Week 3 — Perform Exploratory Data Analysis

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
```

Load datasets.

Kaggle already gave us the dataset split in train/test, but we will not load it for now since the model is not supposed to see this data.

We will split the train dataset in train/validation

In [2]:
```
pd.set_option('display.max_columns', None)

train = pd.read_csv('train_final.csv')
# test = pd.read_csv('test_ver2.csv') #Kaggle already gave us the dataset split in tra
```

```
C:\Users\MARIA\AppData\Local\Temp\ipykernel_2420\1536958298.py:3: DtypeWarning: Colum
ns (11) have mixed types. Specify dtype option on import or set low_memory=False.
  train = pd.read_csv('train_final.csv')
```

In [3]:
```
#Setting a variable for all products the bank offers
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']

train['total_products'] = train[products].sum(axis=1)
```

NA values treatment

Double checking for dtypes and null values, but as per last week's code it shouldn't be any

We will deal with the null values we replaced

In [4]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10501007 entries, 0 to 10501006
Data columns (total 46 columns):
 #   Column                Dtype  
---  ------                -----  
 0   date                  object 
 1   customer_code         int64  
 2   employee_index        object 
 3   country               object 
 4   sex_H                 int64  
 5   age                   int64  
 6   first_contract_date   object 
 7   new_cust              int64  
 8   seniority_in_months   int64  
 9   primary_cust          int64  
 10  last_date_primary     object 
 11  cust_type             object 
 12  cust_relationship     object 
 13  residency_spain       int64  
 14  birth_spain           int64  
 15  join_channel          object 
 16  deceased              int64  
 17  province_name         object 
 18  active_cust           int64  
 19  income                float64
 20  segment               object 
 21  savings_acct          int64  
 22  guarantees            int64  
 23  current_acct          int64  
 24  derivada_acct         int64  
 25  payroll_acct          int64  
 26  junior_acct           int64  
 27  mas_particular_acct   int64  
 28  particular_acct       int64  
 29  particular_plus_acct  int64  
 30  short_term_depo       int64  
 31  medium_term_depo      int64  
 32  long_term_depo        int64  
 33  e_acct                int64  
 34  funds                 int64  
 35  mortgage              int64  
 36  pension               int64  
 37  loans                 int64  
 38  taxes                 int64  
 39  credit_card           int64  
 40  securities            int64  
 41  home_acct             int64  
 42  payroll_acct.1        int64  
 43  pensions_2            int64  
 44  direct_debt           int64  
 45  total_products        int64  
dtypes: float64(1), int64(35), object(10)
memory usage: 3.6+ GB
```

In [5]:
```
#Changing dates columns to datetime type

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

In [6]:
```
# Checking unique variables per column and NA/0 values

list_col = list(train.columns)

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

```
date variables: <DatetimeArray>
['2015-06-28 00:00:00', '2015-07-28 00:00:00', '2015-08-28 00:00:00',
 '2015-09-28 00:00:00', '2015-10-28 00:00:00', '2015-11-28 00:00:00',
 '2015-12-28 00:00:00', '2016-01-28 00:00:00', '2016-02-28 00:00:00',
 '2016-03-28 00:00:00', '2016-04-28 00:00:00', '2016-05-28 00:00:00']
Length: 12, dtype: datetime64[ns]
NA values: 0
customer_code variables: [  16132 1063040 1063041 ... 1173729 1164094 1550586]
NA values: 0
employee_index variables: ['N' 'A' 'B' 'F' 'S']
NA values: 0
country variables: ['ES' 'CL' 'NL' 'AT' 'CH' 'CA' 'IE' 'GB' 'AR' 'DE' 'DO' 'BE' 'MX'
'FR'
 'VE' 'QA' 'US' 'HN' 'EC' 'CR' 'CO' 'NI' 'BR' 'PT' 'MZ' 'AL' 'SE' 'IT'
 'PE' 'IN' 'PY' 'MA' 'PL' 'CN' 'FI' 'TW' 'GR' 'AE' 'PR' 'HK' 'RO' 'GT'
 'NO' 'BG' 'GA' 'RU' 'UA' 'SN' 'MR' 'EE' 'SV' 'CZ' 'IL' 'SA' 'CI' 'LU'
 'PA' 'ET' 'CM' 'BA' 'BO' 'HR' 'SG' 'BY' 'NG' 'CU' 'JP' 'SK' 'AU' 'MD'
 'TR' 'KE' 'UY' 'ZA' 'GE' 'DK' 'AD' 'GQ' 'EG' 'DZ' 'TH' 'PK' 'LY' 'TN'
 'TG' 'LB' 'KR' 'KH' 'GH' 'RS' 'KW' 'PH' 'VN' 'AO' 'MM' 'NZ' 'GI' 'LV'
 'SL' 'GN' 'GW' 'CG' 'ML' 'HU' 'MK' 'OM' 'LT' 'IS' 'CD' 'GM' 'KZ' 'CF'
 'BZ' 'ZW' 'DJ' 'JM' 'BM' 'MT']
NA values: 0
sex_H variables: [0 1]
NA values: 0
age variables: [ 48  25  24  26  23  22  29  36  32  30  28  56  27  40  34  63  53
39
  60  42  31  41  45  37  35  57  55  51  58  46  44  50  65  47  75  38
  49  43  52   5  18  13  11  59  33  70  69  61  82  68  54  12  67  14
  71  77  92   6  10   7  84  73  62  95  17  87  15  72  64  21  66  85
  83  16   8  20  86   9  19  79  74  80  96  81  89  90  78  88 100  76
  91  94  93  98   4  97 104 106 101 103  99   3   2 102 107 111 109 105
 110 112 115 108 116 113 126 117 163 127 114 164]
NA values: 0
first_contract_date variables: <DatetimeArray>
['1995-03-08 00:00:00', '2012-09-19 00:00:00', '2013-06-18 00:00:00',
 '2012-10-16 00:00:00', '2013-10-08 00:00:00', '2012-10-05 00:00:00',
 '2013-02-04 00:00:00', '2014-01-31 00:00:00', '2012-09-20 00:00:00',
 '2012-10-17 00:00:00',
 ...
 '2016-05-10 00:00:00', '2016-05-02 00:00:00', '2016-05-14 00:00:00',
 '2016-05-22 00:00:00', '2016-05-11 00:00:00', '2016-04-30 00:00:00',
 '2016-05-27 00:00:00', '2016-05-25 00:00:00', '2016-05-01 00:00:00',
 '2016-05-15 00:00:00']
Length: 6756, dtype: datetime64[ns]
NA values: 0
new_cust variables: [0 1]
NA values: 0
seniority_in_months variables: [    244      34      25      33      22      10
21       9      17
      12      20      30      18       2       5      24       6      27
      19       8      13      32       7      11      28      15      35
      16      23       3      26      31       1       4      29     157
      14      36      40     139      46      38      45      44      43
      41      39      47      42      37      49      50      48      51
      56      54      55       0      53      52      57      58     209
     165     164     105      81     129     109     128     108     156
     121     136     150     142      64     122     125     146     138
     101      69     163     116      96     117     107     137     145
      61     162     160     102      88      65     114     113     161
     217      77     154     152     126     159     166     104     119
```

```
      94     149     103      82      76     151      70      86      79
     135     169      60     118     134     120     110     148      78
     141      66     140      99     147     100      95     133     124
     127     193      80     132      83     123     231     158     143
     187     111      85      98     170     106      84      63     155
     189     175      87     177     115     112     232      97     144
      93     203     131     172     190      72     176     153      89
     174     194      71     173     212      68      59      74     130
      73     183     180      62     216     179     178     168     184
     171     167     198      92     199     206     235     213     208
      75     195     201     186      67     188      90     207     185
     192     182      91     215     211     181     196     219     205
     202     200     214     191     227     218     225     224     226
     242     210     223     237     222     204     233     220     228
     197     221     241     229     240     234     243     230     238
     246     236     239     245 -999999     247     248     249     250
     251     252     253     254     255     256]
NA values: 0
primary_cust variables: [ 1 99]
NA values: 0
last_date_primary variables: ['0' '2015-07-13' '2015-07-29' '2015-07-30' '2015-07-23'
'2015-07-06'
 '2015-07-03' '2015-07-01' '2015-07-21' '2015-07-14' '2015-07-10'
 '2015-07-27' '2015-07-16' '2015-07-15' '2015-07-08' '2015-07-20'
 '2015-07-07' '2015-07-09' '2015-07-02' '2015-07-28' '2015-07-22'
 '2015-07-17' '2015-07-24' '2015-08-21' '2015-08-19' '2015-08-25'
 '2015-08-14' '2015-08-24' '2015-08-17' '2015-08-18' '2015-08-10'
 '2015-08-13' '2015-08-27' '2015-08-03' '2015-08-06' '2015-08-20'
 '2015-08-26' '2015-08-28' '2015-08-05' '2015-08-11' '2015-08-07'
 '2015-08-04' '2015-08-12' '2015-09-17' '2015-09-01' '2015-09-18'
 '2015-09-03' '2015-09-02' '2015-09-14' '2015-09-16' '2015-09-29'
 '2015-09-28' '2015-09-09' '2015-09-22' '2015-09-08' '2015-09-11'
 '2015-09-21' '2015-09-04' '2015-09-25' '2015-09-07' '2015-09-10'
 '2015-09-23' '2015-09-24' '2015-09-15' '2015-10-08' '2015-10-07'
 '2015-10-13' '2015-10-26' '2015-10-29' '2015-10-05' '2015-10-28'
 '2015-10-09' '2015-10-22' '2015-10-20' '2015-10-15' '2015-10-06'
 '2015-10-01' '2015-10-21' '2015-10-16' '2015-10-27' '2015-10-19'
 '2015-10-23' '2015-10-02' '2015-10-14' '2015-11-23' '2015-11-24'
 '2015-11-12' '2015-11-04' '2015-11-13' '2015-11-25' '2015-11-19'
 '2015-11-20' '2015-11-03' '2015-11-16' '2015-11-17' '2015-11-11'
 '2015-11-27' '2015-11-18' '2015-11-10' '2015-11-26' '2015-11-02'
 '2015-11-05' '2015-11-06' '2015-11-09' '2015-12-21' '2015-12-18'
 '2015-12-28' '2015-12-24' '2015-12-04' '2015-12-29' '2015-12-16'
 '2015-12-11' '2015-12-30' '2015-12-15' '2015-12-01' '2015-12-09'
 '2015-12-10' '2015-12-17' '2015-12-02' '2015-12-14' '2015-12-03'
 '2015-12-22' '2015-12-23' '2015-12-07' '2016-01-08' '2016-01-14'
 '2016-01-13' '2016-01-28' '2016-01-05' '2016-01-19' '2016-01-12'
 '2016-01-18' '2016-01-21' '2016-01-22' '2016-01-07' '2016-01-20'
 '2016-01-26' '2016-01-15' '2016-01-27' '2016-01-25' '2016-01-11'
 '2016-01-04' '2016-02-23' '2016-02-19' '2016-02-18' '2016-02-26'
 '2016-02-12' '2016-02-24' '2016-02-09' '2016-02-08' '2016-02-11'
 '2016-02-05' '2016-02-04' '2016-02-03' '2016-02-15' '2016-02-22'
 '2016-02-10' '2016-02-16' '2016-02-01' '2016-02-17' '2016-02-02'
 '2016-02-25' '2016-03-07' '2016-03-29' '2016-03-10' '2016-03-18'
 '2016-03-14' '2016-03-22' '2016-03-08' '2016-03-21' '2016-03-30'
 '2016-03-01' '2016-03-23' '2016-03-02' '2016-03-24' '2016-03-03'
 '2016-03-09' '2016-03-11' '2016-03-04' '2016-03-16' '2016-03-28'
 '2016-03-15' '2016-03-17' '2016-04-22' '2016-04-01' '2016-04-06'
 '2016-04-12' '2016-04-05' '2016-04-15' '2016-04-13' '2016-04-19'
 '2016-04-04' '2016-04-18' '2016-04-26' '2016-04-11' '2016-04-25'
```

```
 '2016-04-27' '2016-04-08' '2016-04-07' '2016-04-21' '2016-04-28'
 '2016-04-20' '2016-04-14' '2016-05-23' '2016-05-05' '2016-05-17'
 '2016-05-19' '2016-05-12' '2016-05-06' '2016-05-03' '2016-05-20'
 '2016-05-02' '2016-05-16' '2016-05-18' '2016-05-04' '2016-05-13'
 '2016-05-24' '2016-05-27' '2016-05-10' '2016-05-30' '2016-05-25'
 '2016-05-11' '2016-05-09' '2016-05-26']
NA values: 0
cust_type variables: [1.0 3.0 2.0 0.0 '1.0' '1' '3' '3.0' '2.0' '4.0' 'P' '4' 4.0 '2'
'0']
NA values: 0
cust_relationship variables: ['A' 'I' 'P' '0' 'R' 'N']
NA values: 0
residency_spain variables: [1 0]
NA values: 0
birth_spain variables: [0 1]
NA values: 0
join_channel variables: ['KAT' 'KHE' 'KHD' 'KFC' 'KFA' 'KHC' 'KAZ' 'KHK' 'KHL' 'KGN'
'RED' 'KHN'
 'KDH' 'KEH' 'KGC' 'KHM' 'KHO' 'KHF' 'KFK' 'KHA' 'KAF' 'K00' '013' 'KAR'
 'KFJ' 'KAG' 'KAA' 'KFF' 'KAI' 'KCC' 'KFG' 'KFP' 'KFD' 'KGX' 'KAH' 'KAE'
 'KFS' 'KAB' 'other' 'KFN' 'KAP' 'KFL' 'KFU' 'KGY' 'KAQ' 'KGV' 'KAJ' 'KAD'
 'KBG' 'KHQ' 'KAK' '007' 'KDR' 'KCA' 'KDT' 'KBO' 'KBQ' 'KAY' 'KCG' 'KBU'
 'KBZ' '004' 'KDO' 'KCK' 'KEC' 'KAC' 'KEU' 'KDE' 'KDY' 'KCH' 'KCI' 'KCL'
 'KDA' 'KES' 'KAS' 'KDX' 'KCM' 'KCN' 'KDQ' 'KCB' 'KDU' 'KAL' 'KAW' 'KEY'
 'KDZ' 'KCS' 'KCD' 'KCE' 'KEJ' 'KDC' 'KBL' 'KAO' 'KEA' 'KEW' 'KFT' 'KEV'
 'KBH' 'KEG' 'KEI' 'KEO' 'KBD' 'KDP' 'KBV' 'KCO' 'KBR' 'KCV' 'KBF' 'KCU'
 'KBX' 'KDD' 'KBW' 'KCF' 'KAN' 'KEZ' 'KAM' 'KDS' 'KBY' 'KEF' 'KBS' 'KDF'
 'KCP' 'KDB' 'KBP' 'KBE' 'KCT' 'KCX' 'KBN' 'KDV' 'KDG' 'KEB' 'KEL' 'KDW'
 'KBB' 'KBJ' 'KDM' 'KFH' 'KBM' 'KEN' 'KFI' 'KEQ' 'KAV' 'KFM' 'KAU' 'KED'
 'KEK' 'KFR' 'KFB' 'KFE' 'KGW' 'KFV' 'KGU' 'KDI' 'KEE' 'KCQ' 'KCR' 'KDN'
 'KEM' 'KCJ' 'KDL' '025' 'KHP' 'KHR' 'KHS']
NA values: 0
deceased variables: [0 1]
NA values: 0
province_name variables: ['MADRID' 'VALENCIA' 'JAEN' 'CADIZ' 'PONTEVEDRA' 'CORUÑA, A'
'ASTURIAS'
 'MALAGA' 'HUELVA' 'SEVILLA' 'ALBACETE' 'CASTELLON' 'CORDOBA' 'ZARAGOZA'
 'LUGO' 'MURCIA' 'BADAJOZ' 'BALEARS, ILLES' 'TOLEDO' 'LEON' 'ALICANTE'
 'LERIDA' 'GRANADA' 'OURENSE' 'AVILA' 'SALAMANCA' 'TERUEL' 'BARCELONA'
 'CANTABRIA' 'CACERES' 'TARRAGONA' 'PALENCIA' 'PALMAS, LAS' 'BURGOS'
 'CIUDAD REAL' 'HUESCA' 'NAVARRA' 'SANTA CRUZ DE TENERIFE' 'GIPUZKOA'
 'MELILLA' 'ALAVA' 'GUADALAJARA' 'RIOJA, LA' 'SORIA' 'ALMERIA' 'GIRONA'
 'VALLADOLID' 'CUENCA' 'ZAMORA' 'BIZKAIA' 'CEUTA' 'SEGOVIA' 'other']
NA values: 0
active_cust variables: [0 1]
NA values: 0
income variables: [160900.95  74693.67  35053.77 ...  63867.66  34341.18  89018.37]
NA values: 0
segment variables: ['02 - PARTICULARES' '03 - UNIVERSITARIO' '01 - TOP' '0']
NA values: 0
savings_acct variables: [0 1]
NA values: 0
guarantees variables: [0 1]
NA values: 0
current_acct variables: [1 0]
NA values: 0
derivada_acct variables: [0 1]
NA values: 0
payroll_acct variables: [0 1]
NA values: 0
```

```
junior_acct variables: [0 1]
NA values: 0
mas_particular_acct variables: [0 1]
NA values: 0
particular_acct variables: [0 1]
NA values: 0
particular_plus_acct variables: [0 1]
NA values: 0
short_term_depo variables: [0 1]
NA values: 0
medium_term_depo variables: [0 1]
NA values: 0
long_term_depo variables: [0 1]
NA values: 0
e_acct variables: [0 1]
NA values: 0
funds variables: [0 1]
NA values: 0
mortgage variables: [0 1]
NA values: 0
pension variables: [0 1]
NA values: 0
loans variables: [0 1]
NA values: 0
taxes variables: [0 1]
NA values: 0
credit_card variables: [0 1]
NA values: 0
securities variables: [0 1]
NA values: 0
home_acct variables: [0 1]
NA values: 0
payroll_acct.1 variables: [0 1]
NA values: 0
pensions_2 variables: [0 1]
NA values: 0
direct_debt variables: [0 1]
NA values: 0
total_products variables: [ 1  2  0  5  4  8  3  6  7  9 10 11 12 13 14 15]
NA values: 0
```

In [7]:

```
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)
```

primary_cust	Count	Percentage	Products Owned
1	10480153	99.8	14104566
99	20854	0.2	1822

last_date_primary	Count	Percentage	Products Owned
0	10480153	99.80	14104566
2015-07-01	142	0.00	40
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	56
2015-07-08	65	0.00	27
2015-07-09	148	0.00	43
2015-07-10	117	0.00	57
2015-07-13	81	0.00	35
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	64
2015-07-20	110	0.00	46
2015-07-21	130	0.00	52
2015-07-22	102	0.00	53
2015-07-23	78	0.00	31
2015-07-24	97	0.00	50
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	4
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	7

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	13
2015-09-29	66	0.00	5
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	7
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	7
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	14
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	12
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	13
2015-11-26	61	0.00	3
2015-11-27	91	0.00	9
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	8
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	7
2015-12-21	206	0.00	9
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	8
2016-01-11	90	0.00	0
2016-01-12	78	0.00	5
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	7
2016-01-28	99	0.00	7
2016-02-01	121	0.00	2
2016-02-02	96	0.00	5
2016-02-03	65	0.00	7
2016-02-04	74	0.00	8
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	8
2016-02-11	96	0.00	13
2016-02-12	93	0.00	14
2016-02-15	129	0.00	4
2016-02-16	91	0.00	4
2016-02-17	73	0.00	12
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	5
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	8
2016-03-11	84	0.00	5
2016-03-14	97	0.00	6
2016-03-15	96	0.00	9
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               7
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               9
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               5
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               5
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              10
2016-05-03               65        0.00               2
2016-05-04               83        0.00               3
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               7
2016-05-11               74        0.00               3
2016-05-12               73        0.00               4
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               9
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        14076218
1            26861        0.26           30170
                      Count  Percentage  Products Owned
seniority_in_months
-999999                  28        0.00             141
 0                   134357        1.28           80993
 1                   132477        1.26          112184
 2                   128049        1.22          115998
```

3	128348	1.22	118312
4	120189	1.14	111627
5	125917	1.20	116627
6	114032	1.09	106406
7	112509	1.07	105279
8	107713	1.03	103799
9	105700	1.01	101279
10	121312	1.16	115621
11	95961	0.91	91254
12	149217	1.42	143273
13	110316	1.05	105824
14	115248	1.10	110365
15	110974	1.06	107026
16	113837	1.08	113368
17	112308	1.07	110690
18	107755	1.03	111330
19	99779	0.95	105055
20	98493	0.94	105623
21	115992	1.10	123374
22	103688	0.99	107434
23	109845	1.05	114899
24	114855	1.09	119496
25	100606	0.96	104684
26	105067	1.00	110324
27	98761	0.94	102877
28	97947	0.93	101149
29	94570	0.90	98273
30	91540	0.87	96884
31	87694	0.84	94617
32	83742	0.80	93203
33	95070	0.91	104483
34	88816	0.85	98199
35	90189	0.86	98482
36	103372	0.98	112133
37	92019	0.88	100275
38	95849	0.91	105140
39	90396	0.86	98611
40	94975	0.90	102778
41	92266	0.88	100486
42	85560	0.81	94134
43	93260	0.89	102707
44	90608	0.86	100033
45	95837	0.91	105226
46	90374	0.86	99141
47	83273	0.79	90442
48	87697	0.84	94288
49	81310	0.77	86089
50	81877	0.78	86144
51	75641	0.72	79449
52	80410	0.77	84066
53	78248	0.75	81212
54	67857	0.65	71149
55	57137	0.54	60771
56	42483	0.40	45744
57	30535	0.29	34268
58	23325	0.22	27559
59	18519	0.18	22742
60	18589	0.18	22843
61	20235	0.19	25971
62	22311	0.21	28798

63	24035	0.23	32031
64	25026	0.24	34244
65	23366	0.22	32584
66	23877	0.23	34175
67	23289	0.22	33959
68	22430	0.21	34834
69	22089	0.21	34561
70	20651	0.20	33025
71	19851	0.19	32422
72	18972	0.18	32194
73	16070	0.15	28701
74	13331	0.13	25025
75	11721	0.11	22139
76	12367	0.12	22734
77	14122	0.13	24447
78	16120	0.15	26710
79	17066	0.16	27285
80	19928	0.19	31706
81	27153	0.26	40705
82	26567	0.25	37621
83	28216	0.27	38209
84	30605	0.29	40502
85	30923	0.29	40343
86	33366	0.32	42131
87	33912	0.32	41508
88	34946	0.33	40451
89	34669	0.33	39318
90	34391	0.33	38536
91	32990	0.31	35121
92	30092	0.29	30800
93	32329	0.31	32767
94	30613	0.29	30903
95	31754	0.30	32155
96	33503	0.32	34231
97	32987	0.31	33933
98	33943	0.32	35136
99	31364	0.30	33701
100	31312	0.30	34260
101	31940	0.30	35296
102	34352	0.33	38831
103	33307	0.32	38964
104	33722	0.32	40980
105	35477	0.34	44487
106	32983	0.31	41397
107	34338	0.33	43611
108	34511	0.33	44722
109	33252	0.32	43696
110	36763	0.35	48706
111	35059	0.33	46737
112	35845	0.34	48332
113	33510	0.32	46563
114	34692	0.33	48872
115	33684	0.32	48393
116	31886	0.30	46465
117	36185	0.34	53695
118	34030	0.32	51314
119	34252	0.33	52363
120	33865	0.32	52556
121	31267	0.30	49255
122	31519	0.30	50576

123	31063	0.30	50756
124	32338	0.31	53161
125	32208	0.31	53699
126	32269	0.31	54345
127	32473	0.31	55047
128	29976	0.29	52059
129	29509	0.28	50869
130	28768	0.27	49923
131	27157	0.26	47367
132	28615	0.27	49596
133	28906	0.28	50200
134	31085	0.30	53905
135	28451	0.27	48961
136	30146	0.29	52556
137	29556	0.28	51166
138	29931	0.29	51959
139	30079	0.29	52539
140	29953	0.29	52616
141	30348	0.29	53134
142	30189	0.29	52733
143	29914	0.28	51984
144	30203	0.29	52624
145	28389	0.27	50404
146	29192	0.28	52788
147	27382	0.26	49568
148	27150	0.26	49914
149	25470	0.24	47709
150	26667	0.25	49988
151	27286	0.26	51436
152	26150	0.25	50574
153	27156	0.26	52871
154	25813	0.25	50322
155	24703	0.24	48161
156	29772	0.28	57701
157	28565	0.27	54692
158	29615	0.28	56793
159	33162	0.32	62923
160	35951	0.34	67269
161	37581	0.36	69386
162	42914	0.41	80188
163	43631	0.42	80632
164	46647	0.44	85313
165	52452	0.50	95272
166	51356	0.49	93760
167	46816	0.45	84930
168	50527	0.48	92732
169	50698	0.48	93202
170	50209	0.48	92287
171	47537	0.45	87739
172	48847	0.47	91901
173	43700	0.42	81824
174	42631	0.41	81537
175	38755	0.37	75387
176	35140	0.33	70617
177	35750	0.34	73555
178	34457	0.33	71025
179	32634	0.31	67657
180	33582	0.32	70709
181	28600	0.27	61124
182	28840	0.27	62804

183	25310	0.24	56819
184	26273	0.25	59714
185	25488	0.24	58240
186	24352	0.23	55643
187	23605	0.22	53602
188	22642	0.22	51219
189	22337	0.21	51109
190	20292	0.19	46347
191	19287	0.18	44089
192	19366	0.18	44089
193	19869	0.19	46184
194	20261	0.19	47054
195	18705	0.18	44335
196	17680	0.17	42535
197	16993	0.16	41430
198	17586	0.17	42802
199	18342	0.17	45400
200	17627	0.17	43531
201	18539	0.18	45734
202	17789	0.17	43727
203	16843	0.16	41070
204	16329	0.16	39679
205	16346	0.16	39781
206	17183	0.16	42092
207	16146	0.15	39648
208	17219	0.16	41836
209	16647	0.16	41060
210	15686	0.15	37808
211	16630	0.16	40076
212	15878	0.15	38115
213	15637	0.15	37064
214	15534	0.15	37088
215	15098	0.14	35866
216	14908	0.14	35191
217	14439	0.14	33782
218	13876	0.13	32043
219	12341	0.12	28497
220	12460	0.12	28929
221	11783	0.11	27129
222	10694	0.10	24140
223	10979	0.10	24813
224	10241	0.10	22613
225	10871	0.10	23662
226	9914	0.09	21532
227	9295	0.09	20193
228	9523	0.09	20901
229	9118	0.09	19692
230	8979	0.09	19198
231	9218	0.09	19704
232	8585	0.08	18272
233	7646	0.07	16116
234	7839	0.07	16775
235	7999	0.08	16990
236	7135	0.07	15133
237	8632	0.08	18266
238	8271	0.08	17592
239	7329	0.07	15569
240	7490	0.07	16290
241	7302	0.07	16096
242	6467	0.06	14257

243	5919	0.06	13080
244	5553	0.05	12484
245	4618	0.04	10525
246	4170	0.04	9828
247	3516	0.03	8332
248	2271	0.02	5617
249	1777	0.02	4640
250	1512	0.01	3989
251	1071	0.01	2901
252	676	0.01	2000
253	416	0.00	1364
254	261	0.00	904
255	179	0.00	680
256	102	0.00	421

Analyzing columns: Last Date Primary and Primary Customer

In [8]:
```
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
primary_cust
1     10480153
99       20854
Name: count, dtype: int64
```

In [9]:
```
non_primary = train[train['primary_cust'] == 99]
non_primary['total_products'].sum()
```

Out[9]:
```
1822
```

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 [10]:
```
train.groupby('deceased')[products].sum()
```

Out[10]:

deceased	savings_acct	guarantees	current_acct	derivada_acct	payroll_acct	junior_acct	mas_particul
0	958	214	6488365	3821	539367	90875	
1	0	0	12798	12	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 [11]:
```
(train['income'] == 0).sum()
```

Out[11]:
```
2238903
```

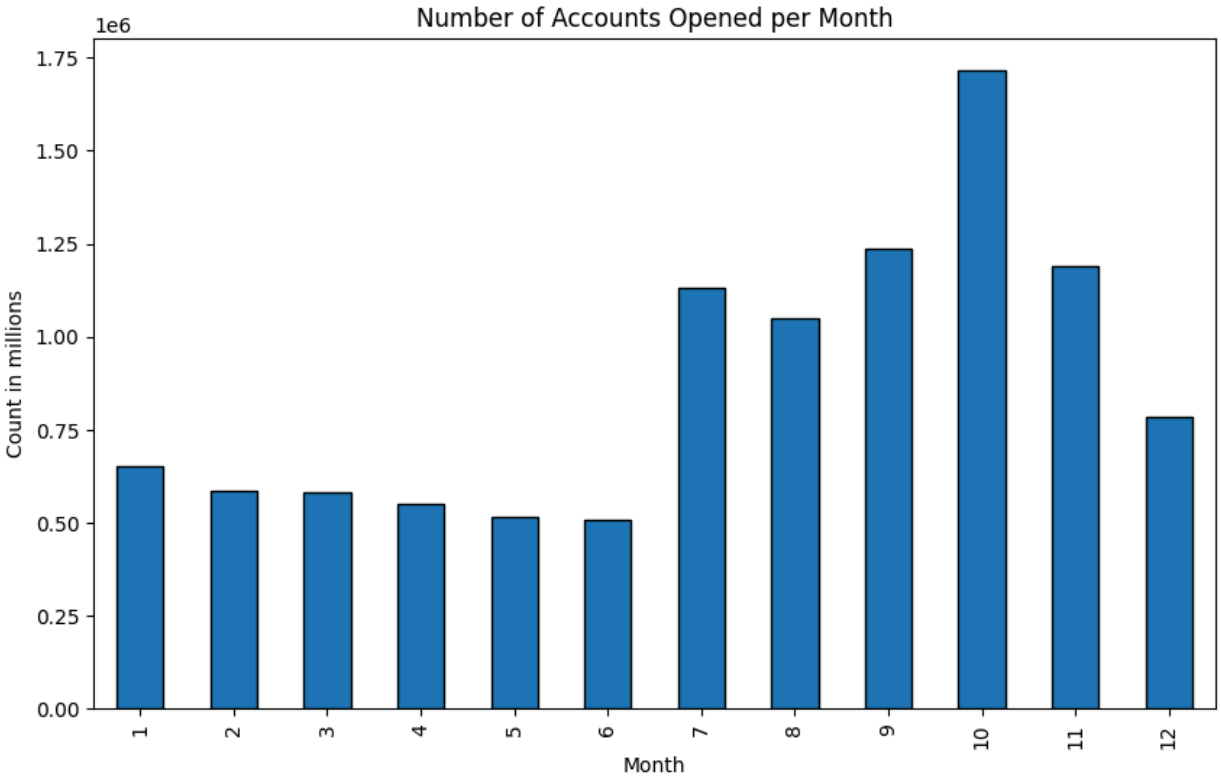
Analyzing columns: Seniority in Months and First Contract Date
We had the impression these two columns were giving us the same information and we decided to check the correlation. Turns out it is highly correlated, so we'll keep the column seniority in months.
Before we delete the first contract date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

In [12]:
```
correlation = train[['seniority_in_months', 'first_contract_date']].corr()
print(correlation)
```

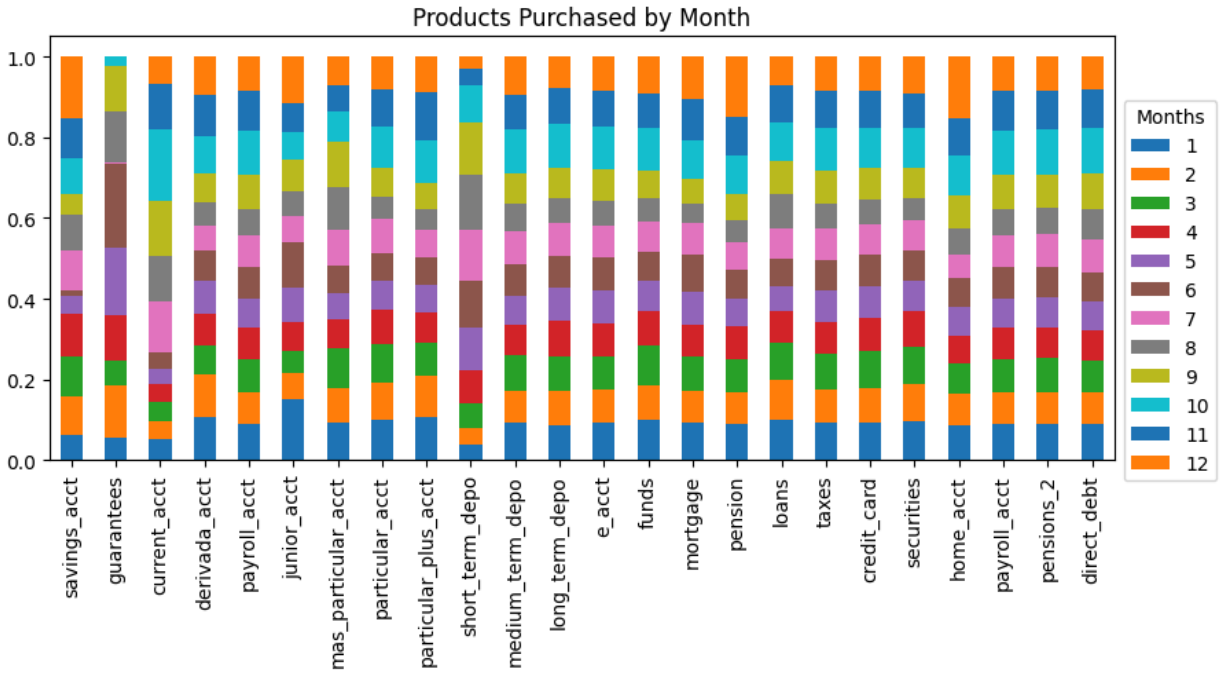
	seniority_in_months	first_contract_date
seniority_in_months	1.00000	-0.03798
first_contract_date	-0.03798	1.00000

In [13]:
```
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[13]:
```
Text(0, 0.5, 'Count in millions')
```



In [14]:
```
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

In [15]:
```
train = train.drop(columns=['primary_cust'])
train = train.drop(columns=['first_contract_date'])

train = train[train['deceased'] != 1]
train = train.drop(columns=['deceased'])

train = train[train['province_name'] != 'other']
train = train[train['seniority_in_months'] != -999999]

train = train[(train['age'] > 0) & (train['age'] <= 100)]
```

In [16]:
```
train = train[train['income'] != 0]
```

Splitting the dataset into train and validation

We will use the 80/20 ratio

In [17]:
```
train, val_set = train_test_split(train, test_size=0.2, random_state=42)

# Check the shapes
print(f"Training set shape: {train.shape}")
print(f"Validation set shape: {val_set.shape}")
```

```
Training set shape: (6585426, 43)
Validation set shape: (1646357, 43)
```

In [18]:
```
train.to_csv('train_cleaned.csv')
val_set.to_csv('val_set.csv')
```

EDA

In [19]:
```
train.describe().round(2)
```

Out[19]:

	date	customer_code	sex_H	age	new_cust	seniority_in_months	re
count	6585426	6585426.00	6585426.00	6585426.00	6585426.00	6585426.00	
mean	2015-12-16 17:06:37.483170048	812506.38	0.45	40.68	0.03	83.14	
min	2015-06-28 00:00:00	15889.00	0.00	2.00	0.00	0.00	
25%	2015-09-28 00:00:00	437619.00	0.00	25.00	0.00	26.00	
50%	2015-12-28 00:00:00	909502.50	0.00	40.00	0.00	54.00	
75%	2016-03-28 00:00:00	1180925.00	1.00	51.00	0.00	139.00	
max	2016-05-28 00:00:00	1454620.00	1.00	100.00	1.00	256.00	
std	NaN	423752.09	0.50	17.08	0.18	66.29	

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

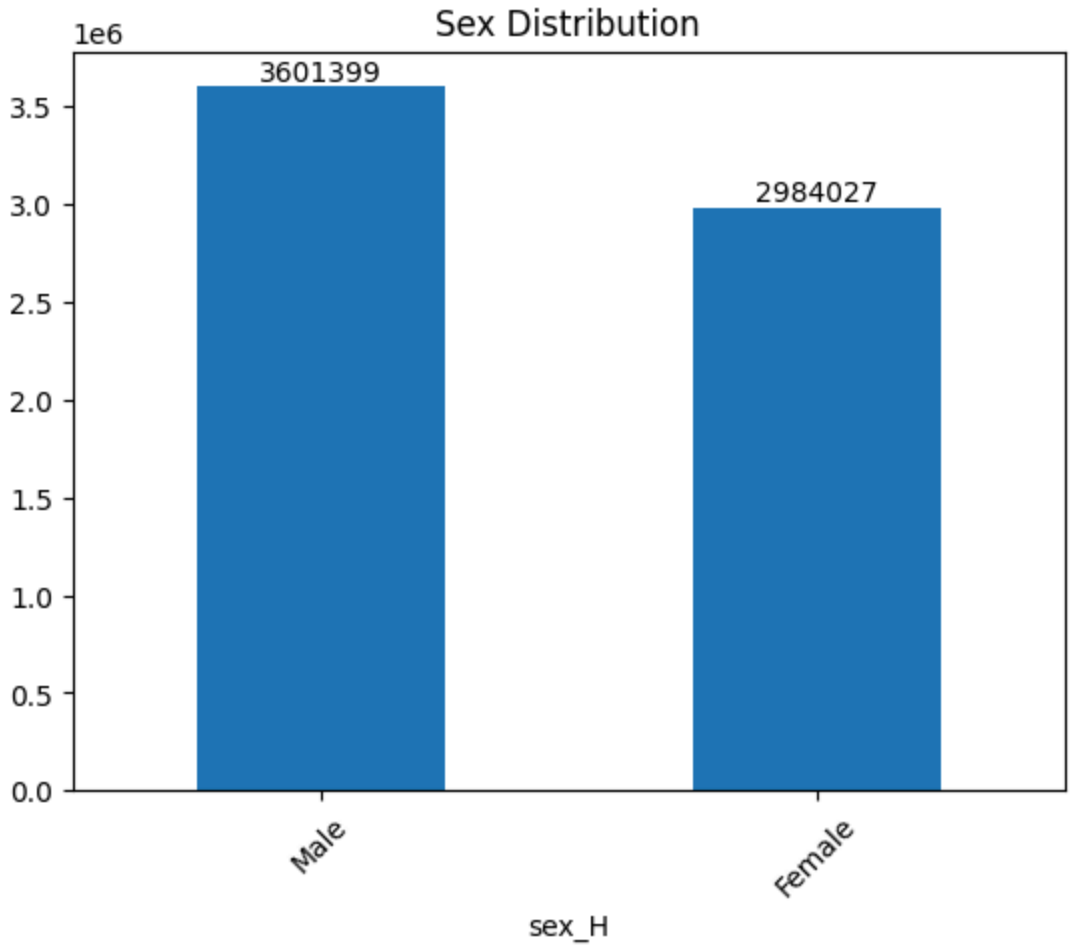
In [20]:
```
pd.set_option('display.max_rows', None)

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

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
employee_index
A                  1332        0.02            8112
B                  1875        0.03            6643
F                  1360        0.02            5102
N                6580848       99.93         9248149
S                     11        0.00              88
           Count  Percentage  Products Owned
country
BO            12         0.0              12
DE            10         0.0              10
ES       6585381       100.0         9268045
IT            18         0.0              27
PY             5         0.0               0
                   Count  Percentage  Products Owned
residency_spain
0                     45         0.0              49
1                6585381       100.0         9268045
               Count  Percentage  Products Owned
birth_spain
0            6293897       95.57         8928708
1             291529        4.43          339386
```

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

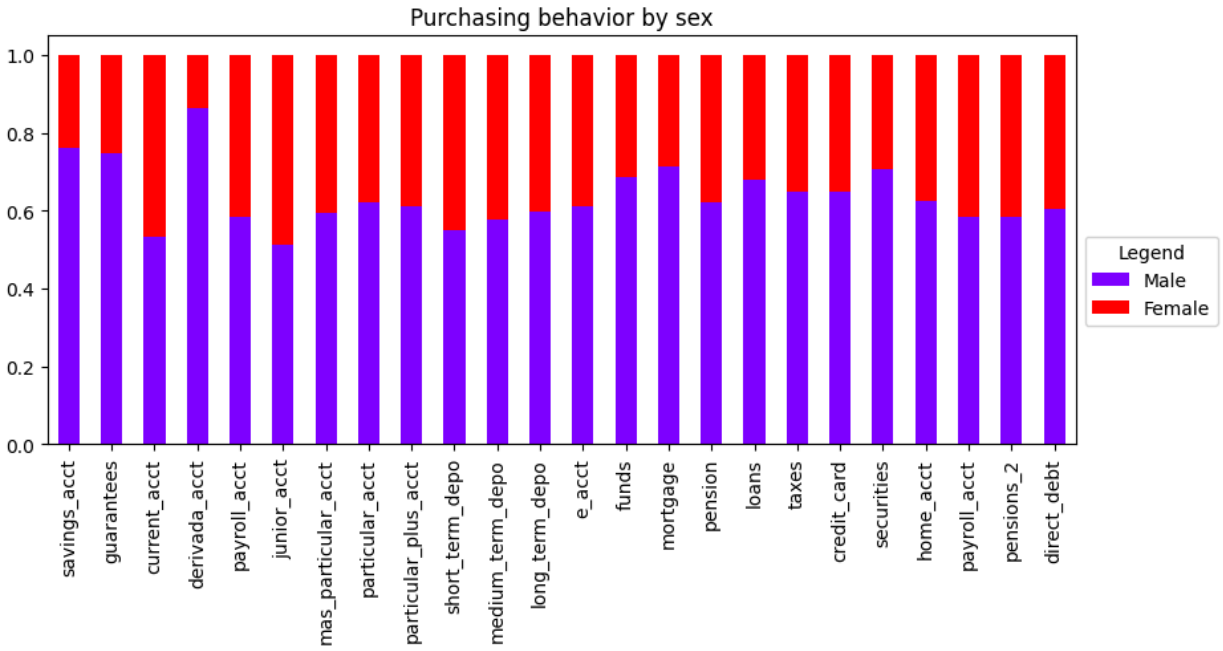
Analyzing column: Sex

In [21]:
```
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()
```

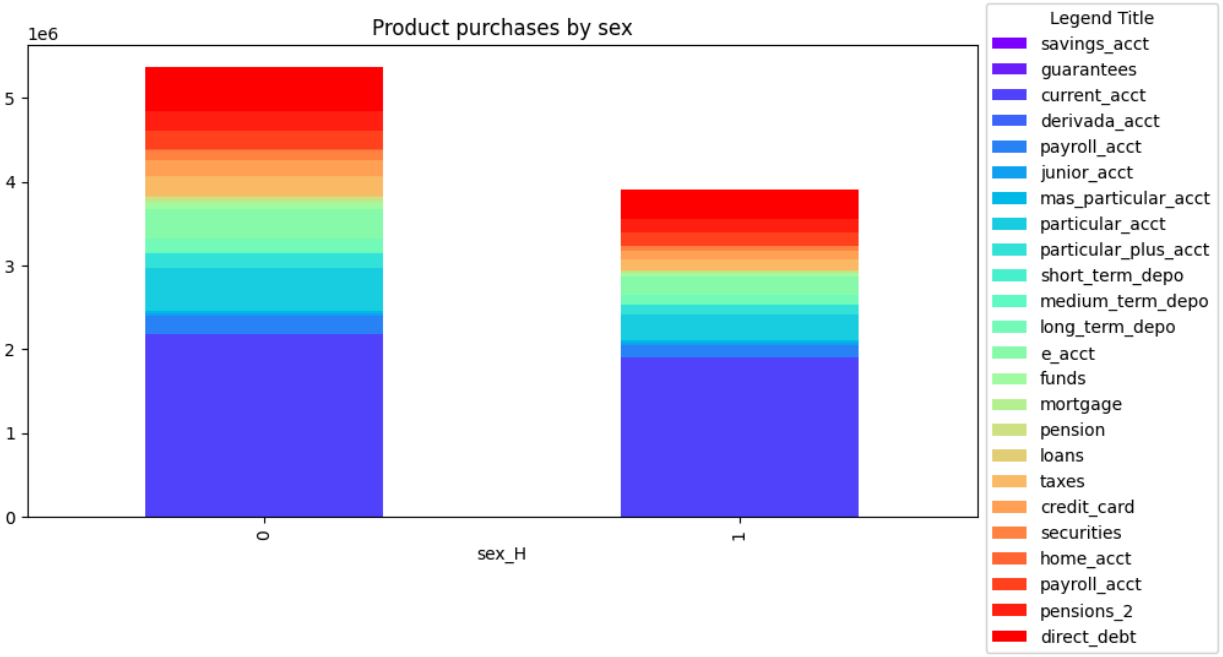


In [22]:
```
# Defining function to plot column data and see product purchasing behavior
def plot_grouped_data(df, group_col, products, colormap='rainbow', title='Grouped Data
    dummy = df.groupby(group_col)[products].sum()
    dummy = (dummy/dummy.sum()).T
    ax = dummy.plot(kind='bar', stacked=True, 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 [23]:
```
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 [24]:
```
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 [25]:
```
dummy
```

Out[25]:

sex_H	savings_acct	guarantees	current_acct	derivada_acct	payroll_acct	junior_acct	mas_particular_a
0	524	113	2172258	2229	216873	31998	32
1	163	38	1895462	348	155275	30389	21

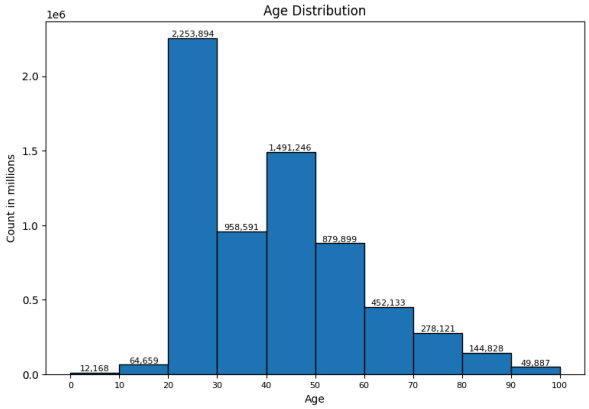
- We can see that the bank has more male customers and they have bought more products than female customers
- Male customers have bought most of the products, showing a skewed distribution in the total products bought
- All the products, except derivada account show a relative similar ratio to the amount 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

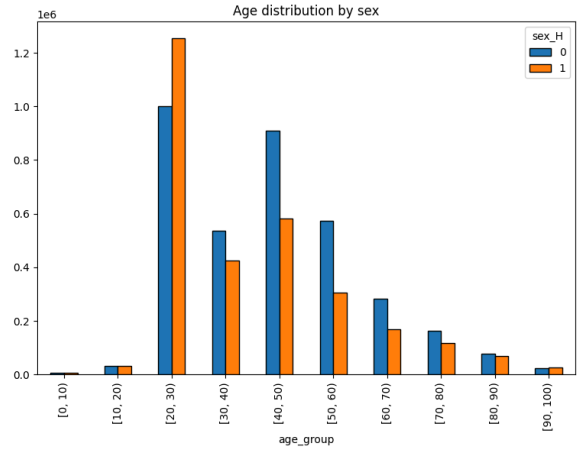
Analyzing column: Age

In [26]:
```
age_sex = train.groupby('sex_H')['age'].size()
```

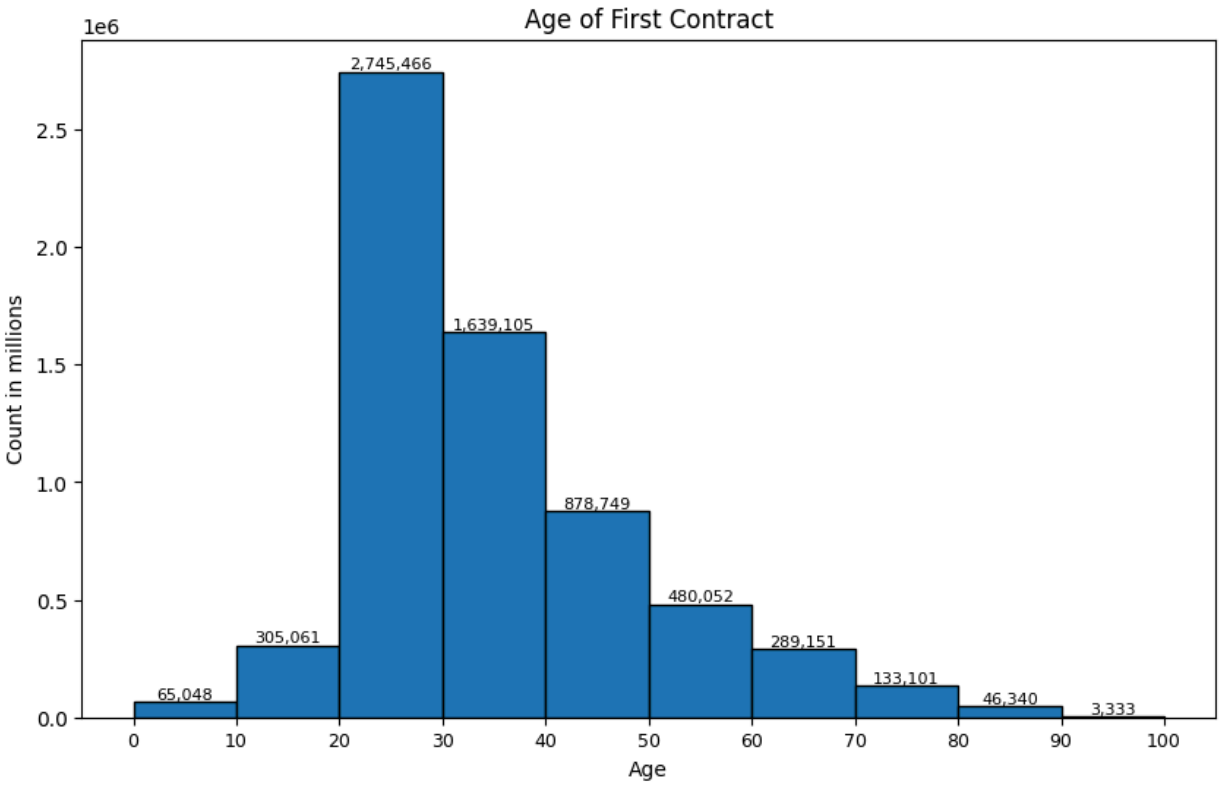
In [27]:
```
bin_edges = np.arange(0, 101, 10)
labels = [f'{i}-{i+19}' for i in bin_edges[:-1]]

# 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_2420\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 [28]:
```
train['first_contract_age'] = train['age']-(train['seniority_in_months']/12).round()

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()
```

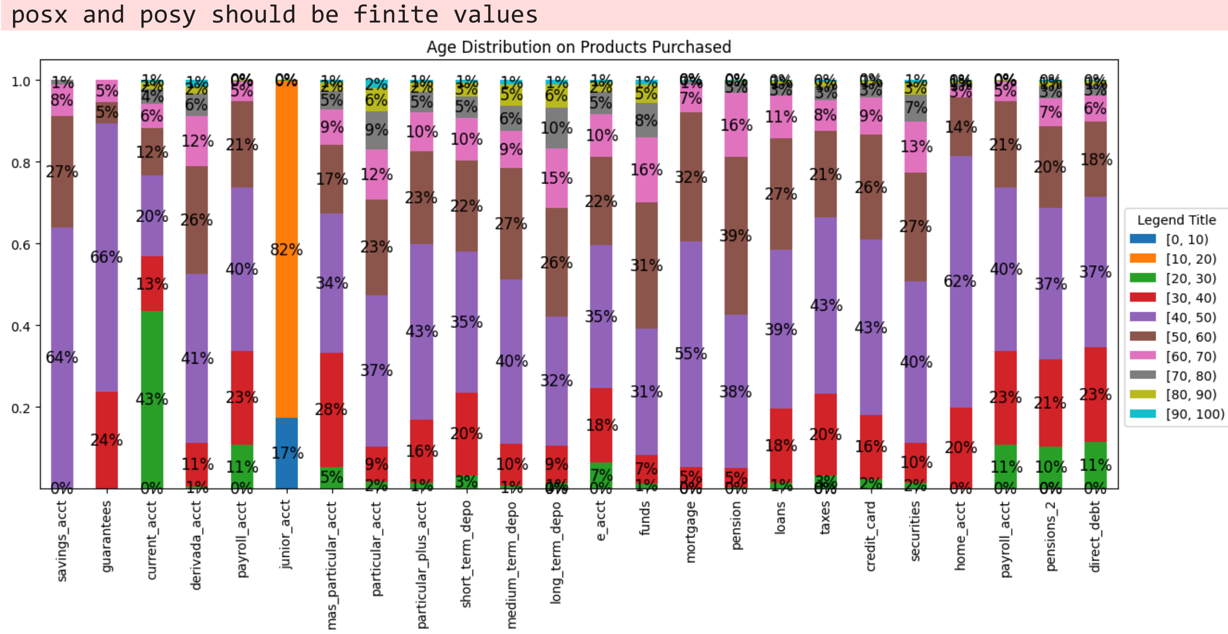


In [29]:
```
dummy = train.groupby('age_group')[products].sum()
dummy = (dummy/dummy.sum()).T
ax = dummy.plot(kind='bar', stacked=True, figsize=(15,6))
plt.legend(loc='center left', title='Legend Title', bbox_to_anchor=(1, .4))
plt.title('Age Distribution on Products Purchased')
```

```
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_2420\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
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 [30]:
```
dummy*100
```

Out[30]:

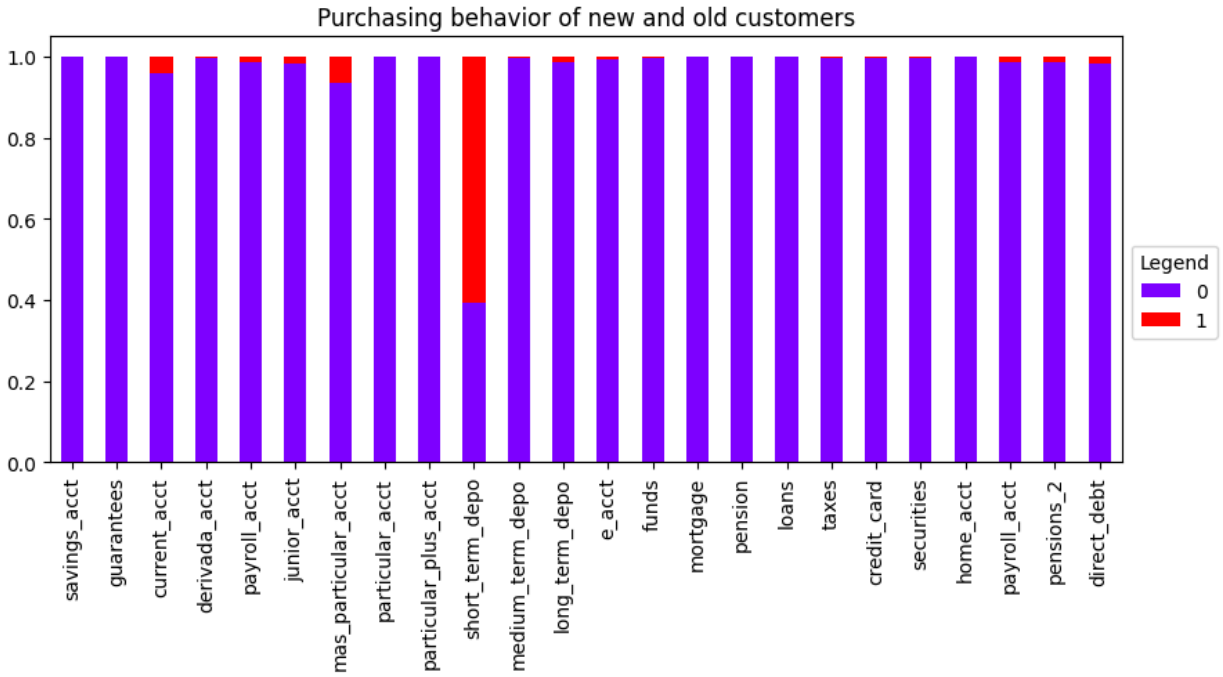
age_group	[0, 10)	[10, 20)	[20, 30)	[30, 40)	[40, 50)	[50, 60)	[60, 70)	[7(
savings_acct	0.000000	0.000000	0.000000	0.145560	63.755459	27.365357	7.714702	1.0
guarantees	0.000000	0.000000	0.000000	23.841060	65.562914	5.298013	5.298013	0.0(
current_acct	0.000270	0.136605	43.306401	13.436175	19.704963	11.637991	5.932956	3.5
derivada_acct	0.000000	0.000000	0.504463	10.787738	41.404734	26.154443	12.378735	5.5
payroll_acct	0.000000	0.012629	10.804841	22.998377	39.886013	21.118748	4.830874	0.2(
junior_acct	17.328931	82.472310	0.182730	0.016029	0.000000	0.000000	0.000000	0.0(
mas_particular_acct	0.000000	0.000000	5.418674	27.914660	34.084491	16.841076	8.620785	4.5(
particular_acct	0.000000	0.000000	1.736690	8.622410	37.027076	23.414315	12.245682	9.3
particular_plus_acct	0.000000	0.000000	1.369650	15.600912	42.812701	22.856516	9.579762	4.9
short_term_depo	0.000000	0.000000	3.344082	20.088379	34.623194	22.190374	10.414427	5.4
medium_term_depo	0.000000	0.000000	0.764647	10.218471	40.327706	27.140020	9.056604	6.2
long_term_depo	0.003196	0.024500	1.424539	9.126351	31.612193	26.423917	14.675378	10.0
e_acct	0.000000	0.001247	6.515015	18.042415	34.998905	21.596691	10.458393	5.4
funds	0.000000	0.000000	1.300137	7.027943	30.886435	30.936377	15.826790	8.3
mortgage	0.025626	0.000000	0.028189	5.335315	55.129026	31.542936	6.875432	1.0
pension	0.000000	0.000000	0.312480	4.692054	37.517000	38.653585	15.714656	2.7
loans	0.000000	0.000000	1.349631	18.340973	38.700025	27.298192	10.555131	2.8
taxes	0.004384	0.031782	3.196861	20.127513	43.048580	21.104545	7.678166	3.2
credit_card	0.000000	0.000000	2.455730	15.554743	42.955324	25.675368	9.153754	3.4
securities	0.000000	0.000000	1.518865	9.667670	39.603536	26.529872	12.622705	6.5
home_acct	0.000000	0.000000	0.085205	19.756003	61.665376	14.182804	2.811774	0.7
payroll_acct	0.000000	0.012629	10.804841	22.998377	39.886013	21.118748	4.830874	0.2(
pensions_2	0.006394	0.051641	10.270673	21.244648	37.151085	20.038706	6.813180	2.9
direct_debt	0.000000	0.000583	11.407592	23.138093	36.819093	18.481505	6.378104	2.5(

- 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 [31]:
```
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    6368320
1     217106
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 [32]:
```
print(train['seniority_in_months'].value_counts().sort_index())

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

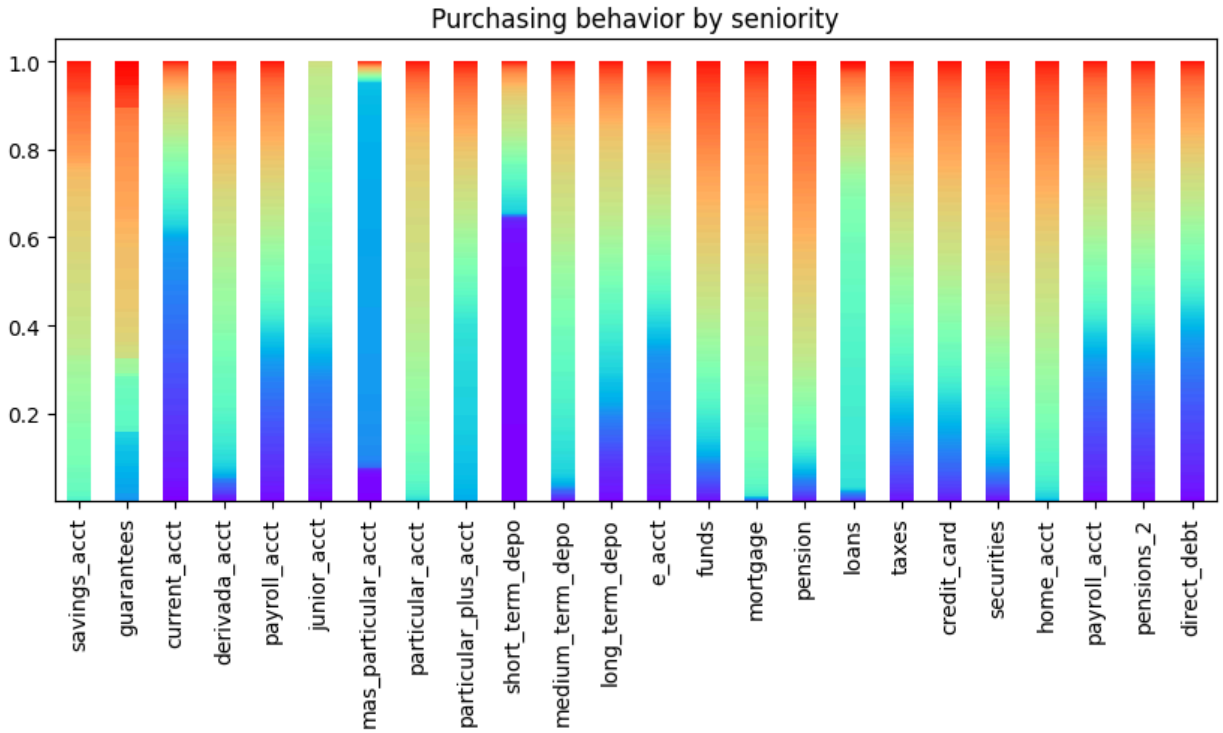
	seniority_in_months
0	27449
1	33013
2	34634
3	36757
4	37257
5	43738
6	41338
7	48320
8	62647
9	69713
10	79474
11	63051
12	97607
13	72213
14	75454
15	72552
16	74903
17	73913
18	70665
19	65593
20	64672
21	76152
22	67885
23	71651
24	74963
25	65815
26	68901
27	64776
28	64193
29	61969
30	59988
31	57310
32	54736
33	62712
34	58457
35	59613
36	68173
37	60739
38	63232
39	59855
40	62900
41	60929
42	56550
43	61565
44	59926
45	63447
46	59611
47	55124
48	57793
49	53531
50	54128
51	50015
52	53277
53	51749
54	45031
55	37716
56	28051
57	20313
58	15714

59	12488
60	12486
61	13549
62	15003
63	16144
64	16827
65	15668
66	16046
67	15616
68	14913
69	14742
70	13847
71	13260
72	12714
73	10694
74	8992
75	7961
76	8352
77	9399
78	10693
79	11533
80	13407
81	18320
82	17730
83	18864
84	20470
85	19863
86	21150
87	20721
88	21329
89	21228
90	21243
91	19951
92	18154
93	19597
94	18599
95	19192
96	20834
97	20966
98	22106
99	20754
100	20800
101	21224
102	22823
103	22294
104	22599
105	23852
106	22120
107	22830
108	23302
109	22557
110	24757
111	23593
112	23988
113	22545
114	23424
115	22644
116	21305
117	24296
118	22768

```
119    22884
120    22785
121    21018
122    21310
123    20871
124    21771
125    21668
126    21656
127    21795
128    20157
129    19945
130    19388
131    18293
132    19341
133    19407
134    20875
135    18946
136    20191
137    19667
138    20070
139    20183
140    19962
141    20310
142    20132
143    19878
144    19980
145    18838
146    19302
147    18176
148    17915
149    16728
150    17608
151    18047
152    17163
153    17986
154    16982
155    16325
156    19728
157    18855
158    19750
159    21994
160    23755
161    24873
162    28559
163    29069
164    31107
165    35147
166    34235
167    31075
168    33631
169    33701
170    33555
171    31626
172    32781
173    29151
174    28463
175    25883
176    23555
177    23871
178    23188
```

179	21971
180	22704
181	19369
182	19612
183	17178
184	17970
185	17312
186	16512
187	16046
188	15236
189	15118
190	13702
191	12931
192	13071
193	13439
194	13803
195	12652
196	11973
197	11552
198	11934
199	12558
200	12016
201	12675
202	12163
203	11481
204	11136
205	11127
206	11731
207	11073
208	11695
209	11226
210	10674
211	11275
212	10830
213	10622
214	10626
215	10203
216	10183
217	9811
218	9361
219	8323
220	8435
221	7945
222	7252
223	7458
224	6988
225	7437
226	6747
227	6321
228	6481
229	6214
230	6174
231	6224
232	5827
233	5194
234	5358
235	5430
236	4826
237	5861
238	5556

```
239     4936
240     5067
241     4875
242     4370
243     3968
244     3781
245     3102
246     2850
247     2426
248     1582
249     1219
250     1063
251      724
252      471
253      301
254      190
255      137
256       78
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 [33]:
```
train['cust_type'] = train['cust_type'].astype(str).str.strip()
train['cust_type'].unique()
```

Out[33]:
```
array(['1.0', '1', '0', '3.0', 'P', '3', '2', '2.0', '4', '0.0', '4.0'],
      dtype=object)
```

In [34]:
```
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)

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

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

```
cust_type
1    6558782
0      25543
3        729
2        193
P        136
4         43
Name: count, dtype: int64
cust_relationship
I    3656172
A    2902803
0      25543
P        772
R        136
Name: count, dtype: int64
```

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

In [35]:
```
na_rel = train[train['cust_type'] == '0']
na_rel[products].sum().sort_values(ascending=False)
```

Out[35]:
```
current_acct            18516
short_term_depo           261
mas_particular_acct       130
direct_debt               113
pensions_2                 67
payroll_acct               67
payroll_acct               67
long_term_depo             62
junior_acct                55
e_acct                      8
funds                       2
securities                  2
medium_term_depo            0
particular_plus_acct        0
guarantees                  0
mortgage                    0
pension                     0
loans                       0
taxes                       0
credit_card                 0
home_acct                   0
particular_acct             0
derivada_acct               0
savings_acct                0
dtype: int64
```

In [36]:
```
train['total_products'].groupby(train['cust_type']).sum()
```

Out[36]:
```
cust_type
0      19350
1    9247935
2        125
3        532
4         42
P        110
Name: total_products, dtype: int64
```

In [37]:
```
dummy = train.groupby('cust_type')[products].sum()
dummy = (dummy/dummy.sum()).T
# dummy.plot(kind='bar',stacked=True, colormap='rainbow',figsize=(20,10))
# plt.legend(loc='center left', title='Legend Title', bbox_to_anchor=(1, .4))
# plt.title('Number of purchases in each month')
# plt.show()

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

cust_type	0	1	2	3	4	P
savings_acct	0.00	100.00	0.00	0.00	0.0	0.00
guarantees	0.00	100.00	0.00	0.00	0.0	0.00
current_acct	0.46	99.53	0.00	0.01	0.0	0.00
derivada_acct	0.00	100.00	0.00	0.00	0.0	0.00
payroll_acct	0.02	99.98	0.00	0.00	0.0	0.00
junior_acct	0.09	99.91	0.00	0.00	0.0	0.00
mas_particular_acct	0.24	99.55	0.02	0.16	0.0	0.03
particular_acct	0.00	100.00	0.00	0.00	0.0	0.00
particular_plus_acct	0.00	100.00	0.00	0.00	0.0	0.00
short_term_depo	3.12	96.07	0.08	0.50	0.1	0.13
medium_term_depo	0.00	100.00	0.00	0.00	0.0	0.00
long_term_depo	0.02	99.96	0.01	0.01	0.0	0.00
e_acct	0.00	100.00	0.00	0.00	0.0	0.00
funds	0.00	100.00	0.00	0.00	0.0	0.00
mortgage	0.00	100.00	0.00	0.00	0.0	0.00
pension	0.00	100.00	0.00	0.00	0.0	0.00
loans	0.00	100.00	0.00	0.00	0.0	0.00
taxes	0.00	100.00	0.00	0.00	0.0	0.00
credit_card	0.00	100.00	0.00	0.00	0.0	0.00
securities	0.00	100.00	0.00	0.00	0.0	0.00
home_acct	0.00	100.00	0.00	0.00	0.0	0.00
payroll_acct	0.02	99.98	0.00	0.00	0.0	0.00
pensions_2	0.02	99.98	0.00	0.00	0.0	0.00
direct_debt	0.01	99.98	0.00	0.00	0.0	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 [38]:
```
train['join_channel'].value_counts()
```

Out[38]:

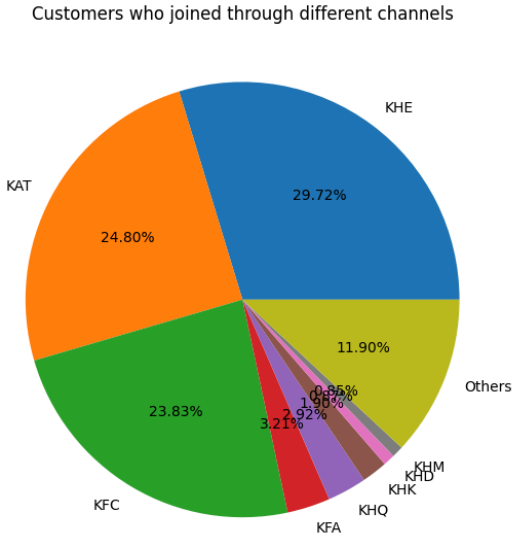
join_channel	
KHE	1956912
KAT	1633256
KFC	1569107
KFA	211098
KHQ	192431
KHK	125414
KHD	57530
KHM	56157
KAS	45597
KHN	44109
other	42224
KAG	40564
RED	36243
KAA	34290
KAY	33643
KAB	33141
KAE	26586
KCC	26390
KBZ	25000
KFD	24848
KHL	23727
KAR	17891
KAW	17828
KAZ	16832
KEY	16556
007	16190
KAF	15776
KCI	14727
KAJ	14113
KCH	13475
013	13475
KAH	12716
KHF	11290
KAQ	10011
KHC	8893
KAP	8032
KAM	5941
KGX	5394
KAD	5317
KFP	5221
KEJ	5140
KFT	4726
KGV	4719
KAL	4217
KDR	4195
KBO	3960
KBH	3924
KFG	3641
KAO	3638
KFJ	3607
KFS	3556
KHO	3355
KCB	2946
KCG	2933
KES	2877
KFF	2855
KEN	2730
KEW	2608
KFU	2579

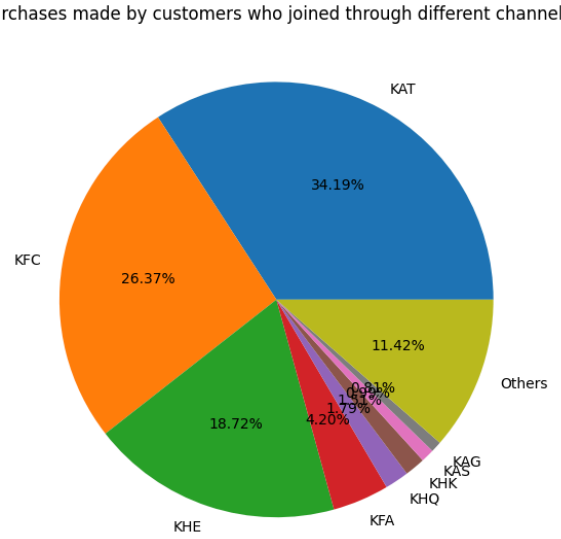
KFN	2388
KCL	2313
KBQ	2158
KBF	2149
KFK	2068
KCD	1893
KCM	1782
KBU	1725
KAI	1678
KDU	1485
KFH	1458
KEZ	1240
KEL	1219
KDM	1195
KDS	1024
KEG	951
KBG	946
KBR	917
KDO	909
KCA	876
KDC	764
KEH	710
KDT	695
KBB	685
KCN	655
KBW	638
KAN	636
KGW	611
KDQ	577
KCU	559
KDP	556
KEI	506
KBV	500
KCK	498
KDE	489
KFI	475
KEO	474
KAK	446
KBS	424
KBY	401
KEA	398
KDF	376
KBL	337
KEK	295
KBM	295
KHP	294
KBJ	280
KED	279
KDD	263
KAC	255
KEV	253
KCF	214
KDG	213
KDA	201
KFM	197
KDV	189
KCE	185
KEB	161
KEF	158
KCV	154

```
KEU          145
KBD          140
KEC          129
KGY          126
KDX          121
KCJ          113
KAU          107
KCP           96
KFE           96
KCR           94
004           92
KDH           91
KCQ           90
KDN           90
KCS           89
KCO           88
KBE           85
KEQ           84
KDY           83
K00           71
KAV           63
KBX           48
KFB           47
KCT           46
KBP           39
KBN           36
KCX           36
KDW           35
KDZ           35
KFV           35
KHA           32
KFL           28
KGU           21
KGC           20
KEM           18
KDI           10
KEE            9
KDB            9
KDL            8
Name: count, dtype: int64
```

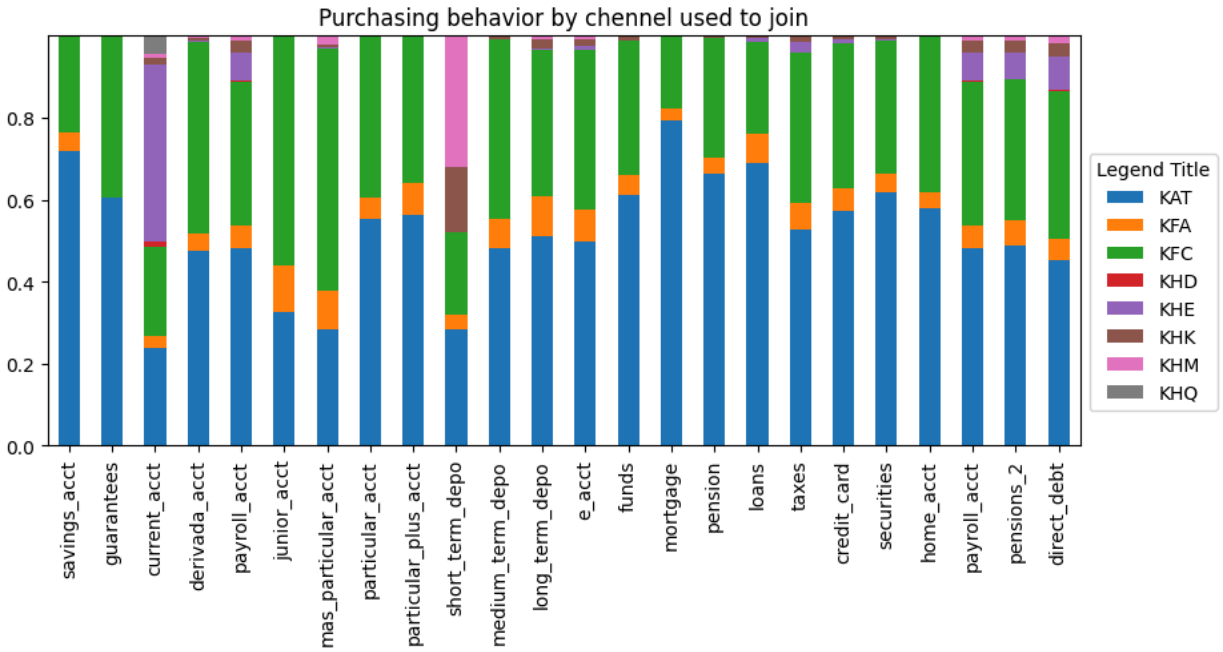
In [39]:

```
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 [40]:
```
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()
```



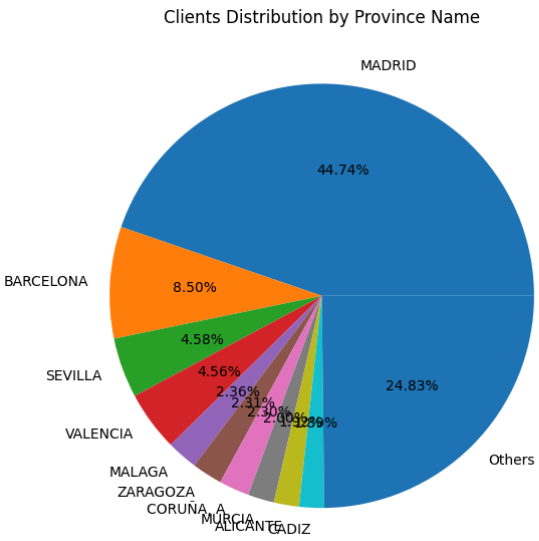
- The 8 top channels used to join the bank make up 88% of the total customers
- When we compare the number of purchases made by clients with to the channel they joined, 7 out of 8 channels are repeated on the number of clients who joined and number of purchases made, with the exception of "others" which 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

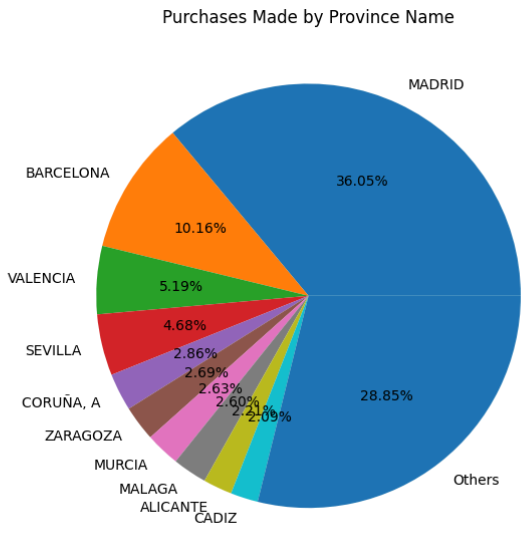
Analyzing column: Province Name

In [41]:
```
train['province_name'].value_counts().head(10)
```

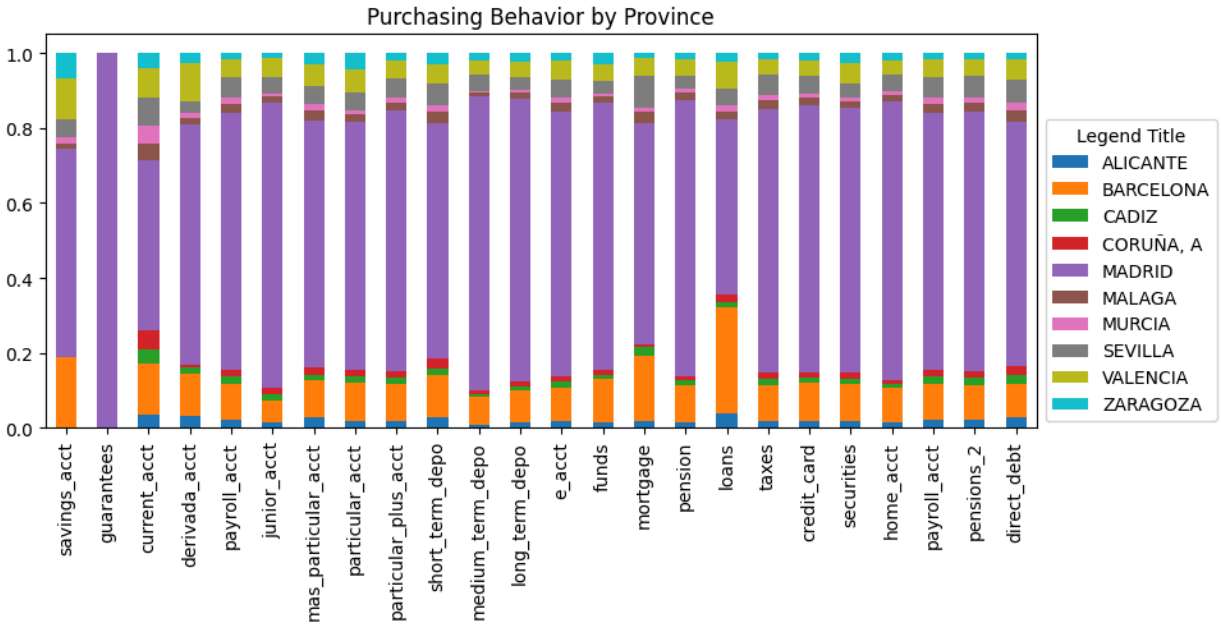
Out[41]:
```
province_name
MADRID        2374175
BARCELONA      668909
VALENCIA       341769
SEVILLA        307924
CORUÑA, A      188185
ZARAGOZA       176829
MURCIA         172920
MALAGA         171454
ALICANTE       145271
CADIZ          137873
Name: count, dtype: int64
```

In [42]:
```
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 [43]:
```
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()
```

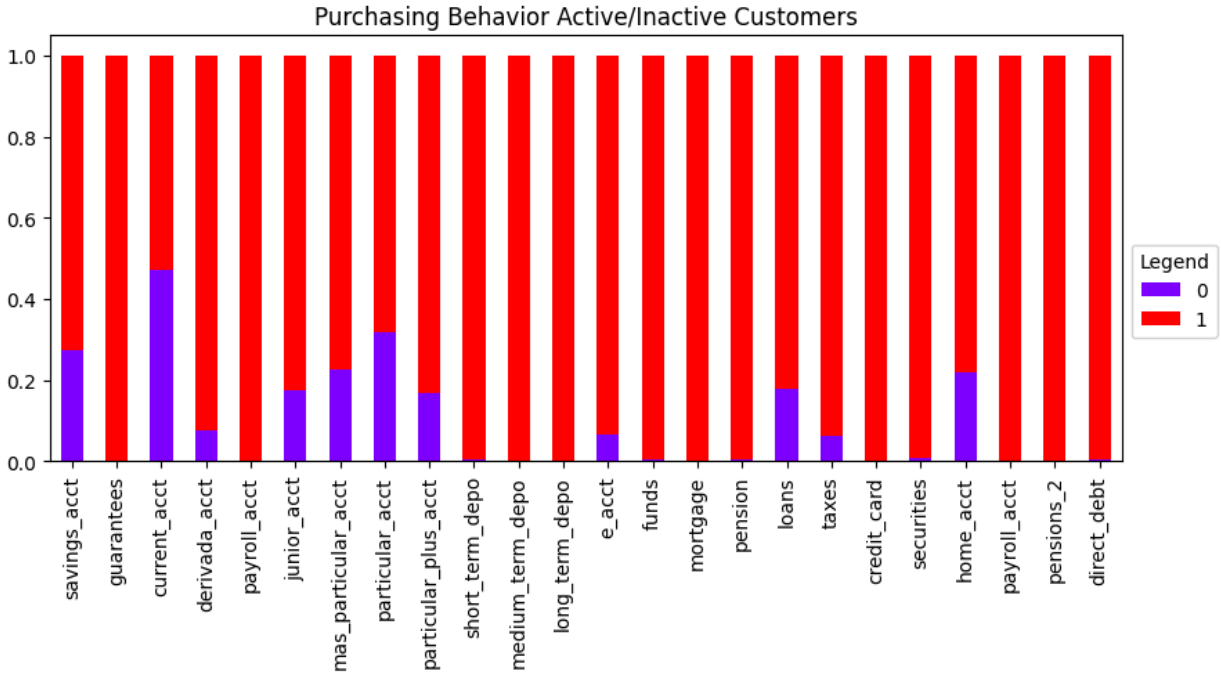


- We can see most of the clients are from Madrid, Spain's capital, followed by other cities like Barcelona, Valencia and Sevilla
- The distribution of number products bought is quite similar to the number of clients from each city
- Guarantees account have only Madrid clients
- Clients from Barcelona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

Analyzing column: Active Customer

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

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

```
active_cust
0    3663219
1    2922207
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 [45]:
```
(train['income'] == 0).sum()
```

Out[45]:
```
0
```

In [46]:
```
train.groupby('province_name')['income'].describe().round()
```

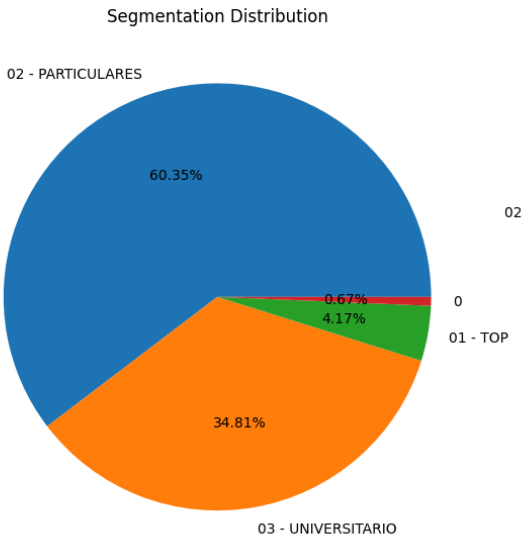
Out[46]:

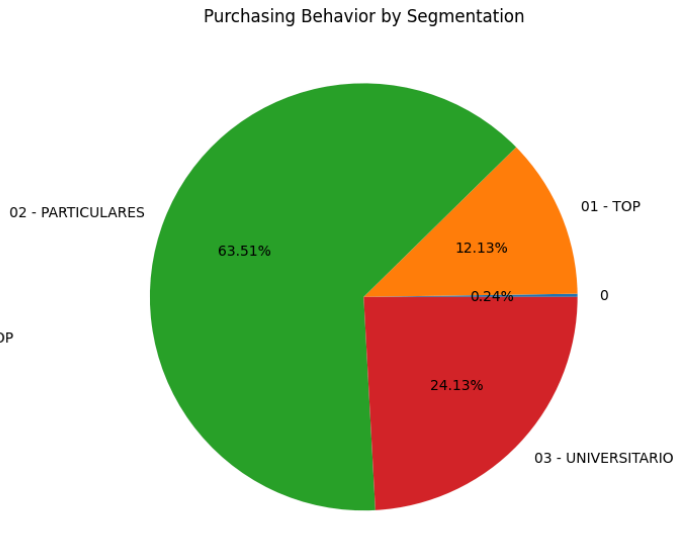
province_name	count	mean	std	min	25%	50%	75%	max
ALAVA	26.0	106274.0	54139.0	55271.0	63120.0	85711.0	164007.0	253563.0
ALBACETE	59642.0	83005.0	37528.0	9180.0	58193.0	78692.0	101539.0	764582.0
ALICANTE	145271.0	87024.0	167250.0	7791.0	45801.0	67277.0	101775.0	17804048.0
ALMERIA	28741.0	84877.0	50430.0	8291.0	53504.0	72941.0	99427.0	578349.0
ASTURIAS	121672.0	101333.0	89158.0	7619.0	66618.0	87096.0	115477.0	4950059.0
AVILA	18804.0	76458.0	70203.0	7290.0	51827.0	68548.0	90950.0	2768593.0
BADAJOZ	91239.0	72081.0	43650.0	7144.0	44542.0	62170.0	87592.0	1103543.0
BALEARS, ILLES	43227.0	171358.0	460933.0	5040.0	89579.0	122979.0	184843.0	15711716.0
BARCELONA	668909.0	164484.0	150320.0	1471.0	91886.0	130826.0	187443.0	5752268.0
BIZKAIA	107.0	106541.0	39712.0	34713.0	88930.0	99719.0	128379.0	247708.0
BURGOS	49268.0	97659.0	51151.0	9040.0	64568.0	89423.0	120292.0	1785512.0
CACERES	60386.0	75411.0	45468.0	5130.0	48687.0	67850.0	92686.0	1309035.0
CADIZ	137873.0	98806.0	89048.0	4560.0	56707.0	79071.0	113797.0	3648374.0
CANTABRIA	68454.0	121084.0	97208.0	10913.0	72010.0	95304.0	136964.0	2276562.0
CASTELLON	45444.0	79077.0	48734.0	6273.0	50630.0	66462.0	91804.0	668527.0
CEUTA	3100.0	201160.0	317236.0	33430.0	94551.0	130307.0	169055.0	4082464.0
CIUDAD REAL	61045.0	69896.0	42737.0	3732.0	46120.0	62186.0	83631.0	952513.0
CORDOBA	68925.0	85591.0	64722.0	7507.0	49462.0	68978.0	106387.0	1496216.0
CORUÑA, A	188185.0	112562.0	78382.0	2336.0	70276.0	97440.0	133304.0	2564976.0
CUENCA	26608.0	69784.0	35828.0	8633.0	43454.0	66915.0	90362.0	408454.0
GIPUZKOA	60.0	151197.0	130015.0	17686.0	53029.0	95784.0	290680.0	387346.0
GIRONA	42624.0	144916.0	191097.0	18545.0	77596.0	108986.0	161281.0	6209401.0
GRANADA	84677.0	96640.0	97844.0	7553.0	62107.0	82307.0	111629.0	4750243.0
GUADALAJARA	32373.0	95272.0	40803.0	9453.0	69108.0	92567.0	114016.0	780996.0
HUELVA	59542.0	76831.0	54469.0	7348.0	50976.0	68898.0	90388.0	1998667.0
HUESCA	18552.0	89305.0	92156.0	13373.0	56719.0	73294.0	97927.0	1137835.0
JAEN	31825.0	76999.0	44213.0	7425.0	48684.0	67826.0	92264.0	553668.0
LEON	40837.0	93359.0	69898.0	7817.0	60448.0	80923.0	110918.0	1985134.0
LERIDA	37842.0	81200.0	73106.0	9482.0	48419.0	64797.0	91611.0	3587378.0
LUGO	34047.0	76396.0	49303.0	7446.0	50427.0	64257.0	86699.0	1126464.0
MADRID	2374175.0	178591.0	338970.0	3797.0	91263.0	138240.0	205186.0	28894396.0
MALAGA	171454.0	121178.0	231228.0	10984.0	67689.0	95103.0	134166.0	13268621.0
MELILLA	4555.0	150564.0	191215.0	32199.0	95299.0	117515.0	159480.0	1959779.0

province_name	count	mean	std	min	25%	50%	75%	max
MURCIA	172920.0	78970.0	51331.0	7717.0	51855.0	67744.0	91445.0	3587378.0
NAVARRA	72.0	105478.0	78461.0	42003.0	54395.0	81614.0	101790.0	386063.0
OURENSE	34305.0	83159.0	37958.0	5438.0	59242.0	79006.0	100170.0	686178.0
PALENCIA	24378.0	92991.0	53240.0	12700.0	67114.0	86593.0	107253.0	833481.0
PALMAS, LAS	106515.0	100506.0	199066.0	7965.0	57261.0	80515.0	118044.0	15957372.0
PONTEVEDRA	119541.0	113487.0	81453.0	9450.0	76486.0	97719.0	124373.0	2570185.0
RIOJA, LA	45488.0	99687.0	47750.0	8917.0	66244.0	89754.0	121571.0	473760.0
SALAMANCA	79193.0	105200.0	192136.0	6313.0	68185.0	89664.0	116751.0	5431378.0
SANTA CRUZ DE TENERIFE	31344.0	102360.0	97827.0	12581.0	60998.0	82209.0	113696.0	3080266.0
SEGOVIA	20547.0	99003.0	57776.0	10462.0	64183.0	89589.0	118126.0	850010.0
SEVILLA	307924.0	117281.0	161543.0	5926.0	63833.0	92481.0	133899.0	11341152.0
SORIA	8223.0	88120.0	44445.0	13167.0	63860.0	78828.0	100382.0	423132.0
TARRAGONA	44119.0	104328.0	88842.0	9939.0	61202.0	87806.0	122572.0	2563288.0
TERUEL	9949.0	88023.0	57385.0	6360.0	56689.0	76703.0	104700.0	933320.0
TOLEDO	90906.0	80383.0	67067.0	1203.0	49090.0	68780.0	93471.0	3988595.0
VALENCIA	341769.0	89393.0	157616.0	7075.0	52437.0	72744.0	105757.0	25547252.0
VALLADOLID	129042.0	101523.0	58287.0	2540.0	66468.0	92593.0	120647.0	2257086.0
ZAMORA	22873.0	83597.0	91523.0	7776.0	55865.0	74742.0	94654.0	1536265.0
ZARAGOZA	176829.0	110188.0	109775.0	5652.0	69120.0	99565.0	131382.0	8516913.0

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

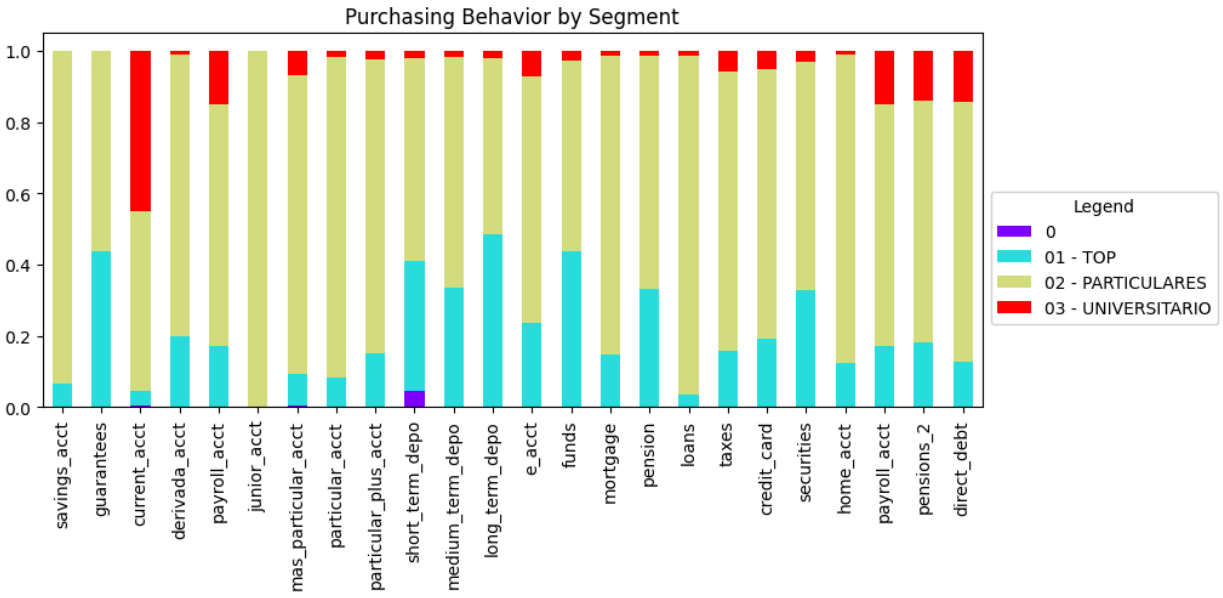
In [47]:
```
segmentation = train['segment'].value_counts()
seg_products = train.groupby('segment')['total_products'].sum()
seg_products = (seg_products/seg_products.sum())*100
seg_products

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 [48]:
```
print(train['segment'].value_counts())

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

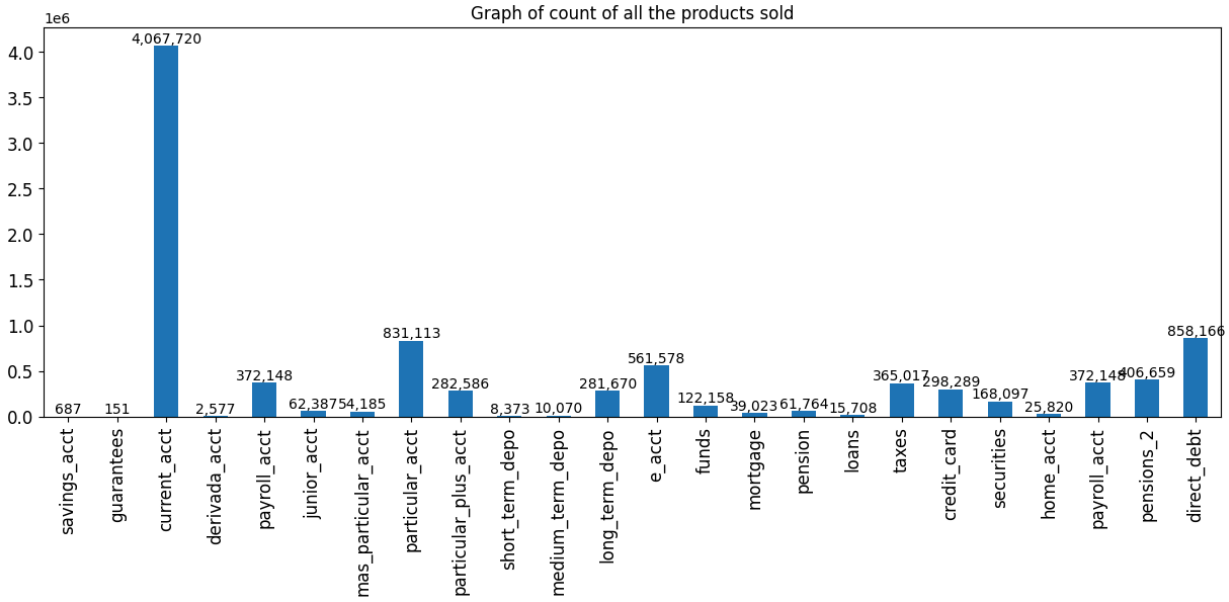
```
segment
02 - PARTICULARES     3974612
03 - UNIVERSITARIO    2292510
01 - TOP               274426
0                       43878
Name: count, dtype: int64
```



- 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 [49]:
```
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

End of Week 3