 | 71 |
| GR | 124 | 0.00 | 147 |
| GT | 69 | 0.00 | 39 |
| GW | 17 | 0.00 | 17 |

```
HK          25        0.00            80
HN         147        0.00            86
HR          31        0.00            13
HU          23        0.00            11
IE         201        0.00           293
IL         205        0.00           250
IN          97        0.00           134
IS           7        0.00             7
IT        1506        0.02          1861
JM           8        0.00             0
JP         133        0.00           238
KE          35        0.00           114
KH           9        0.00            18
KR          43        0.00           145
KW          10        0.00            10
KZ          11        0.00            11
LB           6        0.00             0
LT          22        0.00            25
LU          62        0.00           124
LV           8        0.00             8
LY           9        0.00             9
MA         204        0.00           390
MD          44        0.00            35
MK          26        0.00            17
ML           9        0.00             9
MM           7        0.00            42
MR          26        0.00            35
MX        1271        0.02          1784
MZ          18        0.00            18
NG         104        0.00            94
NI          31        0.00            24
NL         384        0.01           763
NO          69        0.00           143
NZ          21        0.00            21
OM          11        0.00            72
PA          34        0.00            78
PE         451        0.01           484
PH          17        0.00            17
PK          43        0.00            43
PL         297        0.00           349
PR          45        0.00            95
PT         721        0.01           953
PY         725        0.01           500
QA          35        0.00            70
RO        1475        0.02           883
RS          21        0.00            21
RU         368        0.01           444
SA          41        0.00            45
SE         315        0.00           449
SG          61        0.00           141
SK          42        0.00            34
SL           9        0.00             9
SN          38        0.00            47
SV          56        0.00            50
TG           6        0.00            18
TH          50        0.00            49
TN          10        0.00             0
TR          34        0.00           121
TW          16        0.00            16
UA         236        0.00           174
```

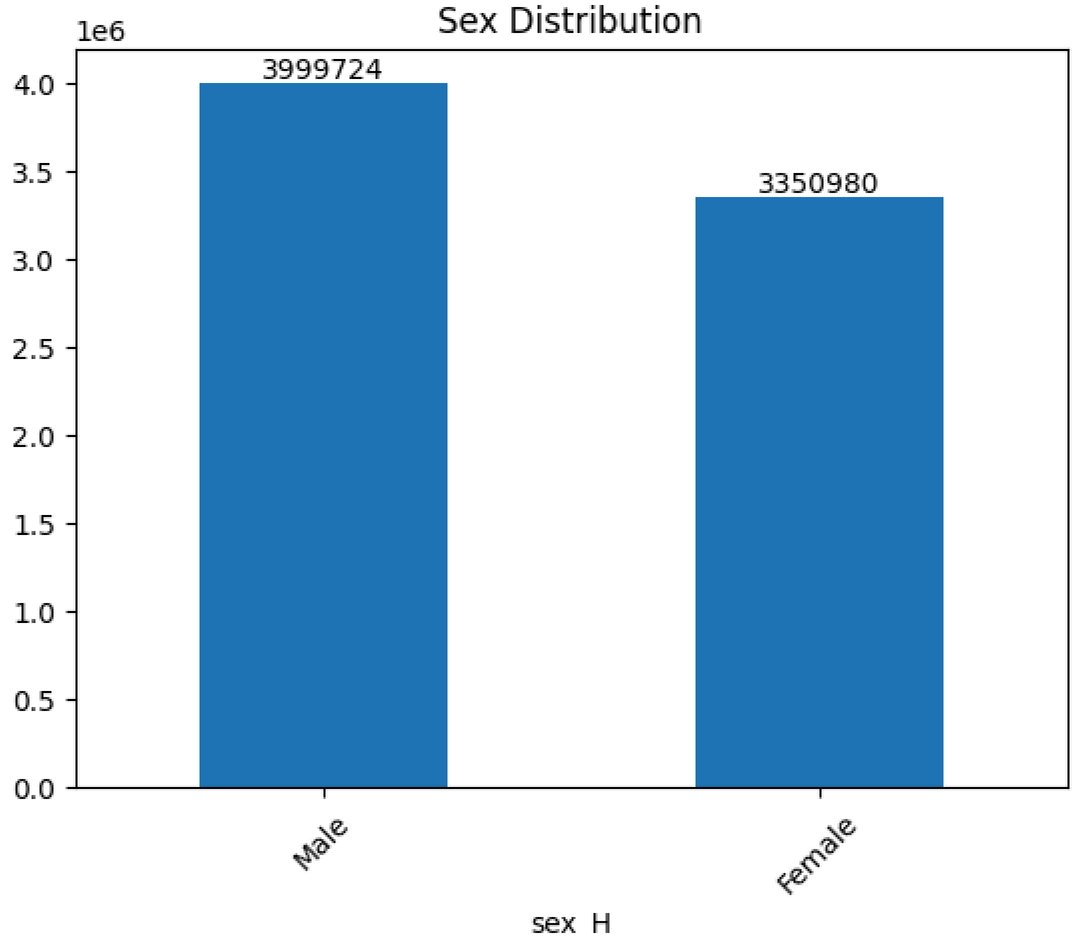
```
US          1829        0.02            2854
UY           252        0.00             371
VE          1148        0.02            1599
VN            18        0.00              39
ZA            58        0.00             121
ZW             8        0.00               0
                   Count  Percentage  Products Owned
residency_spain
0                  32967        0.45           44936
1                7317737       99.55        10588847
               Count  Percentage  Products Owned
birth_spain
0            6995757       95.17        10206257
1             354947        4.83          427526
                Count  Percentage  Products Owned
primary_cust
1             7336142        99.8        10632421
99              14562         0.2            1362
```

Most of the dataset is composed of clients from Spain and non-employees.
Even thought non-spanish and employees clients are the absolute minority in the database, it still can be an important factor for these specific clients when predicting which products they will buy

## Analyzing column: Sex

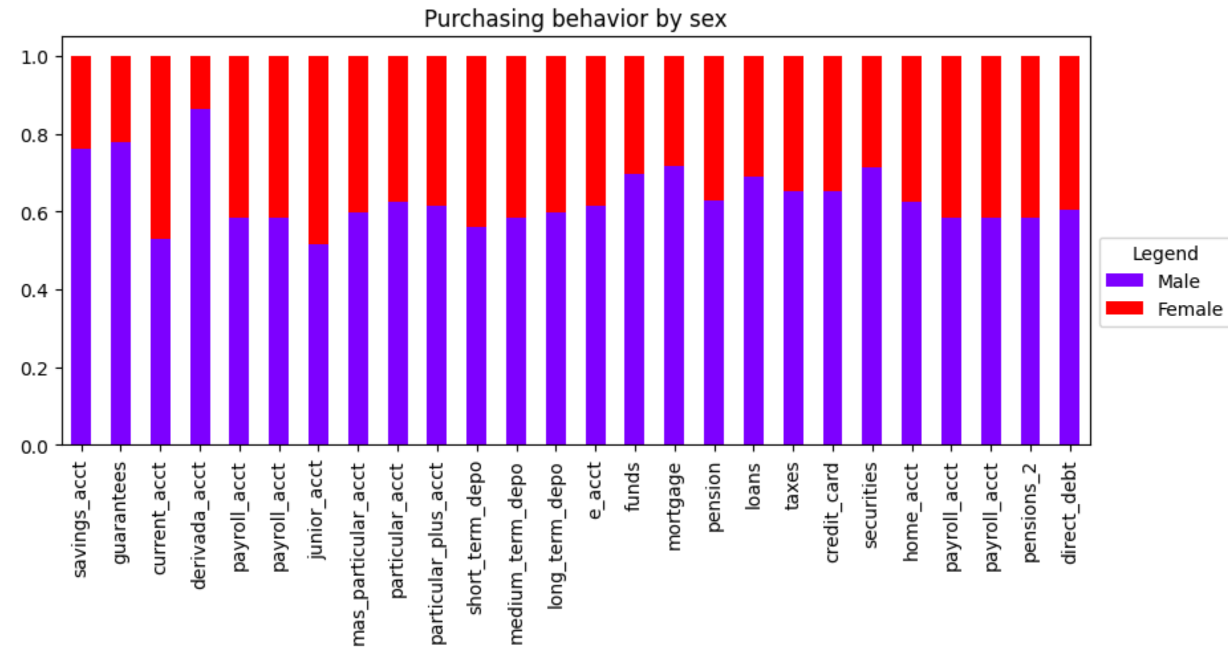
In [27]:

```
sex_plot = train['sex_H'].value_counts().plot(kind='bar')
plt.title('Sex Distribution')
plt.xticks(ticks=[0,1], labels=['Male', 'Female'], rotation=45)
plt.bar_label(sex_plot.containers[0], fmt=int)
plt.show()
```

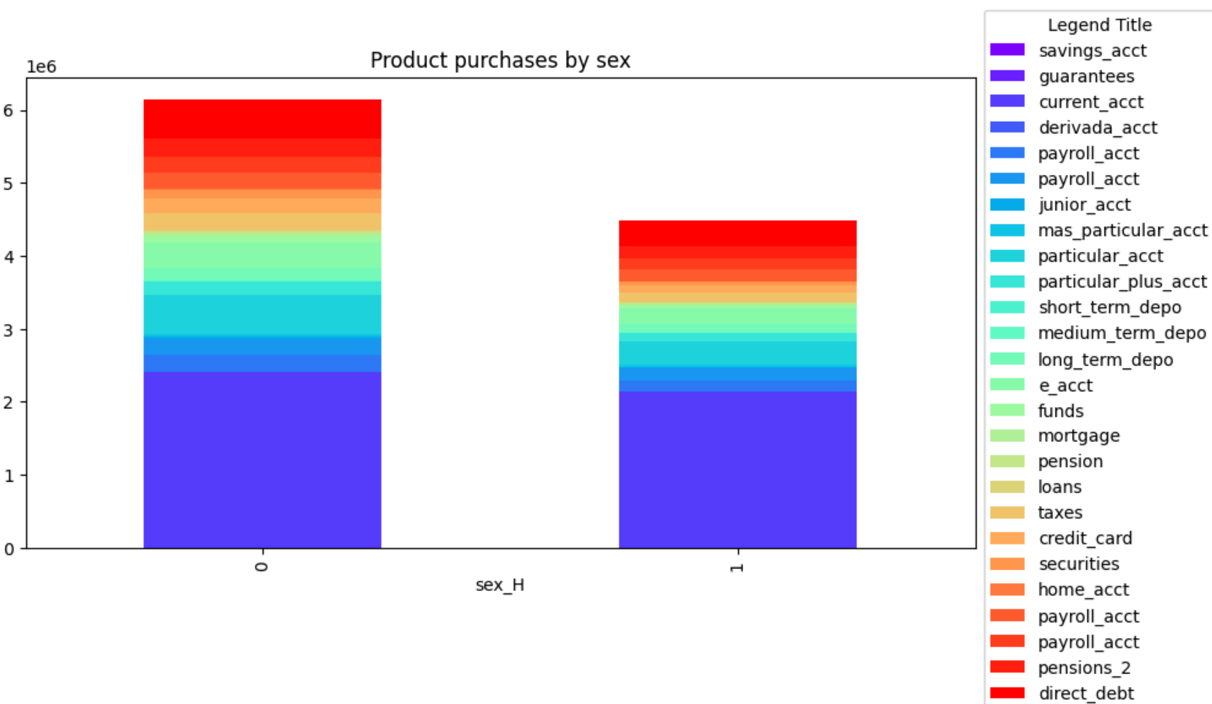



In [28]:
```
# Defining function to plot column data and see product purchasing behavior
def plot_grouped_data(df, group_col, products, colormap='rainbow', title='Grouped Data
    dummy = df.groupby(group_col)[products].sum()
    dummy = (dummy/dummy.sum()).T
    ax = dummy.plot(kind='bar', stacked=True, colormap=colormap, figsize=(10,4))
    ax.set_title(title)

    if show_legend:
        plt.legend(loc='center left', title='Legend', bbox_to_anchor=(1, .4))
    else:
        plt.legend().set_visible(False)
```

In [29]:
```
plot_grouped_data(train, 'sex_H', products, title='Purchasing behavior by sex', show_l
plt.legend(labels=('Male', 'Female'),loc='center left', title='Legend', bbox_to_anchor
plt.show()
```




In [30]:
```
dummy = train[['sex_H']+products].groupby('sex_H').sum()
dummy.plot(kind='bar',stacked=True, colormap='rainbow',figsize=(10,5))
plt.legend(loc='center left', title='Legend Title', bbox_to_anchor=(1, .4))
plt.title('Product purchases by sex')
plt.show()
```




In [31]:
```
dummy
```

Out[31]:

| sex_H | savings_acct | guarantees | current_acct | derivada_acct | payroll_acct | payroll_acct | junior_acct | ma |
|---|---|---|---|---|---|---|---|---|
| 0 | 497 | 102 | 2414549 | 2336 | 220587 | 220587 | 32881 | |
| 1 | 156 | 29 | 2136189 | 370 | 157492 | 157492 | 30972 | |

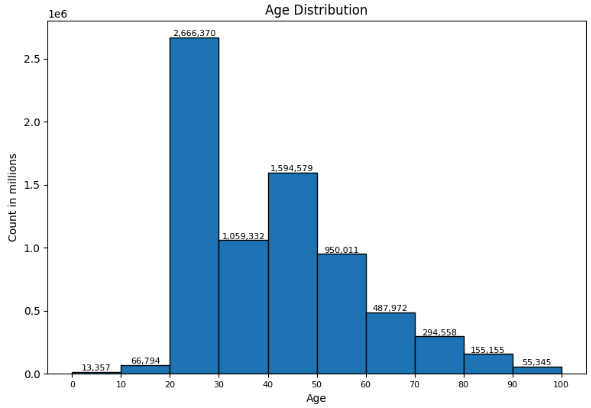
- We can see that the bank has more male customers and they have bought more products than female customers
- Male customers have bought most of the products, showing a skewed distribution in the total products bought
- All the products, except derivada account show a relative similar ratio to the amount of male anad female customers, showing that it there may not be a product preference when it comes to gender, and the difference comes from the number of customers

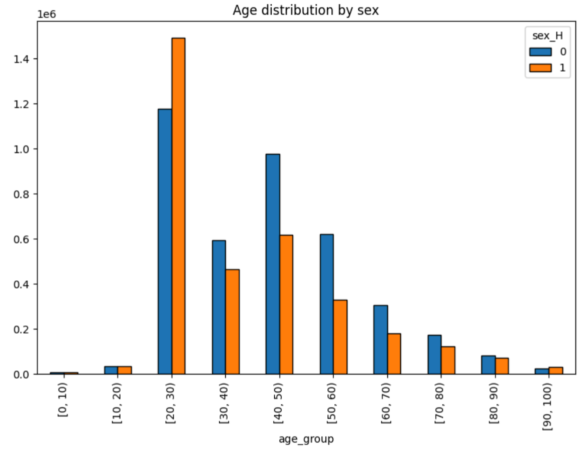
## Analyzing column: Age

In [32]:
```
age_sex = train.groupby('sex_H')['age'].size()
```

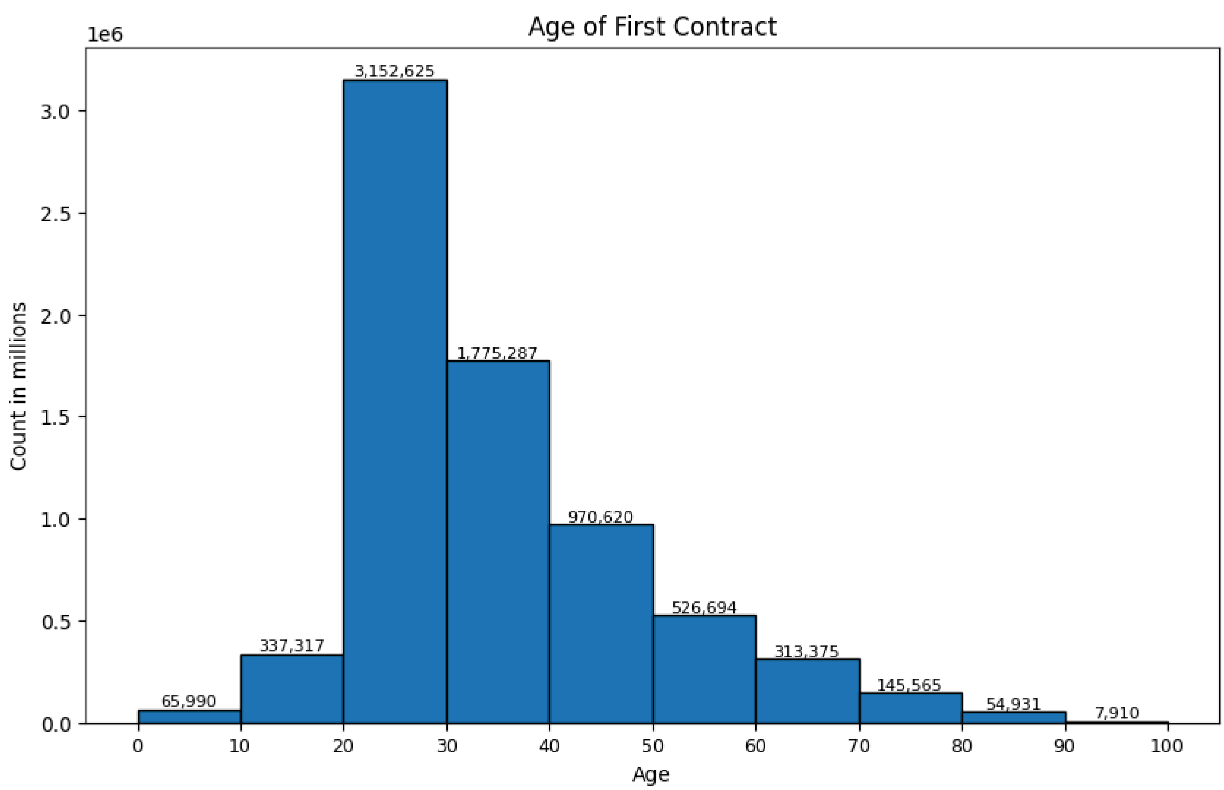
In [33]:
```
bin_edges = np.arange(0, 101, 10)
labels = [f'{i}-{i+19}' for i in bin_edges[:-1]]

# Create age groups
train['age_group'] = pd.cut(train['age'], bins=bin_edges, right=False)
age_sex_counts = train.groupby(['age_group', 'sex_H']).size().unstack(fill_value=0)

plt.figure(figsize=(20,6))
plt.subplot(1,2,1)

#Age Distribution
bin_edges = np.arange(0, 101, 10)
values, bins, bars = plt.hist(train['age'], bins=bin_edges, edgecolor = 'black')
plt.title('Age Distribution')
plt.ylabel('Count in millions')
plt.xlabel('Age')
plt.xticks(bin_edges, fontsize=8)
plt.bar_label(bars, fmt='{:,.0f}', fontsize=8)

#Age distribution by sex
plt.subplot(1,2,2)
age_sex_counts.plot(kind='bar', stacked=False, edgecolor='black', ax=plt.gca())
plt.title('Age distribution by sex')

plt.show()
```

```
C:\Users\MARIA\AppData\Local\Temp\ipykernel_10476\2932124040.py:6: FutureWarning: The
default of observed=False is deprecated and will be changed to True in a future versi
on of pandas. Pass observed=False to retain current behavior or observed=True to adop
t the future default and silence this warning.
  age_sex_counts = train.groupby(['age_group', 'sex_H']).size().unstack(fill_value=0)
```




In [34]:
```
train['first_contract_age'] = train['age']-(train['seniority_in_months']/12).round()
test_set['first_contract_age'] = test_set['age']-(test_set['seniority_in_months']/12).
val_set['first_contract_age'] = val_set['age']-(val_set['seniority_in_months']/12).rou

bin_edges = np.arange(0, 101, 10)

plt.figure(figsize=(10,6))
values, bins, bars = plt.hist(train['first_contract_age'], bins=bin_edges, edgecolor =
plt.title('Age of First Contract')
plt.ylabel('Count in millions')
plt.xlabel('Age')
plt.xticks(bin_edges, fontsize=9)
plt.bar_label(bars, fmt='{:,.0f}', fontsize=8)
plt.show()
```

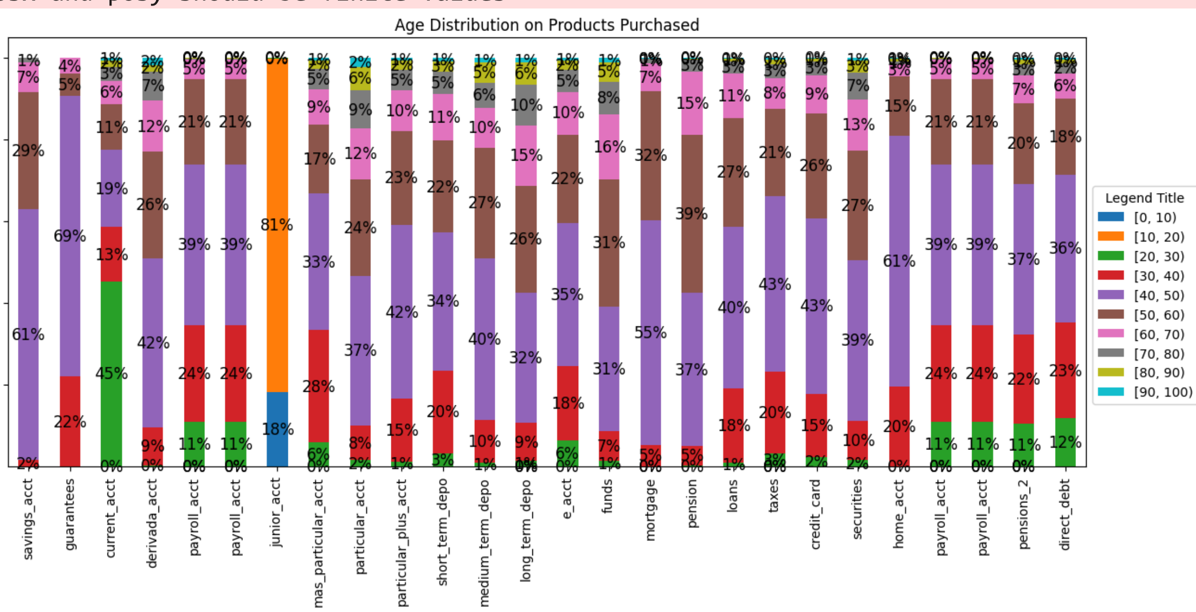



In [35]:
```
dummy = train.groupby('age_group')[products].sum()
dummy = (dummy/dummy.sum()).T
ax = dummy.plot(kind='bar', stacked=True, figsize=(15,6))
plt.legend(loc='center left', title='Legend Title', bbox_to_anchor=(1, .4))
```

```
plt.title('Age Distribution on Products Purchased')

for container in ax.containers:
    ax.bar_label(container, labels=[f'{v*100:.0f}%' for v in container.datavalues], la

plt.show()
```

```
C:\Users\MARIA\AppData\Local\Temp\ipykernel_10476\2298204075.py:1: FutureWarning: The
default of observed=False is deprecated and will be changed to True in a future versi
on of pandas. Pass observed=False to retain current behavior or observed=True to adop
t the future default and silence this warning.
  dummy = train.groupby('age_group')[products].sum()
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
```

```
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
```




In [36]:
```
dummy*100
```

Out[36]:

| age_group | [0, 10) | [10, 20) | [20, 30) | [30, 40) | [40, 50) | [50, 60) | [60, 70) | [70, |
|---|---|---|---|---|---|---|---|---|
| **savings_acct** | 0.000000 | 0.000000 | 0.000000 | 1.684533 | 61.255743 | 28.637060 | 7.350689 | 1.07 |
| **guarantees** | 0.000000 | 0.000000 | 0.000000 | 22.137405 | 68.702290 | 5.343511 | 3.816794 | 0.00 |
| **current_acct** | 0.000308 | 0.146629 | 45.165336 | 13.313117 | 18.856746 | 11.248619 | 5.710774 | 3.35 |
| **derivada_acct** | 0.000000 | 0.000000 | 0.406504 | 9.164819 | 41.537324 | 25.979305 | 12.490761 | 7.13 |
| **payroll_acct** | 0.000794 | 0.013225 | 11.077608 | 23.507614 | 39.421375 | 20.851220 | 4.780080 | 0.25 |
| **payroll_acct** | 0.000794 | 0.013225 | 11.077608 | 23.507614 | 39.421375 | 20.851220 | 4.780080 | 0.25 |
| **junior_acct** | 18.365621 | 81.419824 | 0.203593 | 0.010963 | 0.000000 | 0.000000 | 0.000000 | 0.00 |
| **mas_particular_acct** | 0.000000 | 0.001529 | 5.898004 | 27.604557 | 33.360349 | 16.778041 | 8.792721 | 4.83 |
| **particular_acct** | 0.000000 | 0.000116 | 1.631293 | 8.432069 | 36.683015 | 23.606139 | 12.407763 | 9.35 |
| **particular_plus_acct** | 0.000000 | 0.000000 | 1.314570 | 15.400392 | 42.404262 | 23.107702 | 9.815551 | 5.02 |
| **short_term_depo** | 0.000000 | 0.000000 | 3.218104 | 20.357106 | 33.738192 | 22.474826 | 11.439842 | 5.41 |
| **medium_term_depo** | 0.000000 | 0.000000 | 0.943306 | 10.433540 | 39.704621 | 26.965222 | 9.861839 | 6.03 |
| **long_term_depo** | 0.002441 | 0.029288 | 1.452898 | 9.213196 | 31.797103 | 26.340453 | 14.642302 | 9.95 |
| **e_acct** | 0.000000 | 0.001221 | 6.468729 | 18.178315 | 35.056398 | 21.631768 | 10.439694 | 5.28 |
| **funds** | 0.000000 | 0.000000 | 1.396801 | 7.243026 | 30.585020 | 31.132458 | 15.803399 | 8.11 |
| **mortgage** | 0.022965 | 0.000000 | 0.012758 | 5.368717 | 54.963001 | 31.592243 | 6.825721 | 1.13 |
| **pension** | 0.000000 | 0.000000 | 0.311899 | 4.785145 | 37.452124 | 38.743354 | 15.499604 | 2.82 |
| **loans** | 0.000000 | 0.000000 | 1.093898 | 17.993141 | 39.652318 | 26.637890 | 10.873936 | 2.98 |
| **taxes** | 0.002471 | 0.027452 | 3.292100 | 19.934608 | 42.919460 | 21.307506 | 7.710849 | 3.21 |
| **credit_card** | 0.000000 | 0.000000 | 2.412838 | 15.439503 | 42.880297 | 25.819034 | 9.300244 | 3.42 |
| **securities** | 0.000000 | 0.000000 | 1.630843 | 9.710487 | 39.101229 | 26.887401 | 12.601605 | 6.51 |
| **home_acct** | 0.000000 | 0.000000 | 0.058500 | 19.613120 | 61.311961 | 14.617215 | 2.847003 | 0.82 |
| **payroll_acct** | 0.000794 | 0.013225 | 11.077608 | 23.507614 | 39.421375 | 20.851220 | 4.780080 | 0.25 |
| **payroll_acct** | 0.000794 | 0.013225 | 11.077608 | 23.507614 | 39.421375 | 20.851220 | 4.780080 | 0.25 |
| **pensions_2** | 0.011391 | 0.047017 | 10.540258 | 21.745779 | 36.794565 | 19.852018 | 6.742556 | 2.86 |
| **direct_debt** | 0.000000 | 0.000452 | 11.839415 | 23.416433 | 36.291652 | 18.478117 | 6.313586 | 2.46 |

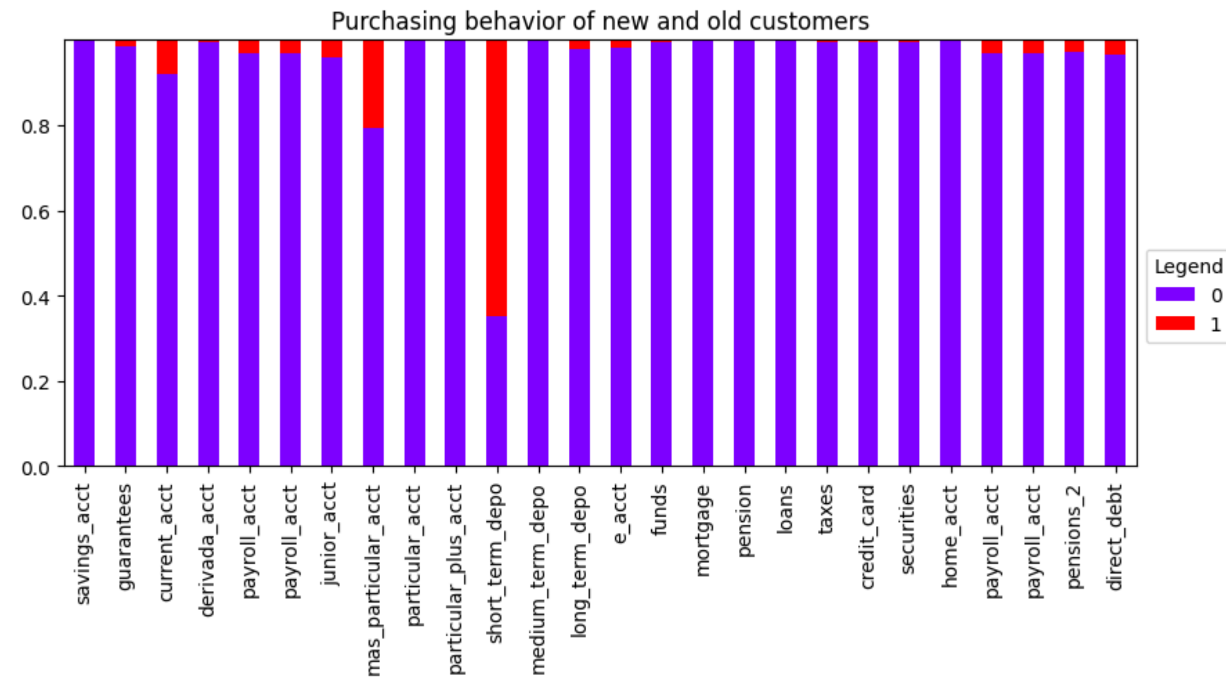
- There are more male than female customers in all age groups, except 90-100
- There is a large number of young customers, in the 20-30 range and the second largest group of clients are on the 40-50 age range
- Most clients buy their first product when they are in the age group of 20-30
- There is a clear difference in the age group signed up for the junior account -predominantly with clients in the 10-20 age range

- There is a good amount of variety base on age group and products bough. Product dominance usually interchanges between age groups 40-50 and 50-60, followed by 30-40

## Analyzing column: New Customers

In [37]:
```
print(train['new_cust'].value_counts())

plot_grouped_data(train, 'new_cust', products, title='Purchasing behavior of new and c
plt.show()
```

```
new_cust
0    6802900
1     547804
Name: count, dtype: int64
```




- New customers purchase mostly short term deposits and mas particular account
- Very few participation on other products, meaning that few products offered by the bank are actually attracting new customers

## Analyzing column: Seniority in Months

In [38]:
```
print(train['seniority_in_months'].value_counts().sort_index())

plot_grouped_data(train, 'seniority_in_months', products, title='Purchasing behavior b
plt.show()
```

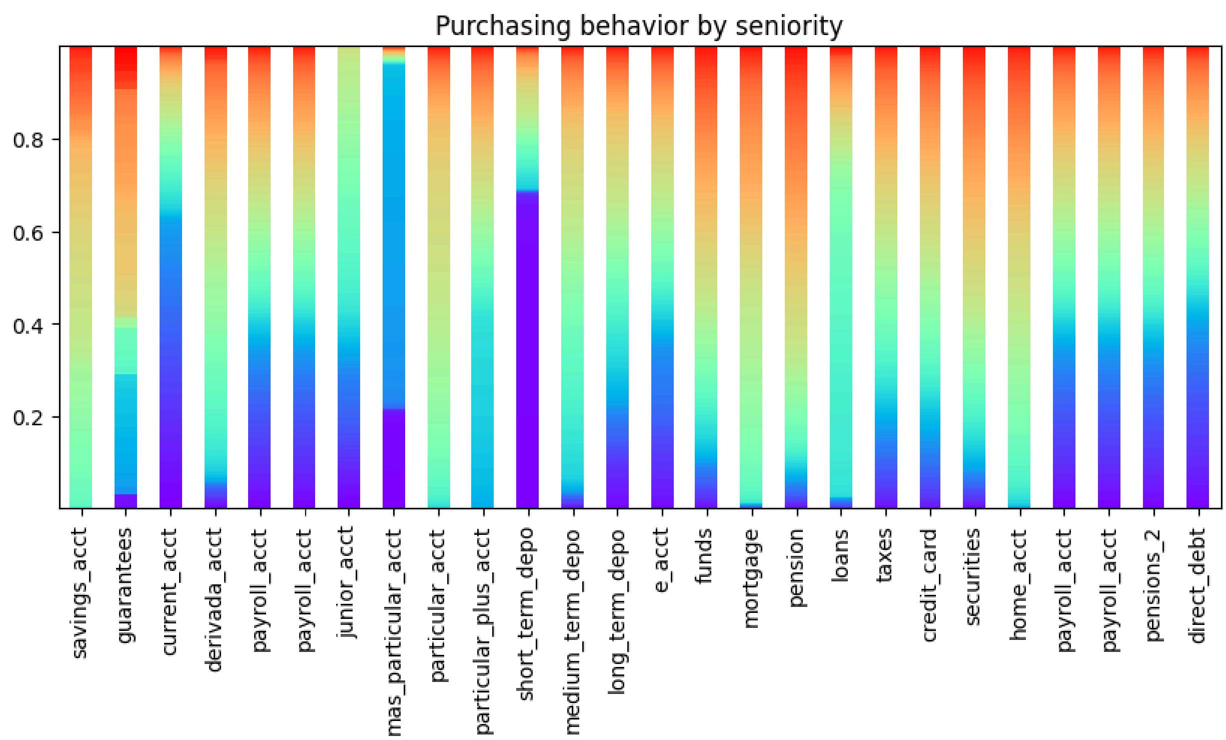
```
seniority_in_months
-999999       14
 0         94063
 1         92587
 2         89380
 3         89877
 4         83933
 5         88095
 6         79742
 7         78786
 8         75629
 9         73743
 10        84822
 11        67135
 12       104715
 13        77327
 14        80727
 15        77936
 16        79900
 17        78522
 18        75535
 19        69870
 20        69070
 21        81152
 22        72686
 23        77104
 24        80156
 25        70212
 26        73359
 27        69215
 28        68679
 29        66298
 30        64299
 31        61612
 32        58795
 33        66440
 34        61948
 35        63292
 36        72391
 37        64670
 38        66988
 39        63180
 40        66430
 41        64465
 42        59939
 43        65549
 44        63357
 45        66941
 46        63069
 47        58018
 48        61523
 49        56901
 50        57398
 51        52852
 52        56166
 53        54687
 54        47683
 55        40071
 56        29713
 57        21359
```

```
58          16298
59          12974
60          13024
61          14370
62          15634
63          16812
64          17474
65          16416
66          16553
67          16312
68          15679
69          15458
70          14361
71          13742
72          13212
73          11240
74           9303
75           8189
76           8669
77           9933
78          11265
79          11970
80          13870
81          18977
82          18582
83          19809
84          21517
85          21625
86          23286
87          23674
88          24354
89          24172
90          24084
91          23053
92          21143
93          22655
94          21445
95          22179
96          23471
97          22999
98          23787
99          21991
100         21904
101         22420
102         24174
103         23195
104         23606
105         24828
106         23070
107         24078
108         24044
109         23321
110         25800
111         24487
112         25124
113         23534
114         24155
115         23606
116         22390
117         25113
```

| | |
|---|---|
| 118 | 23720 |
| 119 | 24069 |
| 120 | 23909 |
| 121 | 21769 |
| 122 | 22079 |
| 123 | 21685 |
| 124 | 22688 |
| 125 | 22582 |
| 126 | 22472 |
| 127 | 22844 |
| 128 | 21057 |
| 129 | 20605 |
| 130 | 19988 |
| 131 | 18953 |
| 132 | 20129 |
| 133 | 20128 |
| 134 | 21927 |
| 135 | 19989 |
| 136 | 21220 |
| 137 | 20596 |
| 138 | 21017 |
| 139 | 21041 |
| 140 | 21007 |
| 141 | 21187 |
| 142 | 21096 |
| 143 | 20935 |
| 144 | 21174 |
| 145 | 19819 |
| 146 | 20552 |
| 147 | 19052 |
| 148 | 18944 |
| 149 | 17860 |
| 150 | 18580 |
| 151 | 19034 |
| 152 | 18312 |
| 153 | 19173 |
| 154 | 18096 |
| 155 | 17288 |
| 156 | 20706 |
| 157 | 20008 |
| 158 | 20639 |
| 159 | 23150 |
| 160 | 25172 |
| 161 | 26311 |
| 162 | 30184 |
| 163 | 30692 |
| 164 | 32592 |
| 165 | 36741 |
| 166 | 35882 |
| 167 | 32544 |
| 168 | 35368 |
| 169 | 35445 |
| 170 | 35167 |
| 171 | 33208 |
| 172 | 34128 |
| 173 | 30623 |
| 174 | 29756 |
| 175 | 27290 |
| 176 | 24663 |
| 177 | 24977 |

```
178        24083
179        22842
180        23487
181        19984
182        20182
183        17718
184        18458
185        17902
186        17064
187        16536
188        15895
189        15609
190        14152
191        13444
192        13609
193        13955
194        14173
195        13073
196        12389
197        11786
198        12205
199        12872
200        12390
201        13116
202        12436
203        11810
204        11515
205        11408
206        12025
207        11217
208        12166
209        11685
210        11076
211        11517
212        11182
213        10998
214        10945
215        10605
216        10471
217        10199
218         9634
219         8549
220         8699
221         8287
222         7526
223         7694
224         7180
225         7630
226         6903
227         6502
228         6709
229         6359
230         6273
231         6467
232         6017
233         5370
234         5515
235         5588
236         5009
237         6028
```

```
238         5767
239         5140
240         5250
241         5131
242         4548
243         4186
244         3855
245         3247
246         2956
247         2454
248         1645
249         1237
250         1048
251          718
252          477
253          304
254          186
255          134
256           66
Name: count, dtype: int64
```




Cool colors indicate more recent clients and warm colors indicate higher seniority.
For example: 0 months will be purple, converging to blue green, orange and finally red will be 256 months

- Current account, juniour account, mas account and short term account have the newest clients of the bank
- Oldest customers have bought savings, particular_plus account, mortgage, loans and home accounts

## Analyzing column: Customer Type

Correcting data type

In [39]:
```
train['cust_type'] = train['cust_type'].astype(str).str.strip()
test_set['cust_type'] = test_set['cust_type'].astype(str).str.strip()
val_set['cust_type'] = val_set['cust_type'].astype(str).str.strip()

train['cust_type'].unique()
```

Out[39]:
```
array(['1', '1.0', '0', '3.0', '2', '3', 'P', '2.0', '4.0', '4'],
      dtype=object)
```

In [40]:
```
cust_type_map = {'0.0': '0', '1.0': '1', '2.0': '2', '3.0': '3', '4.0': '4'}
train['cust_type'] = train['cust_type'].replace(cust_type_map)

test_set['cust_type'] = test_set['cust_type'].replace(cust_type_map)
val_set['cust_type'] = val_set['cust_type'].replace(cust_type_map)

train['cust_type'] = train['cust_type'].astype(object)

test_set['cust_type'] = test_set['cust_type'].astype(object)
val_set['cust_type'] = val_set['cust_type'].astype(object)

print(train['cust_type'].value_counts())
print(train['cust_relationship'].value_counts())
```

```
cust_type
1    7260501
0      85390
3       3040
2        917
P        627
4        229
Name: count, dtype: int64
cust_relationship
I    4084146
A    3177272
0      85390
P       3269
R        623
N          4
Name: count, dtype: int64
```

Same number of NA (0) for both relationship and customer type

In [41]:
```
na_rel = train[train['cust_type'] == '0']
na_rel[products].sum().sort_values(ascending=False)
```

Out[41]:
```
current_acct           48358
mas_particular_acct      962
direct_debt              426
short_term_depo          377
junior_acct              307
pensions_2               302
payroll_acct             297
payroll_acct             297
payroll_acct             297
payroll_acct             297
long_term_depo           169
e_acct                    49
securities                15
taxes                      6
funds                      5
credit_card                2
pension                    1
derivada_acct              1
medium_term_depo           0
guarantees                 0
mortgage                   0
loans                      0
home_acct                  0
particular_plus_acct       0
particular_acct            0
savings_acct               0
dtype: int64
```

In [42]:
```
train['total_products'].groupby(train['cust_type']).sum()
```

Out[42]:
```
cust_type
0       52168
1    10578826
2         434
3        1781
4         163
P         411
Name: total_products, dtype: int64
```

In [43]:
```
dummy = train.groupby('cust_type')[products].sum()
dummy = (dummy/dummy.sum()).T

print((dummy*100).round(2))
```

```
cust_type                0       1     2     3     4     P
savings_acct          0.00  100.00  0.00  0.00  0.00  0.00
guarantees            0.00  100.00  0.00  0.00  0.00  0.00
current_acct          1.06   98.89  0.00  0.03  0.00  0.01
derivada_acct         0.04   99.96  0.00  0.00  0.00  0.00
payroll_acct          0.08   99.92  0.00  0.00  0.00  0.00
payroll_acct          0.08   99.92  0.00  0.00  0.00  0.00
junior_acct           0.48   99.51  0.00  0.00  0.00  0.00
mas_particular_acct   1.47   97.96  0.04  0.42  0.03  0.07
particular_acct       0.00  100.00  0.00  0.00  0.00  0.00
particular_plus_acct  0.00  100.00  0.00  0.00  0.00  0.00
short_term_depo       3.91   95.11  0.09  0.67  0.09  0.11
medium_term_depo      0.00  100.00  0.00  0.00  0.00  0.00
long_term_depo        0.06   99.90  0.03  0.01  0.00  0.00
e_acct                0.01   99.99  0.00  0.00  0.00  0.00
funds                 0.00   99.99  0.00  0.00  0.00  0.00
mortgage              0.00  100.00  0.00  0.00  0.00  0.00
pension               0.00   99.97  0.02  0.00  0.00  0.00
loans                 0.00   99.99  0.01  0.00  0.00  0.00
taxes                 0.00   99.99  0.00  0.01  0.00  0.00
credit_card           0.00   99.99  0.01  0.00  0.00  0.00
securities            0.01   99.99  0.00  0.00  0.00  0.00
home_acct             0.00  100.00  0.00  0.00  0.00  0.00
payroll_acct          0.08   99.92  0.00  0.00  0.00  0.00
payroll_acct          0.08   99.92  0.00  0.00  0.00  0.00
pensions_2            0.07   99.92  0.00  0.00  0.00  0.00
direct_debt           0.05   99.95  0.00  0.00  0.00  0.00
```

- Customers with type and relationship missing have bought mostly current account, mas particular account and short term deposit. These are likely new customers
- Customers with type 2, 3 and P have bought mostly mas particular account and short term deposit accounts

## Analyzing column: Join Channel

In [44]:
```
train['join_channel'].value_counts()
```

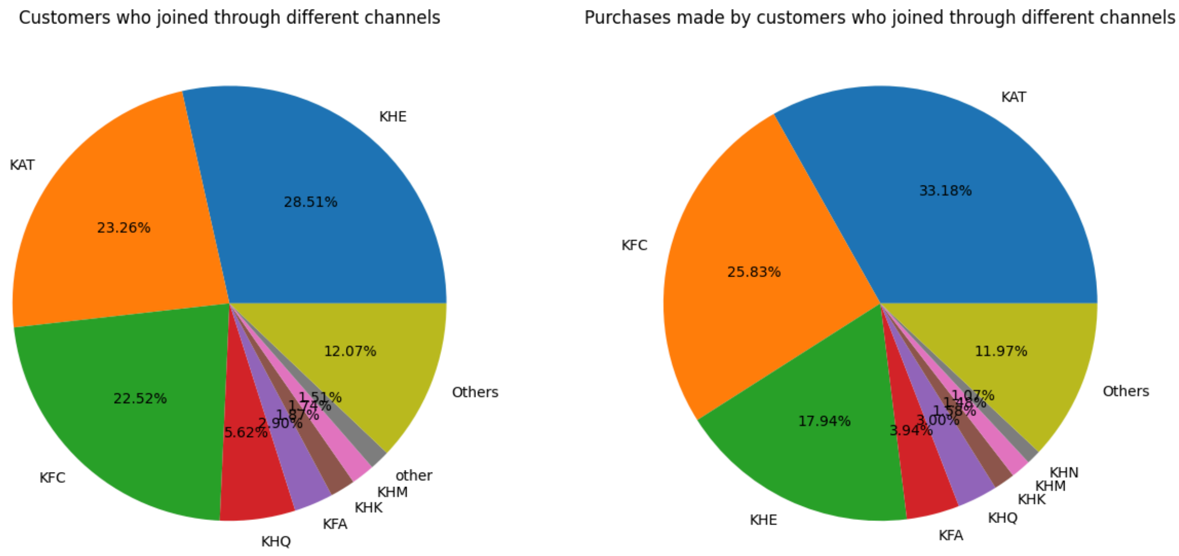
Out[44]:
```
join_channel
KHE      2095903
KAT      1709503
KFC      1655057
KHQ       413265
KFA       213305
KHK       137531
KHM       128004
other     110709
KHN        79888
KHD        61353
KAS        45889
RED        41080
KAG        38692
KAY        36402
KAA        35588
KAB        33183
KAE        27195
KCC        25857
KHL        24952
KBZ        24655
KFD        24092
KAI        20258
KEY        18691
KAW        18073
KAR        17711
KAZ        17615
007        16861
KAF        15964
KCI        14619
013        13763
KAJ        13605
KCH        13494
KAH        13086
KHF        11475
KAQ         9796
KHC         9096
KAP         8306
KHO         6171
KAM         6106
KAD         5467
KEJ         5200
KGX         5179
KFP         5164
KGV         4788
KFT         4528
KDR         4437
KAL         4202
KBO         4072
KAC         4038
KBH         3920
KFS         3884
KFJ         3748
KAO         3715
KFG         3692
KES         3096
KFF         3026
KEW         2951
KCG         2815
KFU         2755
```

```
KCB     2743
KEN     2635
KFN     2483
KGY     2249
KCL     2242
KBQ     2222
KFK     2173
KBF     2134
KFL     2085
KCD     1834
KCM     1773
KBU     1755
KED     1628
KFH     1512
KDU     1461
KDM     1379
KEZ     1269
KEL     1248
KDY     1169
KDS     1165
KDO     1060
KEG     1024
KBR     1008
KDX      989
KDC      877
KBG      865
KCA      832
KEH      800
KBB      726
KAN      704
KDT      693
KBW      673
KCN      635
KCU      629
KDQ      594
KDP      587
KGW      581
KBV      510
KCK      509
KEI      505
KFI      499
KDE      490
KEA      487
KHP      484
KEO      470
KEV      445
KAK      420
KDW      419
KBS      414
KBY      394
KDF      383
KDD      330
KBL      330
KDZ      305
KEK      291
KBJ      280
KBM      277
KDG      276
KCF      244
KDA      235
```

```
KDV          222
KFM          217
KFR          210
KEB          201
KEF          191
KCE          164
KFE          142
KCV          139
KEU          137
KCS          134
KBD          133
KAU          128
KCJ          118
KEC          113
004          110
KDN          108
KDH          102
KCQ          101
KEE           98
KCR           94
KCO           94
KEQ           83
KCP           81
K00           80
KBE           70
KCT           67
KFB           60
KAV           53
KBX           50
KCX           45
KBP           45
KBN           44
KFV           38
KEM           33
KHA           23
KGC           15
KGU           15
KDI           11
025            7
KGN            7
KDB            7
KDL            6
KHS            4
KHR            1
Name: count, dtype: int64
```

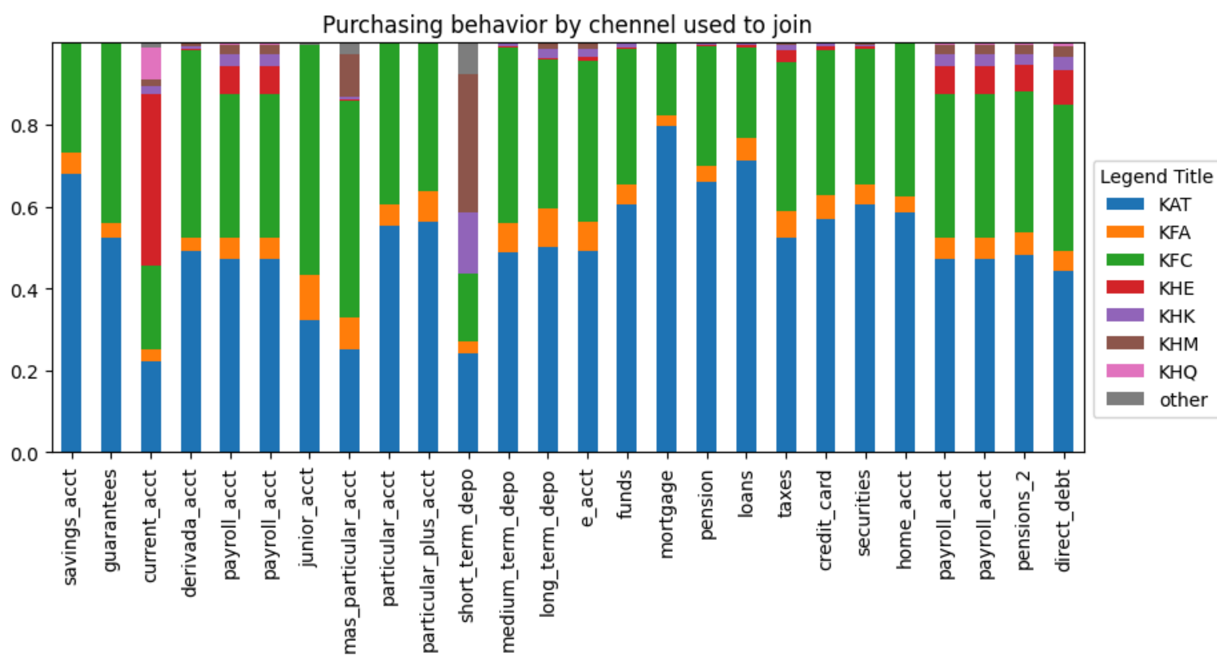
In [45]:

```
channel = train['join_channel'].value_counts()[:8]
others = train['join_channel'].value_counts()[8:]
dummy = train.groupby('join_channel')[products].sum().sum(axis=1)
dummy = dummy.sort_values(ascending=False)

plt.figure(figsize=(15,8))
plt.subplot(1,2,1)
plt.pie(list(channel)+[others.sum()], labels=list(channel.index)+['Others'], autopct='
plt.title('Customers who joined through different channels')
plt.subplot(1,2,2)
plt.pie(list(dummy.values[:8])+[dummy.values[8:].sum()], labels = list(dummy.index[:8]
plt.title('Purchases made by customers who joined through different channels')
plt.show()
```




In [46]:

```python
dummy = train.groupby('join_channel')[products].sum()
dummy = dummy[dummy.index.isin(channel.keys())]
dummy = (dummy/dummy.sum()).T
dummy.plot(kind='bar',stacked=True, figsize=(10,4))
plt.legend(loc='center left', title='Legend Title', bbox_to_anchor=(1, .4))
plt.title('Purchasing behavior by chennel used to join')
plt.show()
```



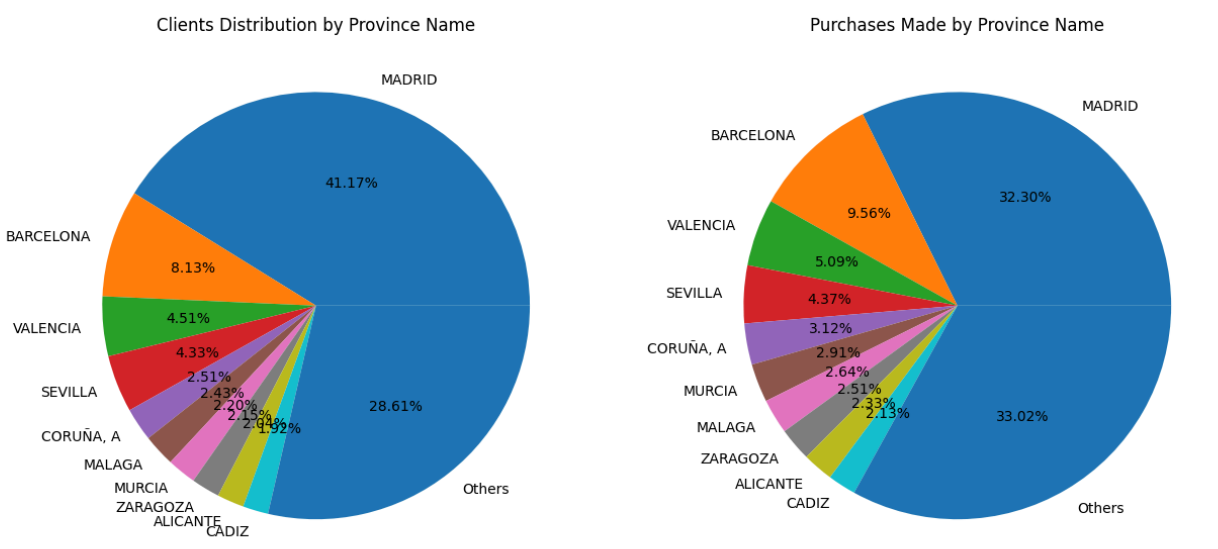

- The 8 top channels used to join the bank make up 88% of the total customers
- When we compare the number of purchases made by clients with to the channel they joined, 7 out of 8 channels are repeated on the number of clients who joined and number of purchases made, with the exception of "others" which were missing values
- Customers who joined through KAT have most purchases
- Customers who joined through KFC have more purchased than KAT on mas particular account and junior account

# Analyzing column: Province Name

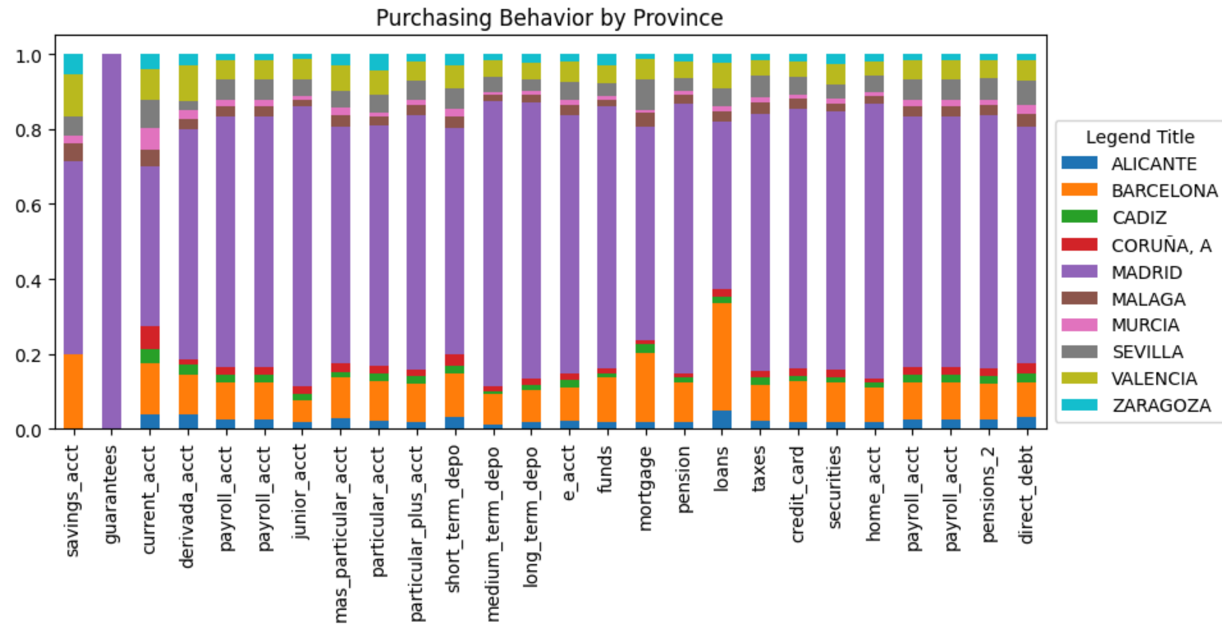
In [47]:
```
train['province_name'].value_counts().head(10)
```

Out[47]:
```
province_name
MADRID        2374509
BARCELONA      703041
VALENCIA       374474
SEVILLA        321534
CORUÑA, A      229212
MURCIA         213652
MALAGA         194405
ZARAGOZA       184413
ALICANTE       171465
CADIZ          156627
Name: count, dtype: int64
```

In [48]:
```
top_province = train['province_name'].value_counts()[:10]
province_others = train['province_name'].value_counts()[10:]
dummy = train.groupby('province_name')[products].sum().sum(axis=1)
dummy = dummy.sort_values(ascending=False)

plt.figure(figsize=(15,8))
plt.subplot(1,2,1)
plt.pie(list(dummy.values[:10])+[dummy.values[10:].sum()], labels = list(dummy.index[:
plt.title('Clients Distribution by Province Name')
plt.subplot(1,2,2)
plt.pie(list(top_province)+[province_others.sum()], labels=list(top_province.index)+['
plt.title('Purchases Made by Province Name')
plt.show()
```




In [49]:
```
dummy = train.groupby('province_name')[products].sum()
dummy = dummy[dummy.index.isin(top_province.keys())]
dummy = (dummy/dummy.sum()).T
dummy.plot(kind='bar',stacked=True, figsize=(10,4))
plt.legend(loc='center left', title='Legend Title', bbox_to_anchor=(1, .4))
plt.title('Purchasing Behavior by Province')
plt.show()
```

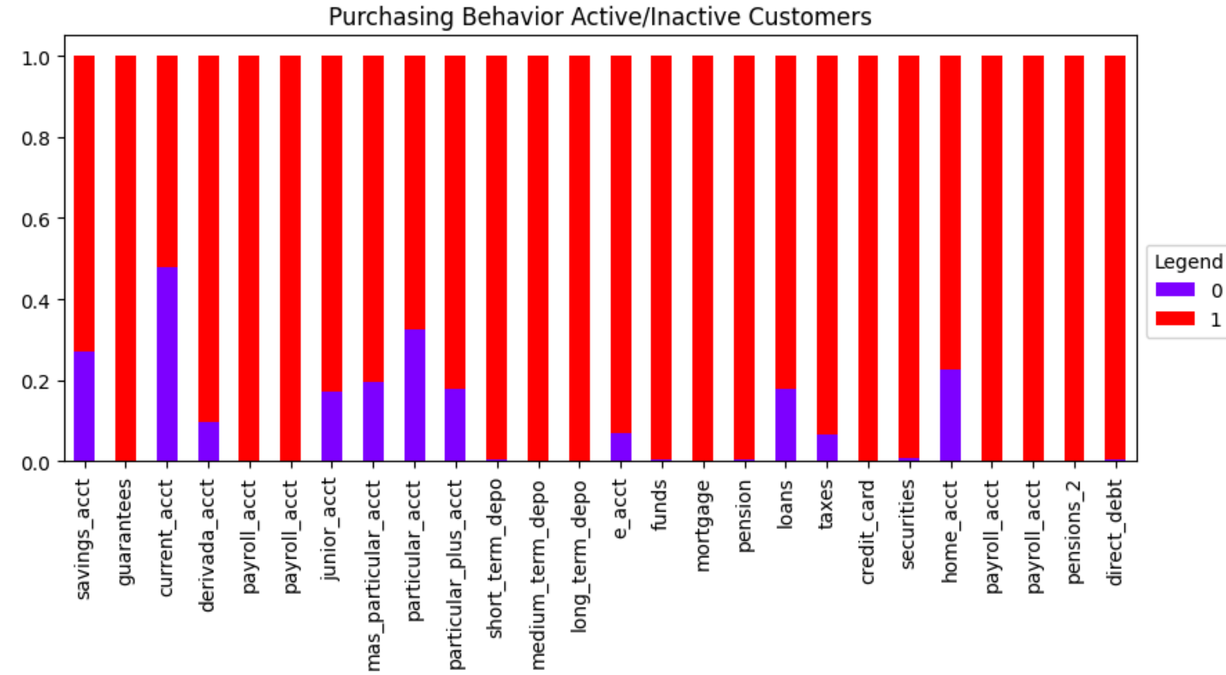



- We can see most of the clients are from Madrid, Spain's capital, followed by other cities like Barcelona, Valencia and Sevilla
- The distribution of number products bought is quite similar to the number of clients from each city
- Guarantees account have only Madrid clients
- Clients from Barcelona buys mostly loans and savings account

## Analyzing column: Active Customer

In [50]:
```
print(train['active_cust'].value_counts())

plot_grouped_data(train, 'active_cust', products, title='Purchasing Behavior Active/Ir
plt.show()
```

```
active_cust
0    4171818
1    3178886
Name: count, dtype: int64
```




- There is a lot of inactive customers on the dataset, however all products have more active than inactive customers
- The three products with most inactive customers are savings, current and particular account

## Analyzing column: Income

In [51]:
```
(train['income'] == 0).sum()
```

Out[51]:
```
1566917
```

In [52]:
```
train.groupby('province_name')['income'].describe().round()
```

Out[52]:

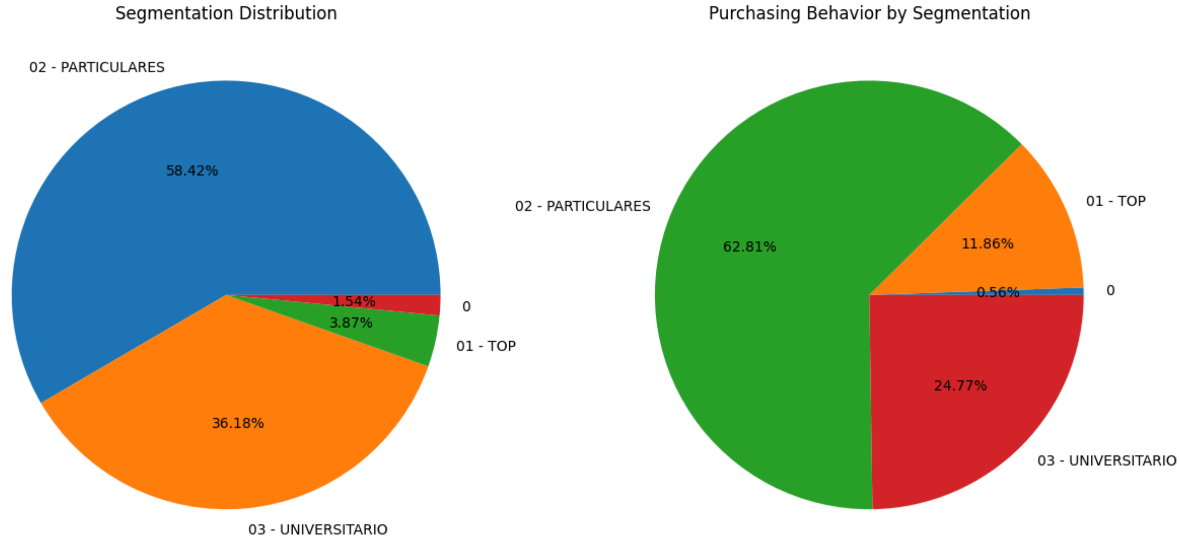
| province_name | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **ALAVA** | 20457.0 | 131.0 | 4213.0 | 0.0 | 0.0 | 0.0 | 0.0 | 253563.0 |
| **ALBACETE** | 61597.0 | 70692.0 | 45598.0 | 0.0 | 43643.0 | 72158.0 | 97240.0 | 764582.0 |
| **ALICANTE** | 171465.0 | 64526.0 | 142980.0 | 0.0 | 0.0 | 51896.0 | 86986.0 | 17804048.0 |
| **ALMERIA** | 32964.0 | 65368.0 | 57020.0 | 0.0 | 27816.0 | 61371.0 | 89659.0 | 578349.0 |
| **ASTURIAS** | 142752.0 | 75717.0 | 86326.0 | 0.0 | 0.0 | 73111.0 | 103932.0 | 4950059.0 |
| **AVILA** | 20854.0 | 60446.0 | 71181.0 | 0.0 | 33132.0 | 59390.0 | 82089.0 | 2768593.0 |
| **BADAJOZ** | 102555.0 | 56180.0 | 48798.0 | 0.0 | 23238.0 | 52291.0 | 79828.0 | 1103543.0 |
| **BALEARS, ILLES** | 69624.0 | 94224.0 | 374586.0 | 0.0 | 0.0 | 63926.0 | 130273.0 | 15711716.0 |
| **BARCELONA** | 703041.0 | 137804.0 | 151217.0 | 0.0 | 67883.0 | 115102.0 | 173602.0 | 5752268.0 |
| **BIZKAIA** | 100358.0 | 108.0 | 3775.0 | 0.0 | 0.0 | 0.0 | 0.0 | 314612.0 |
| **BURGOS** | 52601.0 | 80844.0 | 61975.0 | 0.0 | 47858.0 | 79624.0 | 113205.0 | 1785512.0 |
| **CACERES** | 68476.0 | 58339.0 | 51377.0 | 0.0 | 23851.0 | 56271.0 | 83456.0 | 1309035.0 |
| **CADIZ** | 156627.0 | 76406.0 | 87801.0 | 0.0 | 28262.0 | 65821.0 | 100663.0 | 3648374.0 |
| **CANTABRIA** | 84616.0 | 86161.0 | 98716.0 | 0.0 | 0.0 | 75791.0 | 116158.0 | 2276562.0 |
| **CASTELLON** | 56091.0 | 55684.0 | 54241.0 | 0.0 | 0.0 | 53309.0 | 79119.0 | 668527.0 |
| **CEUTA** | 3972.0 | 134575.0 | 274004.0 | 0.0 | 0.0 | 94871.0 | 146605.0 | 4082464.0 |
| **CIUDAD REAL** | 64483.0 | 58405.0 | 47191.0 | 0.0 | 33792.0 | 56022.0 | 78055.0 | 952513.0 |
| **CORDOBA** | 76747.0 | 67244.0 | 66935.0 | 0.0 | 28706.0 | 57136.0 | 93708.0 | 1496216.0 |
| **CORUÑA, A** | 229212.0 | 81233.0 | 83420.0 | 0.0 | 0.0 | 75812.0 | 117519.0 | 2564976.0 |
| **CUENCA** | 30268.0 | 53501.0 | 42794.0 | 0.0 | 18973.0 | 52647.0 | 82947.0 | 408454.0 |
| **GIPUZKOA** | 38757.0 | 215.0 | 7519.0 | 0.0 | 0.0 | 0.0 | 0.0 | 387346.0 |
| **GIRONA** | 49837.0 | 109189.0 | 177798.0 | 0.0 | 25991.0 | 87371.0 | 139741.0 | 6209401.0 |
| **GRANADA** | 96880.0 | 74024.0 | 94424.0 | 0.0 | 30785.0 | 70502.0 | 100516.0 | 4750243.0 |
| **GUADALAJARA** | 36140.0 | 75195.0 | 53215.0 | 0.0 | 43403.0 | 80531.0 | 107672.0 | 780996.0 |
| **HUELVA** | 64318.0 | 62350.0 | 55401.0 | 0.0 | 35781.0 | 60678.0 | 84070.0 | 1998667.0 |
| **HUESCA** | 21879.0 | 65937.0 | 85482.0 | 0.0 | 0.0 | 61538.0 | 87741.0 | 1137835.0 |
| **JAEN** | 34919.0 | 61626.0 | 50328.0 | 0.0 | 31581.0 | 58351.0 | 84452.0 | 553668.0 |
| **LEON** | 45011.0 | 74643.0 | 74316.0 | 0.0 | 38392.0 | 69489.0 | 101896.0 | 1985134.0 |
| **LERIDA** | 43435.0 | 62368.0 | 74555.0 | 0.0 | 25640.0 | 54210.0 | 81420.0 | 3587378.0 |
| **LUGO** | 45155.0 | 50604.0 | 54113.0 | 0.0 | 0.0 | 50083.0 | 72527.0 | 1126464.0 |
| **MADRID** | 2374509.0 | 156846.0 | 321263.0 | 0.0 | 72767.0 | 125063.0 | 192981.0 | 28894396.0 |
| **MALAGA** | 194405.0 | 93902.0 | 214666.0 | 0.0 | 37794.0 | 78490.0 | 120392.0 | 13268621.0 |
| **MELILLA** | 5095.0 | 115516.0 | 171526.0 | 0.0 | 56451.0 | 105855.0 | 142519.0 | 1959779.0 |

| province_name | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| MURCIA | 213652.0 | 56037.0 | 56010.0 | 0.0 | 0.0 | 54829.0 | 79338.0 | 3587378.0 |
| NAVARRA | 47904.0 | 134.0 | 4501.0 | 0.0 | 0.0 | 0.0 | 0.0 | 386063.0 |
| OURENSE | 44335.0 | 56336.0 | 49925.0 | 0.0 | 0.0 | 60140.0 | 89073.0 | 686178.0 |
| PALENCIA | 26318.0 | 75120.0 | 60033.0 | 0.0 | 43643.0 | 79110.0 | 101424.0 | 833481.0 |
| PALMAS, LAS | 127163.0 | 73423.0 | 154835.0 | 0.0 | 0.0 | 63089.0 | 102447.0 | 15957372.0 |
| PONTEVEDRA | 149868.0 | 79717.0 | 85757.0 | 0.0 | 0.0 | 80610.0 | 110670.0 | 2570185.0 |
| RIOJA, LA | 45990.0 | 86168.0 | 55710.0 | 0.0 | 55381.0 | 81180.0 | 116361.0 | 473760.0 |
| SALAMANCA | 87769.0 | 83510.0 | 176843.0 | 0.0 | 38832.0 | 79315.0 | 107378.0 | 5431378.0 |
| SANTA CRUZ DE TENERIFE | 39041.0 | 71659.0 | 93011.0 | 0.0 | 0.0 | 64234.0 | 98719.0 | 3080266.0 |
| SEGOVIA | 23098.0 | 77486.0 | 65448.0 | 0.0 | 37791.0 | 76568.0 | 109413.0 | 850010.0 |
| SEVILLA | 321534.0 | 98582.0 | 157131.0 | 0.0 | 45688.0 | 81395.0 | 123000.0 | 11341152.0 |
| SORIA | 9606.0 | 65102.0 | 53356.0 | 0.0 | 0.0 | 68271.0 | 90195.0 | 423132.0 |
| TARRAGONA | 56705.0 | 71530.0 | 90800.0 | 0.0 | 0.0 | 62710.0 | 103124.0 | 2563288.0 |
| TERUEL | 12090.0 | 63734.0 | 61455.0 | 0.0 | 0.0 | 61625.0 | 92906.0 | 933320.0 |
| TOLEDO | 99449.0 | 64688.0 | 67848.0 | 0.0 | 32230.0 | 58816.0 | 86453.0 | 3988595.0 |
| VALENCIA | 374474.0 | 71623.0 | 147717.0 | 0.0 | 33919.0 | 62379.0 | 94880.0 | 25547252.0 |
| VALLADOLID | 127977.0 | 90180.0 | 63615.0 | 0.0 | 56373.0 | 86319.0 | 116795.0 | 2257086.0 |
| ZAMORA | 27245.0 | 61070.0 | 83727.0 | 0.0 | 0.0 | 61911.0 | 86200.0 | 1536265.0 |
| ZARAGOZA | 184413.0 | 93023.0 | 107493.0 | 0.0 | 49078.0 | 87634.0 | 123219.0 | 8516913.0 |
| other | 32973.0 | 777.0 | 15372.0 | 0.0 | 0.0 | 0.0 | 0.0 | 725642.0 |

- Median income of the customers of all the products is almost same
- We can see gross house hold income of Ceuta is the highest

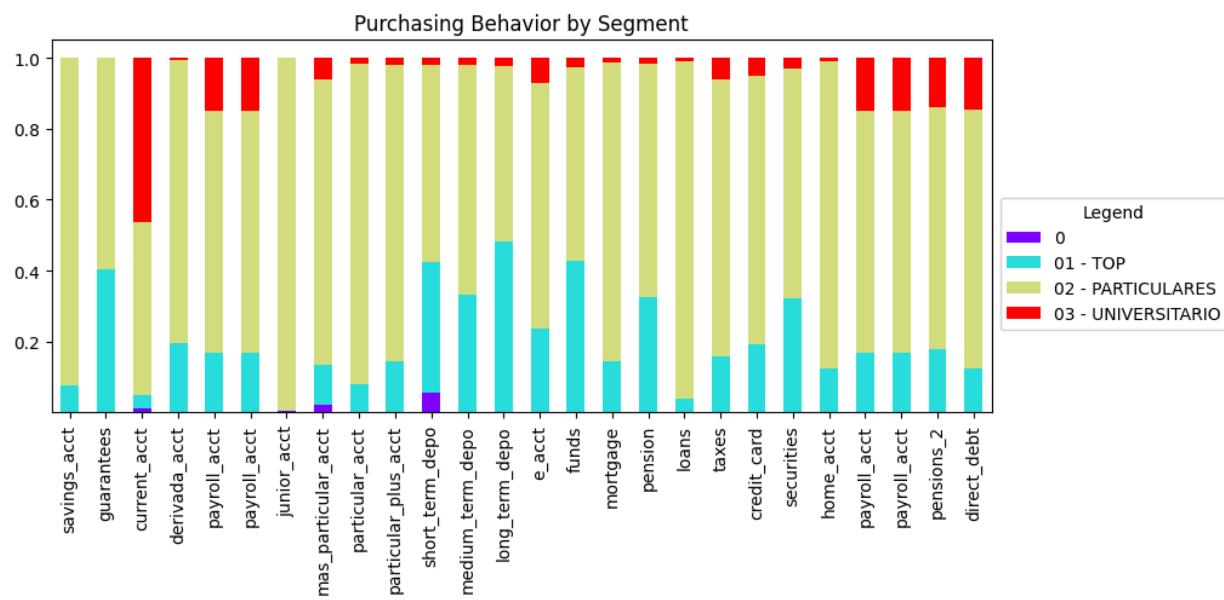
In [53]:

```
segmentation = train['segment'].value_counts()
seg_products = train.groupby('segment')['total_products'].sum()
seg_products = (seg_products/seg_products.sum())*100
seg_products

plt.figure(figsize=(15,8))
plt.subplot(1,2,1)
plt.pie(segmentation, labels=segmentation.keys(), autopct='%1.2f%%')
plt.title('Segmentation Distribution')
plt.subplot(1,2,2)
plt.pie(seg_products, labels=seg_products.index, autopct='%1.2f%%')
plt.title('Purchasing Behavior by Segmentation')
plt.show()
```




In [54]:
```
print(train['segment'].value_counts())

plot_grouped_data(train, 'segment', products, title='Purchasing Behavior by Segment',
plt.show()
```

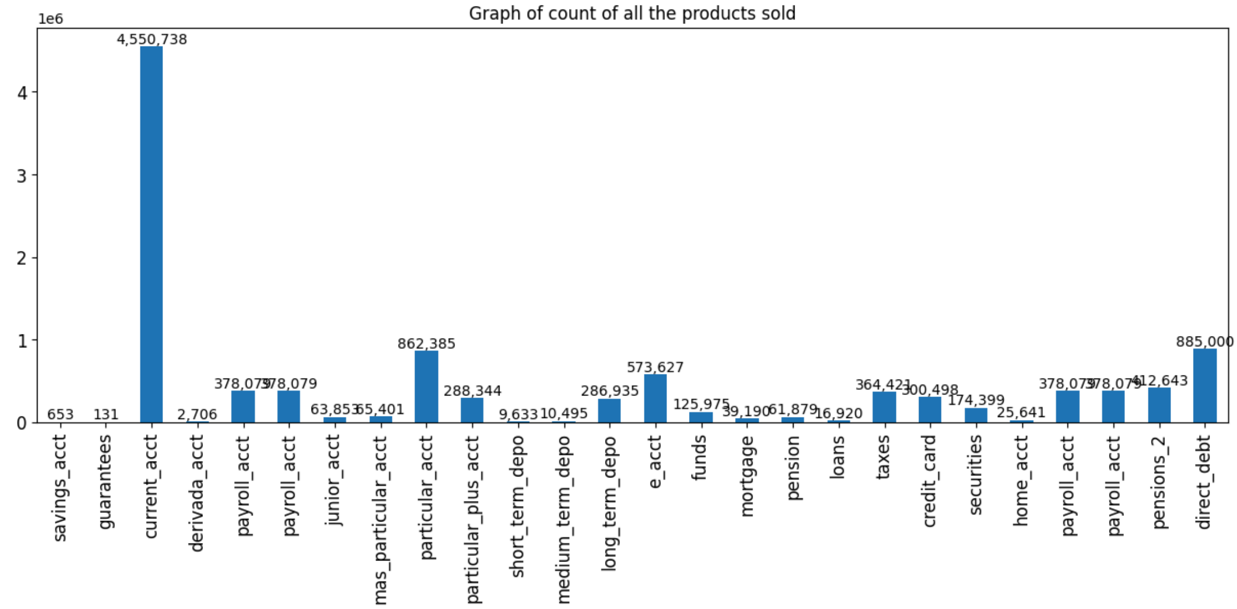
```
segment
02 - PARTICULARES     4293981
03 - UNIVERSITARIO    2659297
01 - TOP               284472
0                      112954
Name: count, dtype: int64
```




- Most customers are from segment #2 - Particulares
- The count of clients on each segment is corrrelated with the products the customers of that segments have bought
- All the customers who have bought juniour account belong to segment 02

## Count of All Products Sold

In [55]:
```
ax = train[products].sum().plot(kind='bar', figsize=(15, 5), fontsize=12)
plt.title('Graph of count of all the products sold')
for container in ax.containers:
    ax.bar_label(container, fmt='{:,.0f}')
# plt.xticks(rotation = 45)
plt.show()
```




- Current account, particular, direct debit and e_account are the most popular accounts.
- Savings account, guarentees, derivada account, short and medium term deposits are the least popular accounts

# Make Data Model Ready

In [56]:
```
train = train.rename(columns={'sex_H': 'female'})
val_set = val_set.rename(columns={'sex_H': 'female'})
test_set = test_set.rename(columns={'sex_H': 'female'})
```

## Data Processing

The first problem we identified was: Non-primary and deceased are only 0.2% of the dataset which imbalances the dataset and may not produce significant results, so we decided to drop them.
In addition to that, since we will be working only with primary customers, the column last_date_primary is not helpful anymore, so we will drop it as well.

In [57]:
```
#Dropping rows with primary_cust == 99
train = train[train['primary_cust'] != 99]
test_set = test_set[test_set['primary_cust'] != 99]
val_set = val_set[val_set['primary_cust'] != 99]

print((train['primary_cust'] == 99).value_counts())
```

```
primary_cust
False    7336142
Name: count, dtype: int64
```

Deceased clients have a few products, but they are not going to buy any more products -which is not valuable to our ultimate goal of predicting which products clients will buy, so we will drop the rows of deceased clients and drop the column

In [58]:
```
#Dropping rows with deceased == 1
# train = train[train['deceased'] == 0]
# test_set = test_set[test_set['deceased'] == 0]
# val_set = val_set[val_set['deceased'] == 0]

print((train['deceased'] == 1).value_counts())
print((test_set['deceased'] == 1).value_counts())
print((val_set['deceased'] == 1).value_counts())
```

```
deceased
False    7336142
Name: count, dtype: int64
deceased
False    1571978
Name: count, dtype: int64
deceased
False    1572033
Name: count, dtype: int64
```

We will drop columns where age is 0 and over 100. We believe these clients are not going to be valuable on our model

In [59]:
```
print(((train['age'] == 0) | (train['age'] > 100)).value_counts())
```

```
age
False    7328925
True        7217
Name: count, dtype: int64
```

In [60]:
```
train = train[(train['age'] > 0) & (train['age'] <= 100)]
test_set = test_set[(test_set['age'] > 0) & (test_set['age'] <= 100)]
val_set = val_set[(val_set['age'] > 0) & (val_set['age'] <= 100)]

print(((train['age'] == 0) | (train['age'] > 100)).value_counts())
```

```
age
False    7328925
Name: count, dtype: int64
```

We will drop the rows on column seniority_in_months with values -999999 and null values under province_name (that were previously filled up with 'other') since those variables cause some noise on the data

In [61]:
```
train = train[train['seniority_in_months'] != -999999]
train = train[train['province_name'] != 'other']

test_set = test_set[test_set['seniority_in_months'] != -999999]
test_set = test_set[test_set['province_name'] != 'other']
```

```
val_set = val_set[val_set['seniority_in_months'] != -999999]
val_set = val_set[val_set['province_name'] != 'other']
```

We had the impression the columns seniority in months and first contract date were giving us the same information and we decided to check the correlation. Turns out it is highly correlated, so we'll keep the column seniority in months.

In [62]:
```
correlation = train[['seniority_in_months', 'first_contract_date']].corr()
correlation
```

Out[62]:

| | seniority_in_months | first_contract_date |
|---|---|---|
| **seniority_in_months** | 1.000000 | -0.965954 |
| **first_contract_date** | -0.965954 | 1.000000 |

Dropping primary_customer , last_date_primary, deceased and first_contract_date columns since after cleaning the dataset, they do not provide any additional information

In [63]:
```
train = train.drop(columns=['primary_cust', 'last_date_primary', 'deceased', 'first_co
test_set = test_set.drop(columns=['primary_cust', 'last_date_primary', 'deceased', 'fi
val_set = val_set.drop(columns=['primary_cust', 'last_date_primary', 'deceased', 'firs
```

In [64]:
```
(train['income'] == 0).value_counts()
```

Out[64]:
```
income
False    5772084
True     1523888
Name: count, dtype: int64
```

There is a lot of missing values for the income variable. We checked if it makes sense for us to fill those missing incomes with the median income per province, however we still got too many null values, so we will drop the income = 0 rows

In [65]:
```
train = train[train['income'] != 0]
test_set = test_set[test_set['income'] != 0]
val_set = val_set[val_set['income'] != 0]

print(train.shape)
print(test_set.shape)
print(val_set.shape)
```

```
(5772084, 44)
(1236744, 43)
(1236546, 43)
```

In [66]:
```
train.isnull().sum()
```

Out[66]:
```
date                       0
customer_code              0
employee_index             0
country                    0
female                     0
age                        0
new_cust                   0
seniority_in_months        0
cust_type                  0
cust_relationship          0
residency_spain            0
birth_spain                0
join_channel               0
province_name              0
active_cust                0
income                     0
segment                    0
savings_acct               0
guarantees                 0
current_acct               0
derivada_acct              0
payroll_acct               0
junior_acct                0
mas_particular_acct        0
particular_acct            0
particular_plus_acct       0
short_term_depo            0
medium_term_depo           0
long_term_depo             0
e_acct                     0
funds                      0
mortgage                   0
pension                    0
loans                      0
taxes                      0
credit_card                0
securities                 0
home_acct                  0
payroll_acct               0
pensions_2                 0
direct_debt                0
total_products             0
age_group               1426
first_contract_age         0
dtype: int64
```

In [67]:
```
# Checking unique variables for cust_type
print(train['cust_type'].unique())
print(train['cust_type'].isna().sum())
```

```
['1' '0' '2' '3' 'P' '4']
0
```

In [68]:
```
# Fixing cust_type variables so they are consistent across rows
train['cust_type'] = train['cust_type'].astype(str).str.strip()
cust_type_map = {'0.0': '0', '1.0': '1', '2.0': '2', '3.0': '3', '4.0': '4'}
train['cust_type'] = train['cust_type'].replace(cust_type_map)

test_set['cust_type'] = test_set['cust_type'].astype(str).str.strip()
test_set['cust_type'] = test_set['cust_type'].replace(cust_type_map)
```

```
val_set['cust_type'] = val_set['cust_type'].astype(str).str.strip()
val_set['cust_type'] = val_set['cust_type'].replace(cust_type_map)

train['cust_type'] = train['cust_type'].astype(object)
test_set['cust_type'] = test_set['cust_type'].astype(object)
val_set['cust_type'] = val_set['cust_type'].astype(object)

print(train['cust_type'].value_counts())
print(train['cust_relationship'].value_counts())
```

```
cust_type
1    5748876
0      22238
3        642
2        165
P        122
4         41
Name: count, dtype: int64
cust_relationship
I    3202187
A    2546854
0      22238
P        683
R        122
Name: count, dtype: int64
```

In [69]:
```
train.describe().round(2)
#Check for 0s, outliers - that are too different from the 4th quartile
```

Out[69]:

| | date | customer_code | age | new_cust | seniority_in_months | active_cust | |
|---|---|---|---|---|---|---|---|
| **count** | 5772084 | 5772084.00 | 5772084.00 | 5772084.00 | 5772084.00 | 5772084.00 | 5 |
| **mean** | 2015-12-16 16:01:43.477912832 | 811258.77 | 40.77 | 0.03 | 83.32 | 0.44 | |
| **min** | 2015-06-28 00:00:00 | 15889.00 | 2.00 | 0.00 | 0.00 | 0.00 | |
| **25%** | 2015-09-28 00:00:00 | 435897.50 | 25.00 | 0.00 | 26.00 | 0.00 | |
| **50%** | 2015-12-28 00:00:00 | 907045.00 | 40.00 | 0.00 | 54.00 | 0.00 | |
| **75%** | 2016-03-28 00:00:00 | 1180245.00 | 51.00 | 0.00 | 140.00 | 1.00 | |
| **max** | 2016-05-28 00:00:00 | 1454620.00 | 100.00 | 1.00 | 256.00 | 1.00 | 28 |
| **std** | NaN | 424061.32 | 17.18 | 0.18 | 66.35 | 0.50 | |

Plotting outliers

In [70]:
```
# Create a variable with numerical columns to plot them easier
numeric_col = train.select_dtypes(include=['int', 'float']).columns.tolist()
```

```python
numeric_col = [col for col in numeric_col if col not in products]
numeric_col
```

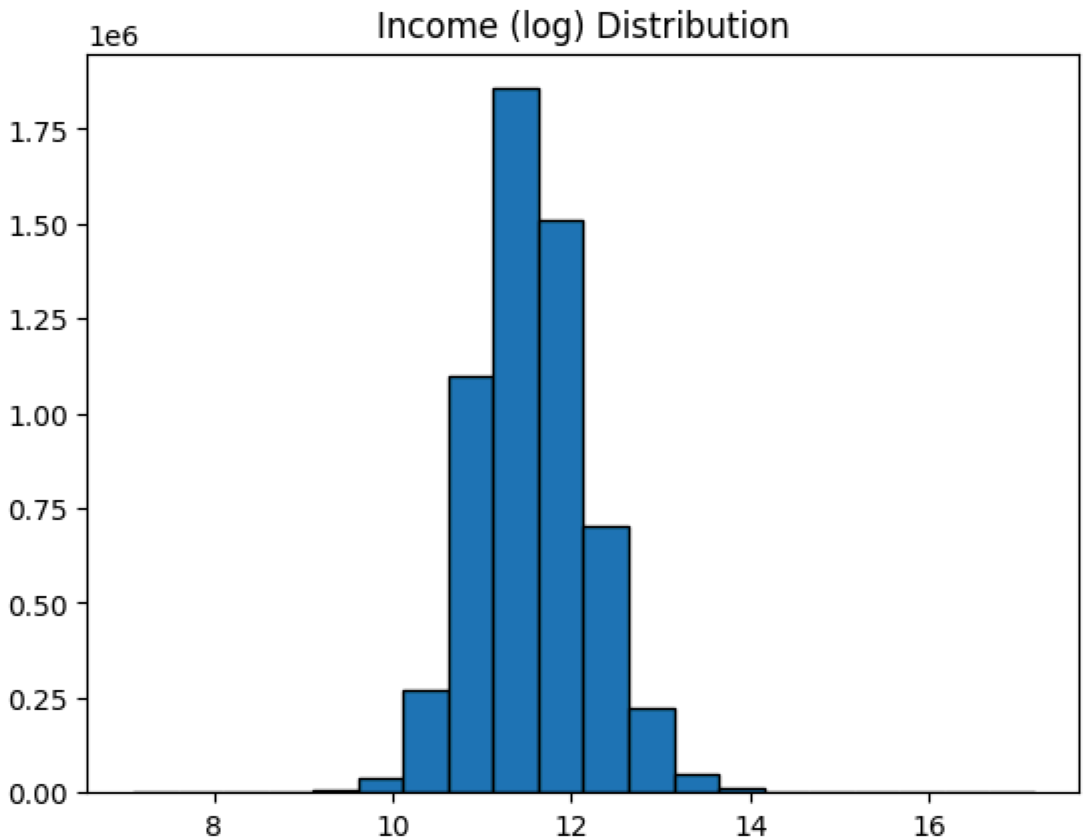
Out[70]:
```
['customer_code',
 'age',
 'new_cust',
 'seniority_in_months',
 'active_cust',
 'income',
 'total_products',
 'first_contract_age']
```

In [71]:
```python
# dummy = pd.DataFrame(columns=['age', 'seniority_in_months', 'income', 'product'])

# for col in products:
#     df = pd.DataFrame({
#         'age': train.age[train[col] == 1],
#         'seniority_in_months': train.seniority_in_months[train[col] == 1],
#         'income': np.log1p(train.income[train[col] == 1]),
#         'product': col
#     })

# dummy = pd.concat([dummy, df], ignore_index=True)
```

In [72]:
```python
# plt.figure(figsize=(15, 18))

# # Plot for Age
# plt.subplot(3, 1, 1)
# sns.boxplot(data=dummy, x='product', y='age', palette='rainbow')
# plt.title('Distribution of Age by Product')
# plt.xticks(rotation=90)

# # Plot for Seniority in Months
# plt.subplot(3, 1, 2)
# sns.boxplot(data=dummy, x='product', y='seniority_in_months', palette='rainbow')
# plt.title('Distribution of Seniority in Months by Product')
# plt.xticks(rotation=90)

# # Plot for Income
# plt.subplot(3, 1, 3)
# sns.boxplot(data=dummy, x='product', y='income', palette='rainbow')
# plt.title('Distribution of Log Income by Product')
# plt.xticks(rotation=90)

# plt.tight_layout()
# plt.show()
```

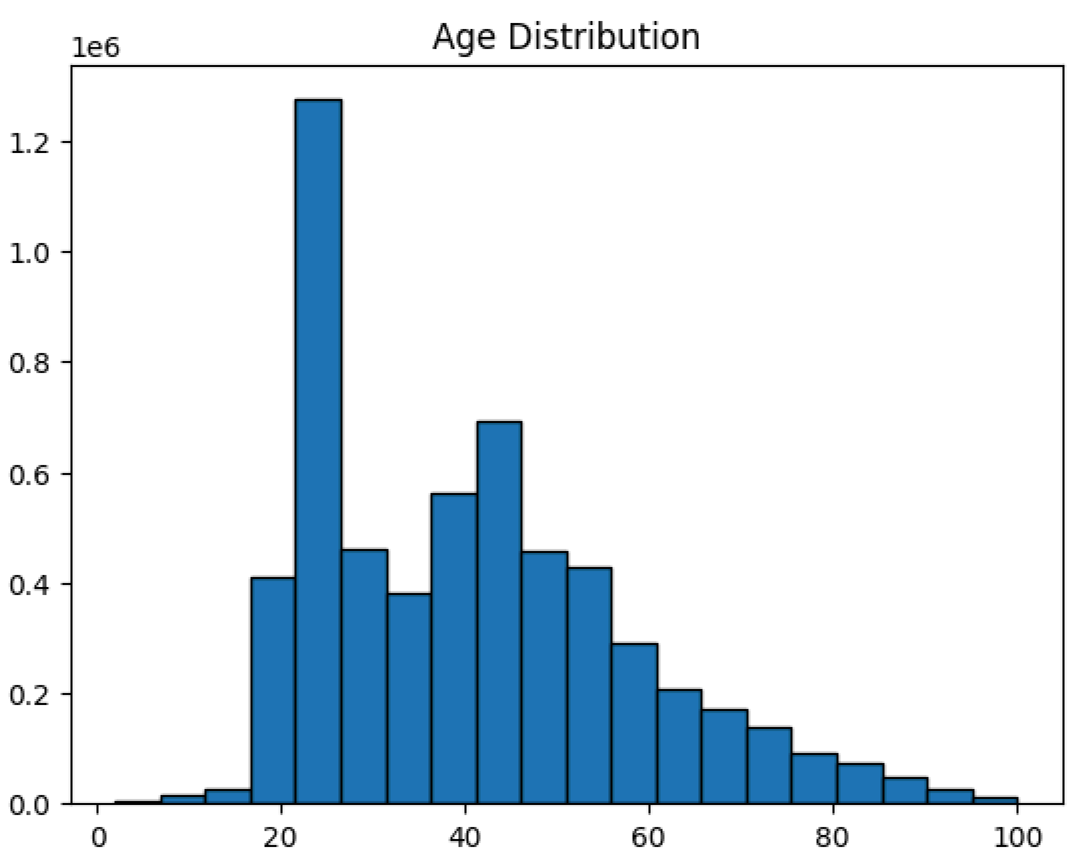
In [73]:
```python
plt.hist(np.log(train['income']), bins=20, edgecolor='black')
plt.title('Income (log) Distribution')
```

Out[73]:
```
Text(0.5, 1.0, 'Income (log) Distribution')
```




In [74]:
```
plt.hist(train['age'], bins=20, edgecolor='black')
plt.title('Age Distribution')
```

Out[74]:
```
Text(0.5, 1.0, 'Age Distribution')
```



Transform categorical variables with label encoder to plot correlation. This will not be a final transformation, it is just to detect correlation

In [75]:
```
le = LabelEncoder()

correlation = train.copy()

columns_to_encode = ['employee_index', 'country', 'cust_type', 'cust_relationship',
                     'join_channel', 'province_name', 'segment']

for col in columns_to_encode:
    correlation[col] = le.fit_transform(correlation[col])
```

In [76]:
```
# plt.figure(figsize=(25,15))
# sns.heatmap(correlation.corr(),annot=True, cmap="YlGnBu", fmt= '.2f')
# plt.show()
```

There are two correlations that draw attention to us. The first one is active_cust and cust_relationship, which is 0.81. The second one is payroll_acct and payroll_acct.1 which is 1. They will both be dropped, the payroll_acct.1 gives us the same information as the payroll_acct and we believe that having such a high correlation between active_cust and cust_relationship would hinder our model

# Dealing with the categorical values

Since most of the data is from clients that comes from Spain, we will combine non-spanish clients into an "others" group so we will have only two variables: ES and others. After that we will transform the column in a dummy variable for Spain.

In [77]:
```
train['country'] = np.where(train['country'] == 'ES', 1, 0)
test_set['country'] = np.where(test_set['country'] == 'ES', 1, 0)
val_set['country'] = np.where(val_set['country'] == 'ES', 1, 0)
```

We will transform the segment column using get_dummies to create two different binary columns for the variables

In [78]:
```
segment_dummy = pd.get_dummies(train['segment'],drop_first=True)
train = pd.concat([train, segment_dummy], axis=1)

segment_dummy_test = pd.get_dummies(test_set['segment'],drop_first=True)
test_set = pd.concat([test_set, segment_dummy_test], axis=1)

segment_dummy_val = pd.get_dummies(val_set['segment'],drop_first=True)
val_set = pd.concat([val_set, segment_dummy_val], axis=1)
```

Since total_products is a continuous variable representing how many products each customer has purchased, we will use it as a target for encoding the categorical variables join_channel and province_name, since those have many different variables. We chose target encoding because we would have a dimensionality problem if we chose to transform each variable in a dummy column, and we did not want to use label encoder since it is not an ordinal variable

In [79]:
```
target_encoder = ce.TargetEncoder(cols=['join_channel', 'province_name', 'employee_inc

train[['join_channel_encoded', 'province_name_encoded', 'employee_index_encoded']] = t
    train[['join_channel', 'province_name', 'employee_index']], train['total_products'

# Apply the same transformation to the test and validation set (use transform, not fit
test_set[['join_channel_encoded', 'province_name_encoded', 'employee_index_encoded']]
    test_set[['join_channel', 'province_name', 'employee_index']])

val_set[['join_channel_encoded', 'province_name_encoded', 'employee_index_encoded']] =
    val_set[['join_channel', 'province_name', 'employee_index']])

# You can now inspect the transformed columns in test and val_set
print(test_set[['join_channel', 'province_name', 'join_channel_encoded', 'province_nam
print(val_set[['join_channel', 'province_name', 'join_channel_encoded', 'province_name
print(train[['join_channel', 'province_name', 'join_channel_encoded', 'province_name_e
```

```
       join_channel province_name  join_channel_encoded  \
70935           KAT     TARRAGONA              2.131292
90608           KHE     CORUÑA, A              0.909825
286157          KHK        MALAGA              1.268794
361018          KHQ        BURGOS              0.868212
13108           KHE     CASTELLON              0.909825

        province_name_encoded  employee_index_encoded
70935                1.257727                1.519747
90608                1.189736                1.519747
286157               1.358065                1.519747
361018               1.302032                1.519747
13108                1.302684                1.519747
       join_channel province_name  join_channel_encoded  \
218355          KAF       SEVILLA              2.191395
255866          KAT        MADRID              2.131292
117532          KHE      ZARAGOZA              0.909825
280037          KAT        MADRID              2.131292
19770           KFC      VALENCIA              1.705759

        province_name_encoded  employee_index_encoded
218355               1.491526                1.519747
255866               1.922745                1.519747
117532               1.273754                1.519747
280037               1.922745                1.519747
19770                1.319347                1.519747
       join_channel province_name  join_channel_encoded  \
207640          KHE        MADRID              0.909825
116039          KHE     BARCELONA              0.909825
300836          KFC     BARCELONA              1.705759
361717          KFC      VALENCIA              1.705759
102962          KHE     BARCELONA              0.909825

        province_name_encoded  employee_index_encoded
207640               1.922745                1.519747
116039               1.268091                1.519747
300836               1.268091                1.519747
361717               1.319347                1.519747
102962               1.268091                1.519747
```

In [80]:
```
train['cust_type'] = train['cust_type'].replace({"P": 5})
train['cust_type'] = train['cust_type'].astype(int)
```

```
test_set['cust_type'] = test_set['cust_type'].replace({"P": 5})
test_set['cust_type'] = test_set['cust_type'].astype(int)

val_set['cust_type'] = val_set['cust_type'].replace({"P": 5})
val_set['cust_type'] = val_set['cust_type'].astype(int)
```

Normalization/Standartization of data

We will normalize data for age and seniority in months since it is not normally distributed

We will standardize income since its log is normally distributed

In [81]:
```
# Normalize using MinMaxScaler on training data
cols_to_normalize = ['age', 'seniority_in_months']
min_max_scaler = MinMaxScaler(feature_range=(0, 1))
train[cols_to_normalize] = min_max_scaler.fit_transform(train[cols_to_normalize])
# Apply the fitted MinMaxScaler to test and validation sets
test_set[cols_to_normalize] = min_max_scaler.transform(test_set[cols_to_normalize])
val_set[cols_to_normalize] = min_max_scaler.transform(val_set[cols_to_normalize])

# Standardize using StandardScaler on training data
cols_to_standardize = ['income']
standard_scaler = StandardScaler()
train[cols_to_standardize] = standard_scaler.fit_transform(train[cols_to_standardize])
# Apply the fitted StandardScaler to test and validation sets
test_set[cols_to_standardize] = standard_scaler.transform(test_set[cols_to_standardize
val_set[cols_to_standardize] = standard_scaler.transform(val_set[cols_to_standardize])

# Output results to verify
print("Training data (normalized):\n", train[cols_to_normalize].head())
print("Training data (standardized):\n", train[cols_to_standardize].head())

print("Test data (normalized):\n", test_set[cols_to_normalize].head())
print("Test data (standardized):\n", test_set[cols_to_standardize].head())

print("Validation data (normalized):\n", val_set[cols_to_normalize].head())
print("Validation data (standardized):\n", val_set[cols_to_standardize].head())
```

```
Training data (normalized):
              age  seniority_in_months
207640  0.234694             0.070312
116039  0.234694             0.140625
300836  0.306122             0.300781
361717  0.489796             0.644531
102962  0.224490             0.210938
Training data (standardized):
           income
207640  2.498775
116039 -0.208421
300836 -0.292548
361717 -0.209325
102962  0.419622
Test data (normalized):
              age  seniority_in_months
70935   0.734694             0.785156
90608   0.214286             0.113281
286157  0.244898             0.011719
361018  0.183673             0.023438
13108   0.234694             0.156250
Test data (standardized):
           income
70935  -0.374282
90608  -0.171172
286157 -0.358730
361018 -0.094848
13108  -0.354137
Validation data (normalized):
              age  seniority_in_months
218355  0.418367             0.527344
255866  0.377551             0.484375
117532  0.234694             0.199219
280037  0.755102             0.902344
19770   0.540816             0.363281
Validation data (standardized):
           income
218355 -0.364616
255866 -0.004342
117532 -0.309610
280037  0.423733
19770   0.083943
```

# Feature Engineering

In [82]:
```python
# Convert all boolean columns (True/False) to integers (1/0)
train = train.map(lambda x: int(x) if isinstance(x, bool) else x)
```

In [83]:
```python
test_set = test_set.map(lambda x: int(x) if isinstance(x, bool) else x)
```

In [84]:
```python
val_set = val_set.map(lambda x: int(x) if isinstance(x, bool) else x)
```

In [85]:
```python
# Display the first few rows to check the changes
print(train.head())
print(test_set.head())
print(val_set.head())
```

```
              date  customer_code employee_index  country female       age  \
207640 2016-04-28        1334092              N        1      0  0.234694   
116039 2015-07-28        1024586              N        1      0  0.234694   
300836 2016-04-28         856204              N        1      0  0.306122   
361717 2015-08-28         295807              N        1      0  0.489796   
102962 2016-03-28         942624              N        1      1  0.224490   

        new_cust  seniority_in_months  cust_type cust_relationship  \
207640         0             0.070312          1                 I   
116039         0             0.140625          1                 A   
300836         0             0.300781          1                 I   
361717         0             0.644531          1                 A   
102962         0             0.210938          1                 I   

       residency_spain birth_spain join_channel province_name  active_cust  \
207640               1           0          KHE        MADRID            0   
116039               1           0          KHE     BARCELONA            1   
300836               1           0          KFC     BARCELONA            0   
361717               1           0          KFC      VALENCIA            1   
102962               1           0          KHE     BARCELONA            0   

          income             segment  savings_acct  guarantees  current_acct  \
207640  2.498775  03 - UNIVERSITARIO             0           0             1   
116039 -0.208421  03 - UNIVERSITARIO             0           0             1   
300836 -0.292548   02 - PARTICULARES             0           0             1   
361717 -0.209325   02 - PARTICULARES             0           0             1   
102962  0.419622  03 - UNIVERSITARIO             0           0             0   

        derivada_acct  payroll_acct  junior_acct  mas_particular_acct  \
207640              0             0            0                    0   
116039              0             0            0                    0   
300836              0             0            0                    0   
361717              0             0            0                    0   
102962              0             0            0                    0   

        particular_acct  particular_plus_acct  short_term_depo  \
207640                0                     0                0   
116039                0                     0                0   
300836                0                     0                0   
361717                0                     0                0   
102962                0                     0                0   

        medium_term_depo  long_term_depo  e_acct  funds  mortgage  pension  \
207640                 0               0       0      0         0        0   
116039                 0               0       0      0         0        0   
300836                 0               0       0      0         0        0   
361717                 0               0       0      0         0        0   
102962                 0               0       0      0         0        0   

        loans  taxes  credit_card  securities  home_acct  payroll_acct  \
207640      0      0            0           0          0             0   
116039      0      0            0           0          0             0   
300836      0      0            0           0          0             0   
361717      0      0            0           0          0             0   
102962      0      0            0           0          0             0   

        pensions_2  direct_debt  total_products age_group  first_contract_age  \
207640           0            0               1  [20, 30)                23.0   
116039           0            0               1  [20, 30)                22.0   
300836           0            0               1  [30, 40)                26.0   
```

```
361717           0           0              1  [50, 60)               36.0
102962           0           0              0  [20, 30)               20.0

        01 - TOP  02 - PARTICULARES  03 - UNIVERSITARIO  join_channel_encoded  \
207640        0                  0                   1              0.909825
116039        0                  0                   1              0.909825
300836        0                  1                   0              1.705759
361717        0                  1                   0              1.705759
102962        0                  0                   1              0.909825

        province_name_encoded  employee_index_encoded
207640               1.922745                1.519747
116039               1.268091                1.519747
300836               1.268091                1.519747
361717               1.319347                1.519747
102962               1.268091                1.519747
             date  customer_code employee_index  country female       age  \
70935  2015-06-28          49335              N        1      1  0.734694
90608  2016-02-28        1174349              N        1      0  0.214286
286157 2015-07-28        1393286              N        1      0  0.244898
361018 2016-03-28        1454346              N        1      0  0.183673
13108  2016-02-28        1074431              N        1      0  0.234694

        new_cust  seniority_in_months  cust_type cust_relationship  \
70935          0             0.785156          1                 A
90608          0             0.113281          1                 I
286157         1             0.011719          1                 A
361018         0             0.023438          1                 A
13108          0             0.156250          1                 I

       residency_spain birth_spain join_channel province_name  active_cust  \
70935                1           0          KAT     TARRAGONA            1
90608                1           0          KHE     CORUÑA, A            0
286157               1           0          KHK        MALAGA            1
361018               1           0          KHQ        BURGOS            1
13108                1           0          KHE     CASTELLON            0

          income             segment  savings_acct  guarantees  current_acct  \
70935  -0.374282   02 - PARTICULARES             0           0             1
90608  -0.171172  03 - UNIVERSITARIO             0           0             1
286157 -0.358730  03 - UNIVERSITARIO             0           0             1
361018 -0.094848  03 - UNIVERSITARIO             0           0             1
13108  -0.354137  03 - UNIVERSITARIO             0           0             1

        derivada_acct  payroll_acct  junior_acct  mas_particular_acct  \
70935               0             0            0                    0
90608               0             0            0                    0
286157              0             0            0                    0
361018              0             0            0                    0
13108               0             0            0                    0

        particular_acct  particular_plus_acct  short_term_depo  \
70935                 1                     0                0
90608                 0                     0                0
286157                0                     0                0
361018                0                     0                0
13108                 0                     0                0

        medium_term_depo  long_term_depo  e_acct  funds  mortgage  pension  \
70935                  0               0       0      0         0        0
```

```
90608                0              0       0      0         0        0
286157               0              0       0      0         0        0
361018               0              0       0      0         0        0
13108                0              0       0      0         0        0

        loans  taxes  credit_card  securities  home_acct  payroll_acct  \
70935       0      0            0           0          0             0
90608       0      0            0           0          0             0
286157      0      0            0           0          0             0
361018      0      0            0           0          0             0
13108       0      0            0           0          0             0

        pensions_2  direct_debt  total_products  first_contract_age  01 - TOP  \
70935            0            0               2                57.0         0
90608            0            0               1                21.0         0
286157           0            0               1                26.0         0
361018           0            0               1                20.0         0
13108            0            0               1                22.0         0

        02 - PARTICULARES  03 - UNIVERSITARIO  join_channel_encoded  \
70935                   1                   0              2.131292
90608                   0                   1              0.909825
286157                  0                   1              1.268794
361018                  0                   1              0.868212
13108                   0                   1              0.909825

        province_name_encoded  employee_index_encoded
70935                1.257727                1.519747
90608                1.189736                1.519747
286157               1.358065                1.519747
361018               1.302032                1.519747
13108                1.302684                1.519747
             date  customer_code employee_index  country female       age  \
218355 2015-12-28         487783              N        1      1  0.418367
255866 2016-01-28         556611              N        1      0  0.377551
117532 2015-11-28         929949              N        1      0  0.234694
280037 2015-10-28          44208              N        1      0  0.755102
19770  2015-10-28         745042              N        1      0  0.540816

        new_cust  seniority_in_months  cust_type cust_relationship  \
218355         0             0.527344          1                 A
255866         0             0.484375          1                 A
117532         0             0.199219          1                 I
280037         0             0.902344          1                 A
19770          0             0.363281          1                 A

       residency_spain birth_spain join_channel province_name  active_cust  \
218355               1           0          KAF       SEVILLA            1
255866               1           0          KAT        MADRID            1
117532               1           0          KHE      ZARAGOZA            0
280037               1           0          KAT        MADRID            1
19770                1           0          KFC      VALENCIA            1

          income             segment  savings_acct  guarantees  current_acct  \
218355 -0.364616   02 - PARTICULARES             0           0             0
255866 -0.004342   02 - PARTICULARES             0           0             1
117532 -0.309610  03 - UNIVERSITARIO             0           0             1
280037  0.423733   02 - PARTICULARES             0           0             0
19770   0.083943   02 - PARTICULARES             0           0             1
```

```
        derivada_acct  payroll_acct  junior_acct  mas_particular_acct  \
218355              0             0            0                    0   
255866              0             0            0                    0   
117532              0             0            0                    0   
280037              0             0            0                    0   
19770               0             0            0                    0   

        particular_acct  particular_plus_acct  short_term_depo  \
218355                0                     0                0   
255866                0                     1                0   
117532                0                     0                0   
280037                0                     0                0   
19770                 0                     0                0   

        medium_term_depo  long_term_depo  e_acct  funds  mortgage  pension  \
218355                 0               0       0      0         1        0   
255866                 0               1       0      0         0        0   
117532                 0               0       0      0         0        0   
280037                 0               0       1      0         0        0   
19770                  0               0       0      0         0        0   

        loans  taxes  credit_card  securities  home_acct  payroll_acct  \
218355      0      0            0           0          0             0   
255866      0      0            0           1          0             0   
117532      0      0            0           0          0             0   
280037      0      0            0           1          0             0   
19770       0      0            0           1          0             0   

        pensions_2  direct_debt  total_products  first_contract_age  01 - TOP  \
218355           0            0               1                32.0         0   
255866           0            0               4                29.0         0   
117532           0            0               1                21.0         0   
280037           1            1               4                57.0         0   
19770            0            0               2                47.0         0   

        02 - PARTICULARES  03 - UNIVERSITARIO  join_channel_encoded  \
218355                  1                   0              2.191395   
255866                  1                   0              2.131292   
117532                  0                   1              0.909825   
280037                  1                   0              2.131292   
19770                   1                   0              1.705759   

        province_name_encoded  employee_index_encoded  
218355               1.491526                1.519747  
255866               1.922745                1.519747  
117532               1.273754                1.519747  
280037               1.922745                1.519747  
19770                1.319347                1.519747  
```

In [86]:

```
train = train.rename(columns={'country': 'country_spain'})
test_set = test_set.rename(columns={'country': 'country_spain'})
val_set = val_set.rename(columns={'country': 'country_spain'})
```

In [87]:

```
print(train.shape)
print(train.columns)
print(test_set.shape)
print(test_set.columns)
print(val_set.shape)
print(val_set.columns)
```

```
(5772084, 50)
Index(['date', 'customer_code', 'employee_index', 'country_spain', 'female',
       'age', 'new_cust', 'seniority_in_months', 'cust_type',
       'cust_relationship', 'residency_spain', 'birth_spain', 'join_channel',
       'province_name', 'active_cust', 'income', 'segment', 'savings_acct',
       'guarantees', 'current_acct', 'derivada_acct', 'payroll_acct',
       'junior_acct', 'mas_particular_acct', 'particular_acct',
       'particular_plus_acct', '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', 'total_products', 'age_group',
       'first_contract_age', '01 - TOP', '02 - PARTICULARES',
       '03 - UNIVERSITARIO', 'join_channel_encoded', 'province_name_encoded',
       'employee_index_encoded'],
      dtype='object')
(1236744, 49)
Index(['date', 'customer_code', 'employee_index', 'country_spain', 'female',
       'age', 'new_cust', 'seniority_in_months', 'cust_type',
       'cust_relationship', 'residency_spain', 'birth_spain', 'join_channel',
       'province_name', 'active_cust', 'income', 'segment', 'savings_acct',
       'guarantees', 'current_acct', 'derivada_acct', 'payroll_acct',
       'junior_acct', 'mas_particular_acct', 'particular_acct',
       'particular_plus_acct', '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', 'total_products', 'first_contract_age',
       '01 - TOP', '02 - PARTICULARES', '03 - UNIVERSITARIO',
       'join_channel_encoded', 'province_name_encoded',
       'employee_index_encoded'],
      dtype='object')
(1236546, 49)
Index(['date', 'customer_code', 'employee_index', 'country_spain', 'female',
       'age', 'new_cust', 'seniority_in_months', 'cust_type',
       'cust_relationship', 'residency_spain', 'birth_spain', 'join_channel',
       'province_name', 'active_cust', 'income', 'segment', 'savings_acct',
       'guarantees', 'current_acct', 'derivada_acct', 'payroll_acct',
       'junior_acct', 'mas_particular_acct', 'particular_acct',
       'particular_plus_acct', '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', 'total_products', 'first_contract_age',
       '01 - TOP', '02 - PARTICULARES', '03 - UNIVERSITARIO',
       'join_channel_encoded', 'province_name_encoded',
       'employee_index_encoded'],
      dtype='object')
```

# New variables

Income to Age Ratio: This metric helps identify customers who might have high disposable income.

In [ ]:

```python
# 2. Income to Age Ratio
train['income_to_age'] = train['income'] / (train['age'] + 1e-5)  # Avoid division by
train['income_to_age']

val_set['income_to_age'] = val_set['income'] / (val_set['age'] + 1e-5)  # Avoid divisi
val_set['income_to_age']
```

```
test_set['income_to_age'] = test_set['income'] / (test_set['age'] + 1e-5)  # Avoid div
test_set['income_to_age']
```

# Modeling approach

In [ ]:
```
# train.head()
```

In [ ]:
```
# val_set.head()
```

In [ ]:
```
# test_set.head()
```

Changing columns name and dropping columns so both datasets are the same

In [ ]:
```
drop = ['join_channel', 'province_name', 'employee_index', 'segment', 'total_products'
train = train.drop(columns=drop + ['age_group'])
val_set = val_set.drop(columns=drop)
test_set = test_set.drop(columns=drop)
```

## Reading into the data

Setting products we want to predict

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

# Transformation #1

Change #1: Instead of dropping these duplicates on customer column and use only the last instance we will keep those duplicates since it could capture some patterns such as if a client buys product x first, it will likely buy y product next.

We will create copies of the original train and test datasets so we don't change the original one.

In [ ]:
```
# train_2 = train.copy()
# test_2 = test_set.copy()
```

# Pre-processing

# Transformation #2

For tranformation #2 we will add the date column as one of the features. For that, we will calculate the time since purchase using the month we are trying to predict on June 2016. For this transformation to make sense, we will also keep the first transformation, since the time line of purchase matters now, we will keep the duplicate clients' purchases instead of only keeping the last one

In [ ]:
```
train['date'] = pd.to_datetime(train['date'], format='%Y-%m-%d')

train['date'] = train['date'].dt.to_period('M').dt.to_timestamp()

# Setting our prediction date, June 28, 2016, as the reference date
reference_date = pd.to_datetime("2016-06-28")

# Calculate time since purchase
train['months_since_purchase'] = (reference_date.year - train['date'].dt.year) * 12 +
                                  (reference_date.month - train['date'].dt.month)

# print(train[['date', 'months_since_purchase']])
```

In [ ]:
```
# Adding feature on test dateased
test_set['date'] = pd.to_datetime(test_set['date'], format='%Y-%m-%d')
test_set['date'] = test_set['date'].dt.to_period('M').dt.to_timestamp()

test_set['months_since_purchase'] = (reference_date.year - test_set['date'].dt.year) *
                            (reference_date.month - test_set['date'].dt.month)

# print(test_set[['date', 'months_since_purchase']])
```

In [ ]:
```
X_train = train.drop(['customer_code', 'date'] + products, axis=1)
y_train = train[products]

X_test = test_set.drop(['customer_code', 'date'] + products, axis=1)
y_test = test_set[products]
```

# Training

In [ ]:
```
# Defining the best training parameter
params = {'C': 10, 'solver': 'liblinear', 'max_iter': 300}
```

Database with second transformation

In [ ]:
```
# Initialize dictionary for storing metrics
metrics = defaultdict(lambda: defaultdict(dict))

# Train and evaluate the model on the 'train_2' dataset
for product in products:
    clf = LogisticRegression(**params)

    # Train data and labels for each product
    y_train_product = y_train[product].values
    y_test_product = y_test[product].values

    # Train the model
    clf.fit(X_train, y_train_product)
```

```
    # Predictions
    y_train_pred = clf.predict(X_train)
    y_test_pred = clf.predict(X_test)
    y_train_pred_proba = clf.predict_proba(X_train)[:, 1]
    y_test_pred_proba = clf.predict_proba(X_test)[:, 1]

    # Calculate metrics
    metrics['train']['train'][product] = {
        'ROC AUC': roc_auc_score(y_train_product, y_train_pred_proba),
        'F1 Score': f1_score(y_train_product, y_train_pred),
        'Confusion Matrix': confusion_matrix(y_train_product, y_train_pred)
    }

    metrics['train']['test'][product] = {
        'ROC AUC': roc_auc_score(y_test_product, y_test_pred_proba),
        'F1 Score': f1_score(y_test_product, y_test_pred),
        'Confusion Matrix': confusion_matrix(y_test_product, y_test_pred)
    }
```

In [ ]:
```
# Summarize the average metrics across all products
summary_data = []
for dataset in ['train', 'test']:
    avg_roc_auc = np.mean([metrics['train'][dataset][p]['ROC AUC'] for p in products])
    avg_f1 = np.mean([metrics['train'][dataset][p]['F1 Score'] for p in products])
    summary_data.append(['train', dataset, avg_roc_auc, avg_f1])

# Create summary DataFrame
summary_df = pd.DataFrame(summary_data, columns=['Dataset', 'Type', 'Avg ROC AUC', 'Av
print("\nEvaluated Model on Dataset: train_2")
print(summary_df.to_string(index=False))
```

```
Evaluated Model on Dataset: train_2
Dataset  Type  Avg ROC AUC  Avg F1 Score
train_2 train     0.876659      0.084174
train_2  test     0.874053      0.185602
```

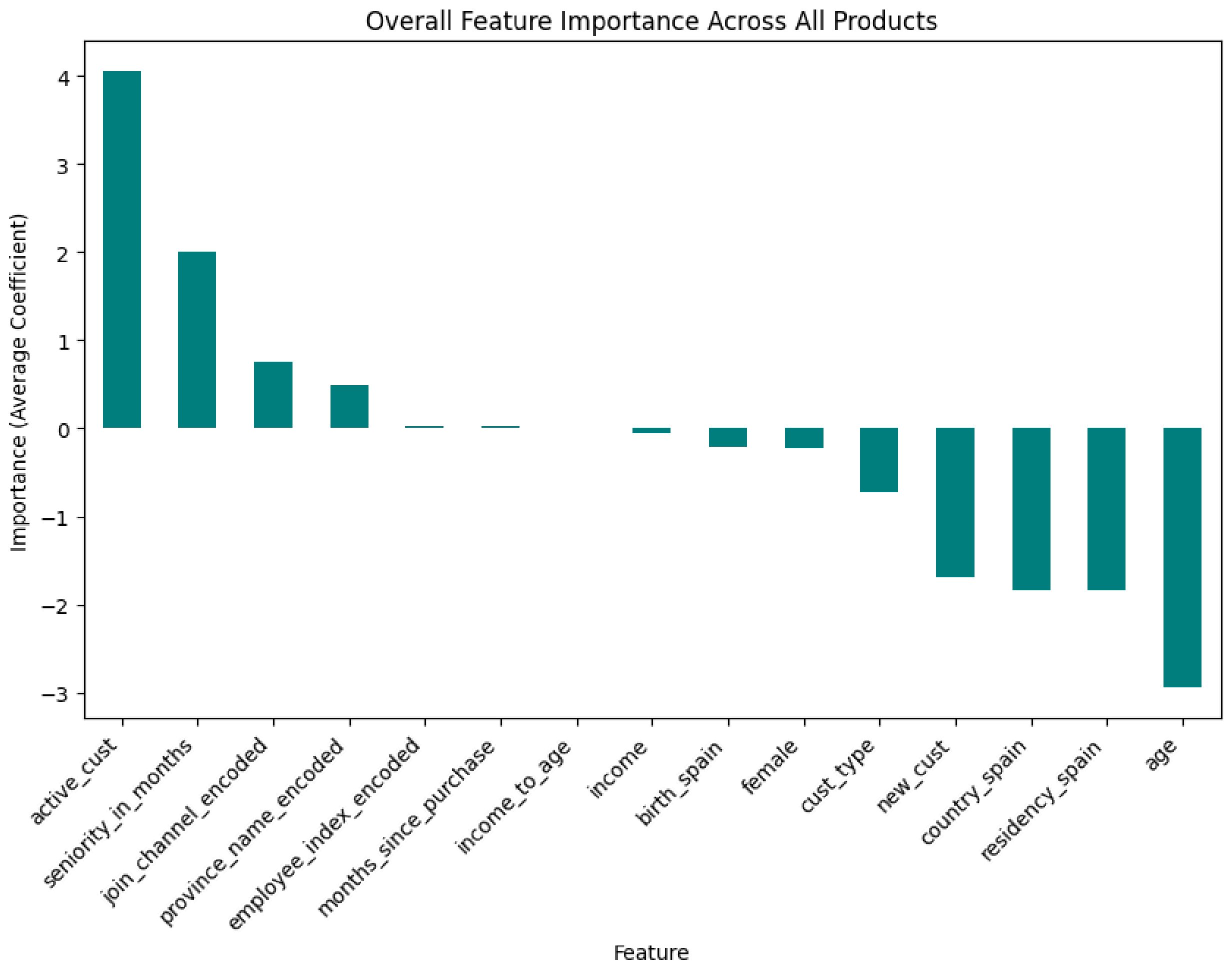
# Feature Importance Analysis

In [ ]:
```
# Dictionary to store feature importances
feature_importances = {}

# Iterate over each product
for product in products:
    clf = LogisticRegression(**params)
    clf.fit(X_train, y_train[product].values)

    feature_importances[product] = clf.coef_[0]

importances_df = pd.DataFrame(feature_importances, index=X_train.columns)
importances_df['Mean_Importance'] = importances_df.mean(axis=1)
sorted_importances = importances_df.sort_values(by='Mean_Importance', ascending=False)
```

In [ ]:
```
sorted_importances
```

| | savings_acct | guarantees | current_acct | derivada_acct | payroll_acct | junior_ac |
|---|---|---|---|---|---|---|
| **active_cust** | 0.830705 | 4.264000 | 1.414410 | 2.427595 | 6.842190 | 3.03010 |
| **seniority_in_months** | 4.603687 | 3.920184 | 0.148217 | 1.993498 | 0.956385 | 5.37899 |
| **join_channel_encoded** | 0.909934 | -0.647531 | -1.445911 | 0.368749 | 1.030798 | 2.54202 |
| **province_name_encoded** | -0.805320 | 6.485177 | -0.823748 | -0.552544 | 0.626762 | 0.89281 |
| **employee_index_encoded** | 0.250569 | 0.528488 | 0.143572 | 0.657579 | 0.122628 | 0.48636 |
| **months_since_purchase** | 0.022000 | 0.026316 | 0.024692 | 0.010854 | 0.002302 | 0.03050 |
| **income_to_age** | 0.000004 | -0.000041 | -0.000020 | 0.000002 | -0.000012 | -0.00007 |
| **income** | 0.029702 | 0.014780 | -0.000532 | 0.003247 | -0.060334 | -0.01263 |
| **birth_spain** | 0.615684 | -3.797005 | -0.192945 | -0.808166 | 0.135772 | -0.09866 |
| **female** | -0.581870 | -0.833165 | 0.029119 | -1.419952 | 0.013041 | 0.00647 |
| **cust_type** | -3.278677 | -5.275618 | 0.531083 | -0.524133 | 0.487662 | -0.87524 |
| **new_cust** | -2.920691 | -3.626413 | -0.168880 | -1.249166 | -0.418773 | -0.66440 |
| **country_spain** | -1.702266 | -5.265817 | 0.772075 | -2.432985 | -3.752419 | 0.37180 |
| **residency_spain** | -1.702266 | -5.265817 | 0.772075 | -2.432985 | -3.752419 | 0.37180 |
| **age** | -2.054414 | -4.500836 | -0.370727 | 1.399406 | -2.178210 | -69.70432 |

In [ ]:
```
plt.figure(figsize=(10, 6))
sorted_importances['Mean_Importance'].plot(kind='bar', color='teal')
plt.title('Overall Feature Importance Across All Products')
plt.ylabel('Importance (Average Coefficient)')
plt.xlabel('Feature')
plt.xticks(rotation=45, ha='right')
plt.show()
```

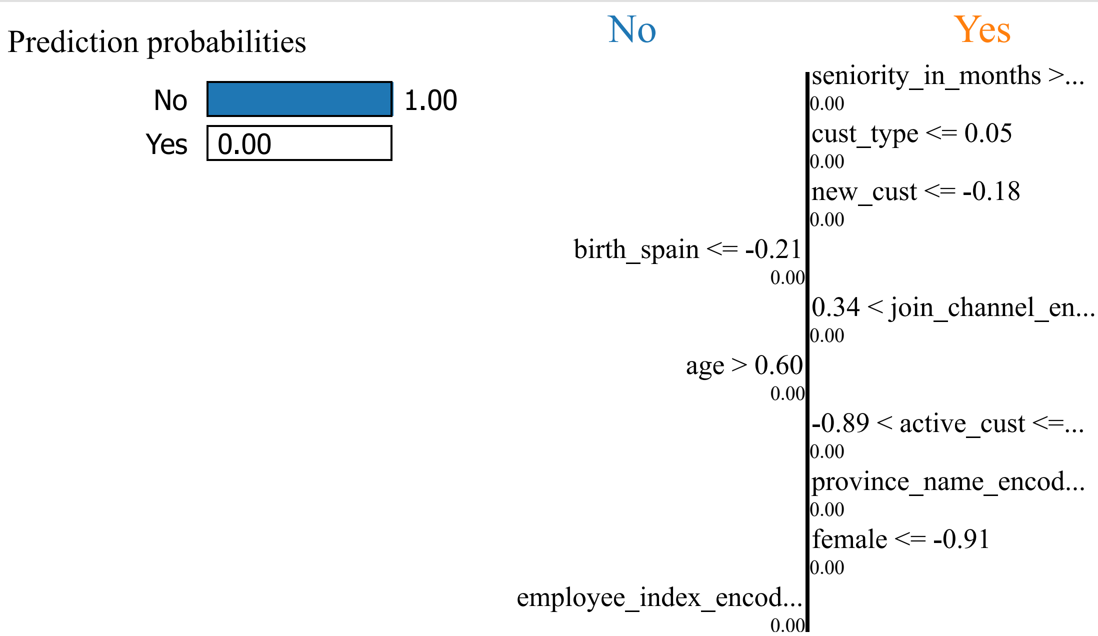



# LIME example on variable 'savings_acct'

Here we are trying to get deeper and understand which variables affect the variable 'savings_acct' the most. For the purpose of this exercise we will just run the code on 'savings_acct'

In [ ]:
```
example_product = 'savings_acct'

#Standardizing features for LIME
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

clf = LogisticRegression(**params)
clf.fit(X_train_scaled, y_train[example_product].values)

# Initializing LIME Tabular Explainer
explainer = LimeTabularExplainer(
    training_data=X_train_scaled,
    training_labels=y_train[example_product],
    feature_names=X_train.columns,
    class_names=['No', 'Yes'],
    mode='classification'
)

# Choosing an instance to explain
```

```python
instance_idx = 0
instance = X_test_scaled[instance_idx]

# Generating explanation
exp = explainer.explain_instance(
    data_row=instance,
    predict_fn=clf.predict_proba  # Probability prediction function
)
exp.show_in_notebook(show_table=True)  # For Jupyter Notebook
# exp.save_to_file('lime_explanation.html')  # Save to HTML file
```




| Feature | Value |
|---|---|
| seniority_in_months | 1.78 |
| cust_type | 0.05 |
| new_cust | -0.18 |
| birth_spain | -0.21 |
| join_channel_encoded | 1.21 |
| age | 1.95 |
| active_cust | 1.12 |
| province_name_encoded | -0.88 |
| female | -0.91 |

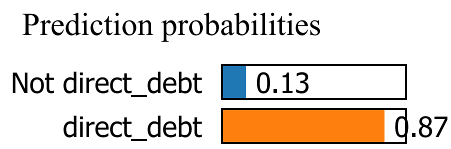
Select 5 predictions at random, explain how the model generated those predictions (which features matter more than others), which features need to change and by how much to move the output in a significant way (e.g., to flip the prediction from one class to another)

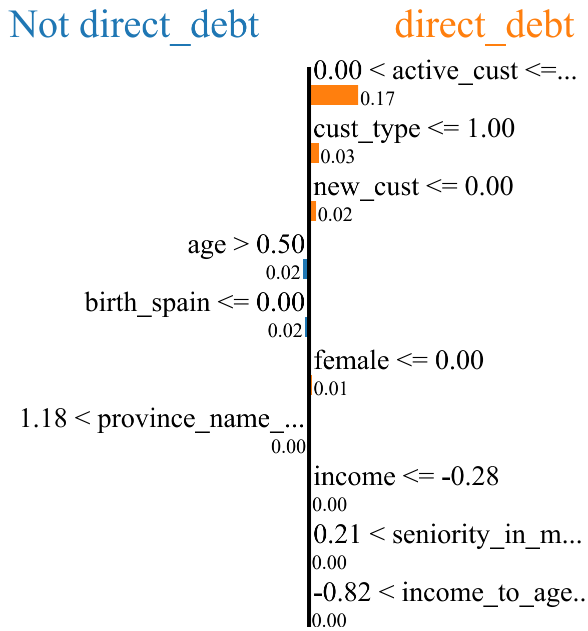
In [ ]:
```python
# Randomly selecting 5 samples
random_indices = random.sample(range(X_test.shape[0]), 5)
selected_samples = X_test.iloc[random_indices]
selected_labels = y_test[product].iloc[random_indices]
```

```python
# LIME explainer
explainer = lime.lime_tabular.LimeTabularExplainer(
    X_train.values,
    feature_names=X_train.columns,
    class_names=[f"Not {product}", product],
    verbose=True,
    mode="classification"
)

# Explain each sample
for i, idx in enumerate(random_indices):
    print(f"\nExplaining Prediction {i+1} (Row {idx}):")
    sample = X_test.iloc[idx].values

    # Probability for the sample
    prediction = clf.predict_proba([sample])[0]
    predicted_class = np.argmax(prediction)
    print(f"Predicted Class: {predicted_class} (Probability: {prediction[predicted_cla

    # Generate explanation
    exp = explainer.explain_instance(sample, clf.predict_proba, num_features=10)
    exp.show_in_notebook(show_table=True)

    # Print explanation in text format
    explanation = exp.as_list()
    print("Feature Contributions:")
    for feature, contribution in explanation:
        print(f"{feature}: {contribution:.4f}")

    # Identify feature changes to flip prediction
    important_feature, contribution = explanation[0]
    print(f"\nTo flip the prediction, try adjusting '{important_feature}' by a signifi

# Visualize the explanation for one of the samples
exp.show_in_notebook(show_table=True)
```

```
Explaining Prediction 1 (Row 950916):
Predicted Class: 1 (Probability: 0.87)
Intercept -0.039240922130545156
Prediction_local [0.15935189]
Right: 0.8716594547837396
```

```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\bas
e.py:464: UserWarning: X does not have valid feature names, but LogisticRegression wa
s fitted with feature names
  warnings.warn(
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\bas
e.py:464: UserWarning: X does not have valid feature names, but LogisticRegression wa
s fitted with feature names
  warnings.warn(
```

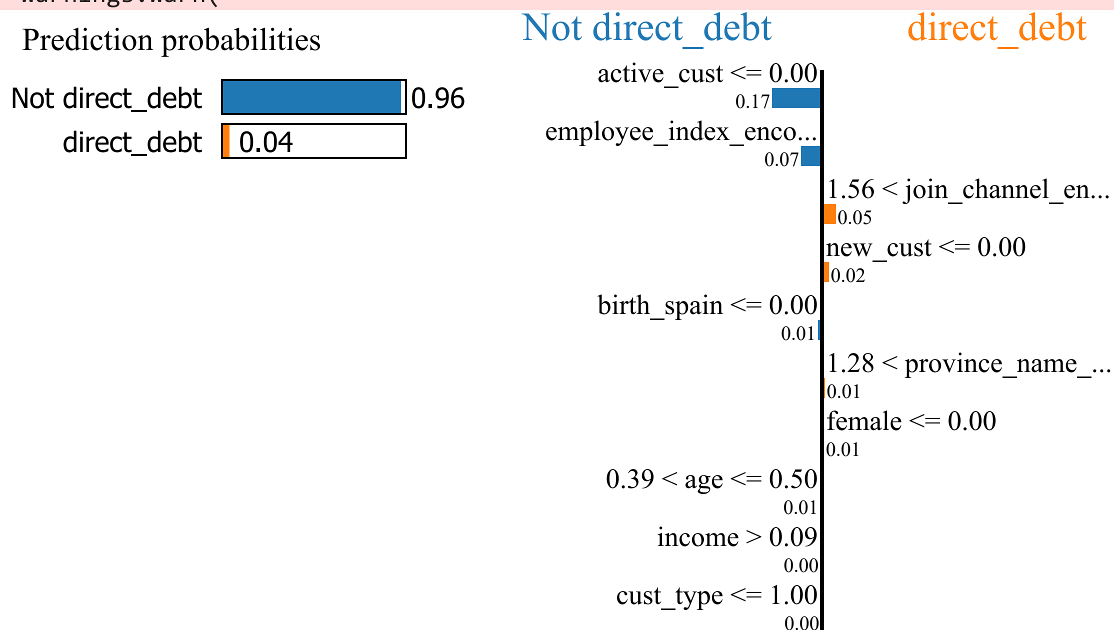



| Feature | Value |
|---|---|
| active_cust | 1.00 |
| cust_type | 1.00 |
| new_cust | 0.00 |
| age | 0.66 |
| birth_spain | 0.00 |
| female | 0.00 |
| province_name_encoded | 1.24 |
| income | -0.46 |
| seniority_in_months | 0.54 |

```
Feature Contributions:
0.00 < active_cust <= 1.00: 0.1687
cust_type <= 1.00: 0.0320
new_cust <= 0.00: 0.0229
age > 0.50: -0.0219
birth_spain <= 0.00: -0.0151
female <= 0.00: 0.0084
1.18 < province_name_encoded <= 1.28: -0.0037
income <= -0.28: 0.0034
0.21 < seniority_in_months <= 0.54: 0.0023
-0.82 < income_to_age <= -0.35: 0.0016

To flip the prediction, try adjusting '0.00 < active_cust <= 1.00' by a significant a
mount.

Explaining Prediction 2 (Row 3239):
Predicted Class: 0 (Probability: 0.96)
Intercept 0.21402389014112455
Prediction_local [0.03316361]
Right: 0.037688068101520936
```

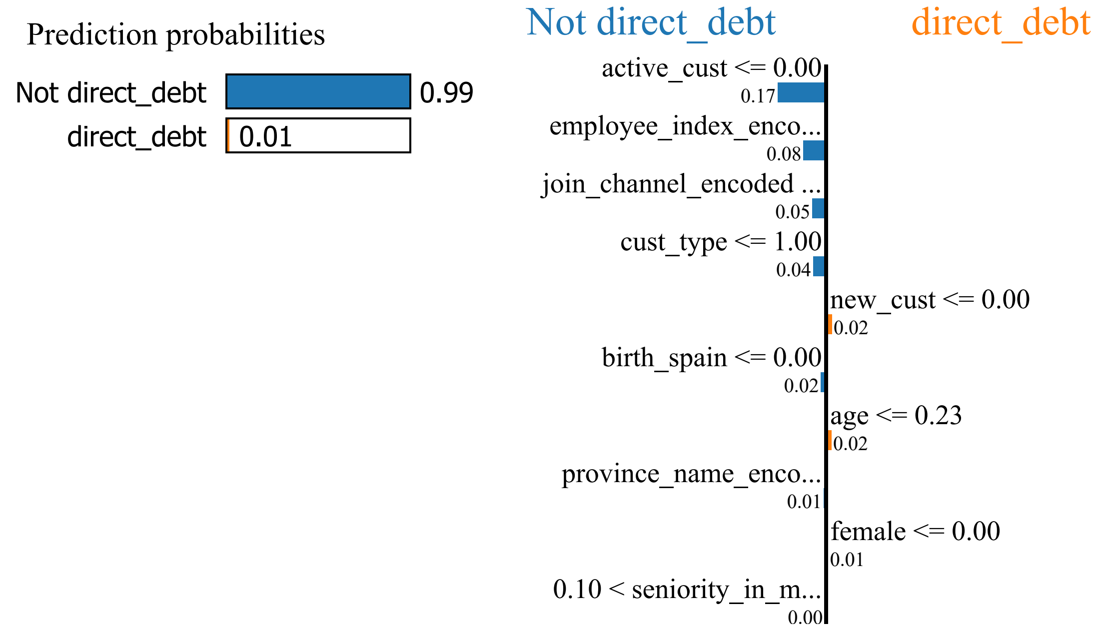
```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\bas
e.py:464: UserWarning: X does not have valid feature names, but LogisticRegression wa
s fitted with feature names
  warnings.warn(
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\bas
e.py:464: UserWarning: X does not have valid feature names, but LogisticRegression wa
s fitted with feature names
  warnings.warn(
```




| Feature | Value |
|---|---|
| active_cust | 0.00 |
| employee_index_encoded | 1.41 |
| join_channel_encoded | 1.81 |
| new_cust | 0.00 |
| birth_spain | 0.00 |
| province_name_encoded | 1.75 |
| female | 0.00 |
| age | 0.50 |
| income | 0.45 |

```
Feature Contributions:
active_cust <= 0.00: -0.1708
employee_index_encoded <= 1.41: -0.0710
1.56 < join_channel_encoded <= 1.94: 0.0472
new_cust <= 0.00: 0.0232
birth_spain <= 0.00: -0.0136
1.28 < province_name_encoded <= 1.75: 0.0093
female <= 0.00: 0.0069
0.39 < age <= 0.50: -0.0056
income > 0.09: -0.0043
cust_type <= 1.00: -0.0021

To flip the prediction, try adjusting 'active_cust <= 0.00' by a significant amount.

Explaining Prediction 3 (Row 458596):
Predicted Class: 0 (Probability: 0.99)
Intercept 0.30015283526118897
Prediction_local [-0.02265368]
Right: 0.014329623063726801
```

```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\bas
e.py:464: UserWarning: X does not have valid feature names, but LogisticRegression wa
s fitted with feature names
  warnings.warn(
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\bas
e.py:464: UserWarning: X does not have valid feature names, but LogisticRegression wa
s fitted with feature names
  warnings.warn(
```



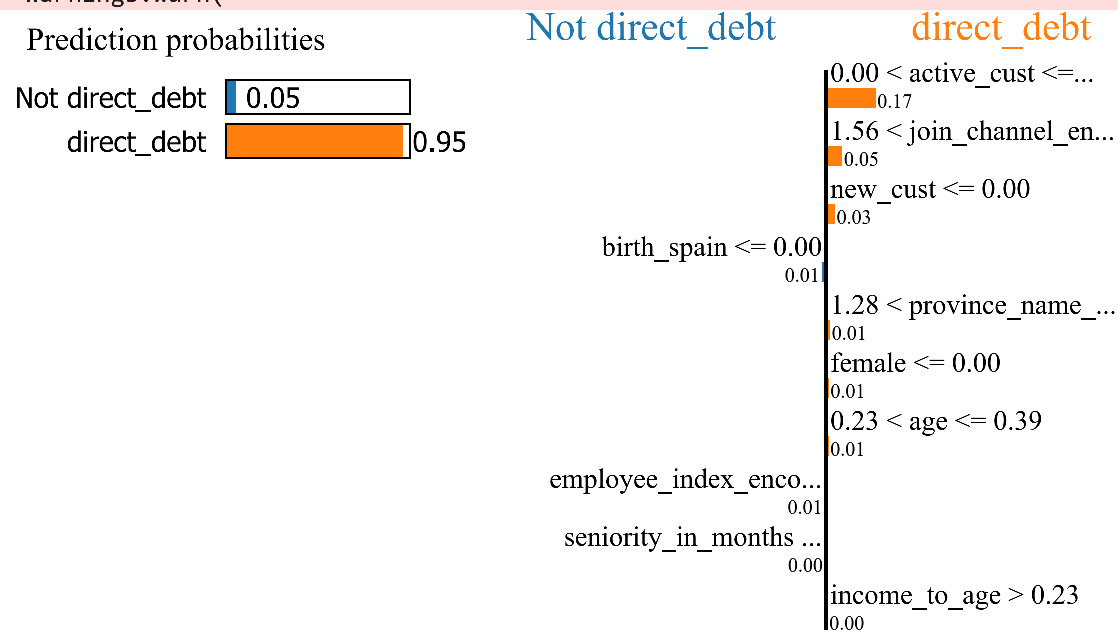

| Feature | Value |
|---|---|
| active_cust | 0.00 |
| employee_index_encoded | 1.41 |
| join_channel_encoded | 0.89 |
| cust_type | 1.00 |
| new_cust | 0.00 |
| birth_spain | 0.00 |
| age | 0.20 |
| province_name_encoded | 1.18 |
| female | 0.00 |

```
Feature Contributions:
active_cust <= 0.00: -0.1668
employee_index_encoded <= 1.41: -0.0780
join_channel_encoded <= 0.89: -0.0475
cust_type <= 1.00: -0.0443
new_cust <= 0.00: 0.0195
birth_spain <= 0.00: -0.0178
age <= 0.23: 0.0173
province_name_encoded <= 1.18: -0.0074
female <= 0.00: 0.0067
0.10 < seniority_in_months <= 0.21: -0.0044

To flip the prediction, try adjusting 'active_cust <= 0.00' by a significant amount.

Explaining Prediction 4 (Row 1115023):
```

```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\bas
e.py:464: UserWarning: X does not have valid feature names, but LogisticRegression wa
s fitted with feature names
  warnings.warn(
```

```
Predicted Class: 1 (Probability: 0.95)
Intercept -0.036817469834049066
Prediction_local [0.22245744]
Right: 0.9472943668943028
```

```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
```



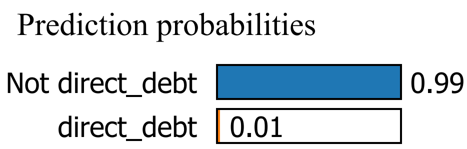

| Feature | Value |
|---|---|
| active_cust | 1.00 |
| join_channel_encoded | 1.94 |
| new_cust | 0.00 |
| birth_spain | 0.00 |
| province_name_encoded | 1.75 |
| female | 0.00 |
| age | 0.36 |
| employee_index_encoded | 1.41 |
| seniority_in_months | 0.05 |

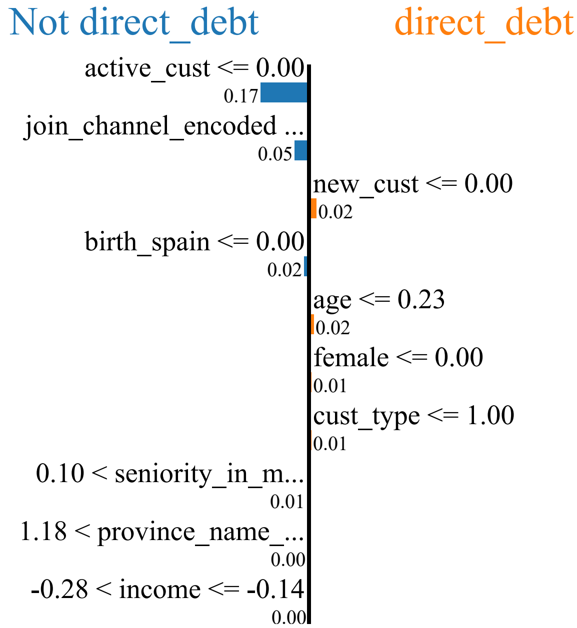
```
Feature Contributions:
0.00 < active_cust <= 1.00: 0.1695
1.56 < join_channel_encoded <= 1.94: 0.0546
new_cust <= 0.00: 0.0291
birth_spain <= 0.00: -0.0150
1.28 < province_name_encoded <= 1.75: 0.0127
female <= 0.00: 0.0079
0.23 < age <= 0.39: 0.0071
employee_index_encoded <= 1.41: -0.0055
seniority_in_months <= 0.10: -0.0045
income_to_age > 0.23: 0.0035

To flip the prediction, try adjusting '0.00 < active_cust <= 1.00' by a significant a
mount.

Explaining Prediction 5 (Row 557414):
Predicted Class: 0 (Probability: 0.99)
```

```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\bas
e.py:464: UserWarning: X does not have valid feature names, but LogisticRegression wa
s fitted with feature names
  warnings.warn(
```

```
Intercept 0.1629545436552834
Prediction_local [-0.02360619]
Right: 0.014531659456663631
```

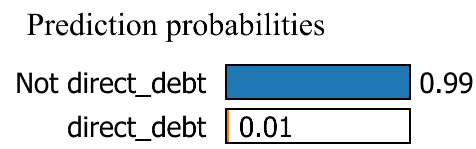
```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\bas
e.py:464: UserWarning: X does not have valid feature names, but LogisticRegression wa
s fitted with feature names
  warnings.warn(
```

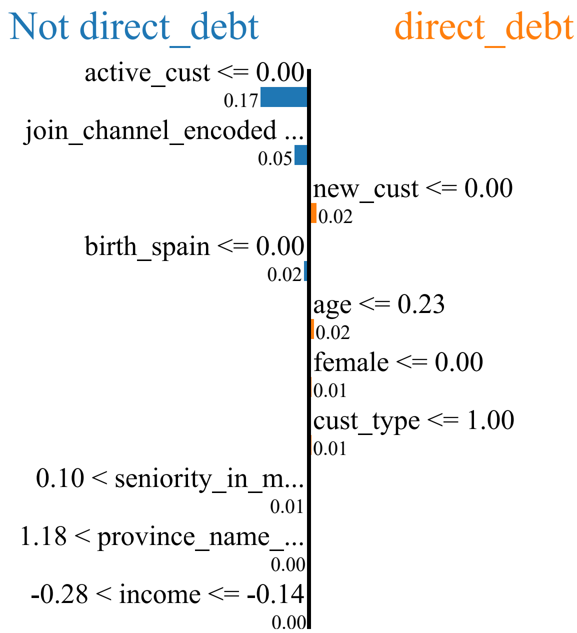



| Feature | Value |
|---|---|
| active_cust | 0.00 |
| join_channel_encoded | 0.89 |
| new_cust | 0.00 |
| birth_spain | 0.00 |
| age | 0.23 |
| female | 0.00 |
| cust_type | 1.00 |
| seniority_in_months | 0.21 |
| province_name_encoded | 1.28 |

```
Feature Contributions:
active_cust <= 0.00: -0.1665
join_channel_encoded <= 0.89: -0.0483
new_cust <= 0.00: 0.0249
birth_spain <= 0.00: -0.0164
age <= 0.23: 0.0161
female <= 0.00: 0.0079
cust_type <= 1.00: 0.0074
0.10 < seniority_in_months <= 0.21: -0.0063
1.18 < province_name_encoded <= 1.28: -0.0042
-0.28 < income <= -0.14: -0.0012

To flip the prediction, try adjusting 'active_cust <= 0.00' by a significant amount.
```




| Feature | Value |
|---|---|
| active_cust | 0.00 |
| join_channel_encoded | 0.89 |
| new_cust | 0.00 |
| birth_spain | 0.00 |
| age | 0.23 |
| female | 0.00 |
| cust_type | 1.00 |
| seniority_in_months | 0.21 |
| province_name_encoded | 1.28 |