Bank Marketing Campaign

Background

Personal loans are a lucrative revenue stream for banks. The typical interest rate of a two-year loan in the UK 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!

Clean the data 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.

Dataset

There's 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:

`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`

`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"`

`economics.csv`

column	data type	description	cleaning requirements
`client_id`	`integer`	Client ID	N/A
`cons_price_idx`	`float`	Consumer price index (monthly indicator)	N/A

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

Instructions

Subset, clean, and reformat the `bank_marketing.csv` dataset to create and store three new files.

- Split and tidy bank_marketing.csv, storing as three DataFrames called `client`, `campaign`, and `economics`, each containing the columns outlined in the notebook and formatted to the data types listed.
- Save the three DataFrames to csv files, without an index, as `client.csv`, `campaign.csv`, and `economics.csv` respectively.

In [6]:
```
import pandas as pd
import numpy as np

# Read in csv
marketing = pd.read_csv("../data_raw/bank_marketing.csv")
marketing
```

Out[6]:

	client_id	age	job	marital	education	credit_default	mortgag
0	0	56	housemaid	married	basic.4y	no	n
1	1	57	services	married	high.school	unknown	n
2	2	37	services	married	high.school	no	ye
3	3	40	admin.	married	basic.6y	no	n
4	4	56	services	married	high.school	no	n
...	...	...	...	...	...	...	..
41183	41183	73	retired	married	professional.course	no	ye
41184	41184	46	blue-collar	married	professional.course	no	n
41185	41185	56	retired	married	university.degree	no	ye
41186	41186	44	technician	married	professional.course	no	n
41187	41187	74	retired	married	professional.course	no	ye

41188 rows × 16 columns

```
In [7]: for col in ["credit_default", "mortgage", "previous_outcome", "campaign_outc
            print(col)
            print("--------------")
            print(marketing[col].value_counts())
```

```
credit_default
--------------
credit_default
no         32588
unknown     8597
yes            3
Name: count, dtype: int64
mortgage
--------------
mortgage
yes        21576
no         18622
unknown      990
Name: count, dtype: int64
previous_outcome
--------------
previous_outcome
nonexistent    35563
failure         4252
success         1373
Name: count, dtype: int64
campaign_outcome
--------------
campaign_outcome
no     36548
yes     4640
Name: count, dtype: int64
```

```
In [8]: # Split into the three tables
        client = marketing[["client_id", "age", "job", "marital",
                            "education", "credit_default", "mortgage"]].copy()
        campaign = marketing[["client_id", "number_contacts", "month", "day",
                         "contact_duration", "previous_campaign_contacts", "previous_c
        economics = marketing[["client_id", "cons_price_idx", "euribor_three_months"
```

```
In [9]: ### Editing the client dataset
        # Clean education column
        client.loc[:, "education"] = client["education"].str.replace(".", "_")
        client.loc[:, "education"] = client["education"].replace("unknown", np.nan)

        # Clean job column
        client.loc[:, "job"] = client["job"].str.replace(".", "_")

        # Clean and convert client columns to bool data type
        for col in ["credit_default", "mortgage"]:
          client.loc[:, col] = client[col].map({"yes": 1,
                                                "no": 0,
                                                "unknown": 0})
          client.loc[:, col] = client[col].astype(bool)
client.head(20)
# client.tail(20)
```

Out[9]:

	client_id	age	job	marital	education	credit_default	mortgage
0	0	56	housemaid	married	basic_4y	False	False
1	1	57	services	married	high_school	False	False
2	2	37	services	married	high_school	False	True
3	3	40	admin_	married	basic_6y	False	False
4	4	56	services	married	high_school	False	False
5	5	45	services	married	basic_9y	False	False
6	6	59	admin_	married	professional_course	False	False
7	7	41	blue-collar	married	NaN	False	False
8	8	24	technician	single	professional_course	False	True
9	9	25	services	single	high_school	False	True
10	10	41	blue-collar	married	NaN	False	False
11	11	25	services	single	high_school	False	True
12	12	29	blue-collar	single	high_school	False	False
13	13	57	housemaid	divorced	basic_4y	False	True
14	14	35	blue-collar	married	basic_6y	False	True
15	15	54	retired	married	basic_9y	False	True
16	16	35	blue-collar	married	basic_6y	False	True
17	17	46	blue-collar	married	basic_6y	False	True
18	18	50	blue-collar	married	basic_9y	False	True
19	19	39	management	single	basic_9y	False	False

In [10]:

```
### 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":
                                                                 "failure":
                                                                 "nonexisten

# 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"] +

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

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

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

campaign.head(20)
# campaign.tail(20)
```

Out[10]:

	client_id	number_contacts	contact_duration	previous_campaign_contacts	previo
0	0	1	261	0	
1	1	1	149	0	
2	2	1	226	0	
3	3	1	151	0	
4	4	1	307	0	
5	5	1	198	0	
6	6	1	139	0	
7	7	1	217	0	
8	8	1	380	0	
9	9	1	50	0	
10	10	1	55	0	
11	11	1	222	0	
12	12	1	137	0	
13	13	1	293	0	
14	14	1	146	0	
15	15	1	174	0	
16	16	1	312	0	
17	17	1	440	0	
18	18	1	353	0	
19	19	1	195	0	

In [11]:
```python
# Save tables to individual csv files
client.to_csv("../data_cleaned/client.csv", index=False)
campaign.to_csv("../data_cleaned/campaign.csv", index=False)
economics.to_csv("../data_cleaned/economics.csv", index=False)
```