Piggy_bank

September 12, 2024

Personal loans are a lucrative revenue stream for banks. The typical interest rate of a two-year loan in the United Kingdom is around 10%. This might not sound like a lot, but in September 2022 alone UK consumers borrowed around £1.5 billion, which would mean approximately £300 million in interest generated by banks over two years!

You have been asked to work with a bank to clean the data they collected as part of a recent marketing campaign, which aimed to get customers to take out a personal loan. They plan to conduct more marketing campaigns going forward so would like you to ensure it conforms to the specific structure and data types that they specify so that they can then use the cleaned data you provide to set up a PostgreSQL database, which will store this campaign's data and allow data from future campaigns to be easily imported.

They have supplied you with a csv file called `"bank_marketing.csv"`, which you will need to clean, reformat, and split the data, saving three final csv files. Specifically, the three files should have the names and contents as outlined below:

0.1 client.csv

column	data type	description	cleaning requirements
client_id	integer	Client ID	N/A
age	integer	Client's age in years	N/A
job	object	Client's type of job	Change "." to "_"
marital	object	Client's marital status	N/A
education	object	Client's level of education	Change "." to "_" and "unknown" to np.NaN
credit_default	bool	Whether the client's credit is in default	Convert to boolean data type: 1 if "yes", otherwise 0
mortgage	bool	Whether the client has an existing mortgage (housing loan)	Convert to boolean data type: 1 if "yes", otherwise 0

0.2 campaign.csv

column	data type	description	cleaning requirements
client_id	integer	Client ID	N/A
number_contacts	integer	Number of contact attempts to the client in the current campaign	N/A
contact_duration	integer	Last contact duration in seconds	N/A
previous_campaign_contacts	integer	Number of contact attempts to the client in the previous campaign	N/A
previous_outcome	bool	Outcome of the previous campaign	Convert to boolean data type: 1 if "success", otherwise 0.
campaign_outcome	bool	Outcome of the current campaign	Convert to boolean data type: 1 if "yes", otherwise 0.
last_contact_date	datetime	Last date the client was contacted	Create from a combination of day, month, and a newly created year column (which should have a value of 2022); **Format** = "YYYY-MM-DD"

0.3 economics.csv

column	data type	description	cleaning requirements
client_id	integer	Client ID	N/A

column	data type	description	cleaning requirements
cons_price_idx	float	Consumer price index (monthly indicator)	N/A
euribor_three_months	float	Euro Interbank Offered Rate (euribor) three-month rate (daily indicator)	N/A

[2]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
import sys
sys.setrecursionlimit(1500)
```

[3]:
```
df = pd.read_csv("bank_marketing.csv")
df.head()
```

[3]:

	client_id	age	job	marital	education	credit_default	mortgage
0	0	56.0	housemaid	married	basic.4y	no	no
1	1	57.0	services	married	high.school	unknown	no
2	2	37.0	services	married	high.school	no	yes
3	3	40.0	admin.	married	basic.6y	no	no
4	4	56.0	services	married	high.school	no	no

	month	day	contact_duration	number_contacts	previous_campaign_contacts
0	may	13.0	261.0	1.0	0.0
1	may	19.0	149.0	1.0	0.0
2	may	23.0	226.0	1.0	0.0
3	may	27.0	151.0	1.0	0.0
4	may	3.0	307.0	1.0	0.0

	previous_outcome	cons_price_idx	euribor_three_months	campaign_outcome
0	nonexistent	93.994	4.857	no
1	nonexistent	93.994	4.857	no
2	nonexistent	93.994	4.857	no
3	nonexistent	93.994	4.857	no
4	nonexistent	93.994	4.857	no

[4]:
```
df.describe(include='all')
```

[4]:

	client_id	age	job	marital	education
count	11692.000000	11691.000000	11691	11691	11691
unique	NaN	NaN	12	4	8

top	NaN	NaN	blue-collar	married	high.school
freq	NaN	NaN	3481	7826	2803
mean	5844.500171	40.312805	NaN	NaN	NaN
std	3375.333712	8.912065	NaN	NaN	NaN
min	0.000000	20.000000	NaN	NaN	NaN
25%	2921.750000	33.000000	NaN	NaN	NaN
50%	5844.500000	39.000000	NaN	NaN	NaN
75%	8767.250000	47.000000	NaN	NaN	NaN
max	11690.000000	61.000000	NaN	NaN	NaN

	credit_default	mortgage	month	day	contact_duration	\
count	11691	11691	11691	11691.000000	11691.000000	
unique	2	3	2	NaN	NaN	
top	no	no	may	NaN	NaN	
freq	8003	6101	7763	NaN	NaN	
mean	NaN	NaN	NaN	15.703105	258.899153	
std	NaN	NaN	NaN	8.851462	253.476336	
min	NaN	NaN	NaN	1.000000	0.000000	
25%	NaN	NaN	NaN	8.000000	108.000000	
50%	NaN	NaN	NaN	16.000000	186.000000	
75%	NaN	NaN	NaN	23.000000	319.000000	
max	NaN	NaN	NaN	31.000000	3631.000000	

	number_contacts	previous_campaign_contacts	previous_outcome	\
count	11691.000000	11691.0	11691	
unique	NaN	NaN	1	
top	NaN	NaN	nonexistent	
freq	NaN	NaN	11691	
mean	2.737833	0.0	NaN	
std	3.130105	0.0	NaN	
min	1.000000	0.0	NaN	
25%	1.000000	0.0	NaN	
50%	2.000000	0.0	NaN	
75%	3.000000	0.0	NaN	
max	56.000000	0.0	NaN	

	cons_price_idx	euribor_three_months	campaign_outcome
count	11691.000000	11691.000000	11691
unique	NaN	NaN	2
top	NaN	NaN	no
freq	NaN	NaN	11283
mean	94.152249	4.881595	NaN
std	0.222479	0.042821	NaN
min	93.994000	4.855000	NaN
25%	93.994000	4.857000	NaN
50%	93.994000	4.859000	NaN
75%	94.465000	4.866000	NaN

```
max          94.465000              4.967000               NaN
```

[5]:
```
for col in ["credit_default", "mortgage", "previous_outcome","campaign_outcome"]:
    print(col)
    print("--------------")
    print(df[col].value_counts())
```

```
credit_default
--------------
credit_default
no         8003
unknown    3688
Name: count, dtype: int64
mortgage
--------------
mortgage
no         6101
yes        5236
unknown     354
Name: count, dtype: int64
previous_outcome
--------------
previous_outcome
nonexistent    11691
Name: count, dtype: int64
campaign_outcome
--------------
campaign_outcome
no     11283
yes      408
Name: count, dtype: int64
```

[6]:
```
# Change "." to "_" in job columns
df['job'].replace('.','_',inplace=True)
```

[7]:
```
#Change "." to "_" and "unknown" to np.NaN in education column
df['education'].replace('.','_',inplace=True)
df['education'].replace('unknown',np.nan,inplace=True)
```

[8]:
```
# Convert to boolean data type: 1 if "yes", otherwise 0 in credit_default col
df['credit_default']= df['credit_default'].map(lambda x:1 if x=='yes' else 0)
```

[9]:
```
df['credit_default'].value_counts()
```

[9]:
```
credit_default
0    11692
```

```
Name: count, dtype: int64
```

[10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11692 entries, 0 to 11691
Data columns (total 16 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   client_id                    11692 non-null  int64
 1   age                          11691 non-null  float64
 2   job                          11691 non-null  object
 3   marital                      11691 non-null  object
 4   education                    11123 non-null  object
 5   credit_default               11692 non-null  int64
 6   mortgage                     11691 non-null  object
 7   month                        11691 non-null  object
 8   day                          11691 non-null  float64
 9   contact_duration             11691 non-null  float64
 10  number_contacts              11691 non-null  float64
 11  previous_campaign_contacts   11691 non-null  float64
 12  previous_outcome             11691 non-null  object
 13  cons_price_idx               11691 non-null  float64
 14  euribor_three_months         11691 non-null  float64
 15  campaign_outcome             11691 non-null  object
dtypes: float64(7), int64(2), object(7)
memory usage: 1.4+ MB
```

[11]:
```
#fill missing columns
df.fillna(df.mode,inplace=True)
```

[12]:
```
df.isnull().sum()
```

[12]:
```
client_id                    0
age                          0
job                          0
marital                      0
education                    0
credit_default               0
mortgage                     0
month                        0
day                          0
contact_duration             0
number_contacts              0
previous_campaign_contacts   0
previous_outcome             0
cons_price_idx               0
```

```
euribor_three_months          0
campaign_outcome              0
dtype: int64
```

[13]:
```
# Split into the three tables
client = df[["client_id", "age", "job", "marital",
                    "education", "credit_default", "mortgage"]]
campaign = df[["client_id", "number_contacts", "month", "day",
               "contact_duration", "previous_campaign_contacts",␣
 ↪"previous_outcome", "campaign_outcome"]]
economics = df[["client_id", "cons_price_idx", "euribor_three_months"]]
```

[14]:
```
client.head()
```

[14]:

	client_id	age	job	marital	education	credit_default	mortgage
0	0	56.0	housemaid	married	basic.4y	0	no
1	1	57.0	services	married	high.school	0	no
2	2	37.0	services	married	high.school	0	yes
3	3	40.0	admin.	married	basic.6y	0	no
4	4	56.0	services	married	high.school	0	no

[15]:
```
# Editing the campaign dataset
# Change campaign_outcome to binary values
campaign["campaign_outcome"] = campaign["campaign_outcome"].map({"yes": 1,
                                                                 "no": 0})

# Convert previous_outcome to binary values
campaign["previous_outcome"] = campaign["previous_outcome"].map({"success": 1,
                                                                 "failure": 0,
                                                                 "nonexistent":␣
 ↪0})
```

[ ]:
```
# Add year column
campaign["year"] = "2022"

# Convert day to string
campaign["day"] = campaign["day"].astype(str)

# Add last_contact_date column
campaign["last_contact_date"] = campaign["year"] + "-" + campaign["month"] +␣
 ↪"-" + campaign["day"]

# Convert to datetime
campaign["last_contact_date"] = pd.to_datetime(campaign["last_contact_date"],
                                               format="%Y-%b-%d")
```

[21]:
```
economics.head()
```

[21]:
```
   client_id cons_price_idx euribor_three_months
0          0         93.994                4.857
1          1         93.994                4.857
2          2         93.994                4.857
3          3         93.994                4.857
4          4         93.994                4.857
```

[23]:
```
campaign.head()
```

[23]:
```
   client_id number_contacts month   day contact_duration  \
0          0             1.0   may  13.0            261.0
1          1             1.0   may  19.0            149.0
2          2             1.0   may  23.0            226.0
3          3             1.0   may  27.0            151.0
4          4             1.0   may   3.0            307.0

  previous_campaign_contacts  previous_outcome  campaign_outcome  year
0                        0.0               0.0               0.0  2022
1                        0.0               0.0               0.0  2022
2                        0.0               0.0               0.0  2022
3                        0.0               0.0               0.0  2022
4                        0.0               0.0               0.0  2022
```

[ ]:
```
# Clean and convert outcome columns to bool
for col in ["campaign_outcome", "previous_outcome"]:
  campaign[col] = campaign[col].astype(bool)

# Drop unneccessary columns
campaign.drop(columns=["month", "day", "year"], inplace=True)

# Save tables to individual csv files
client.to_csv("client.csv", index=False)
campaign.to_csv("campaign.csv", index=False)
economics.to_csv("economics.csv", index=False)
```

[30]:
```
campaign.head()
```

[30]:
```
   client_id number_contacts contact_duration previous_campaign_contacts  \
0          0             1.0            261.0                        0.0
1          1             1.0            149.0                        0.0
2          2             1.0            226.0                        0.0
3          3             1.0            151.0                        0.0
4          4             1.0            307.0                        0.0

   previous_outcome  campaign_outcome
0             False             False
1             False             False
```

```
2             False             False
3             False             False
4             False             False
```

[ ]: