Exercises

1. Produce a “people” file with the following schema. Save it as a CSV with a header line to the working directory.

2. Use the output of #1 to produce an “acquisition_facts” file with the following schema that aggregates stats
about when people in the dataset were acquired. Save it to the working directory.

Data

A dataset simulating CRM data is available in some public AWS S3 files:

● **Constituent Information:** https://als-hiring.s3.amazonaws.com/fake_data/2020-07-01_17%3A11%3A00/cons.csv (https://als-hiring.s3.amazonaws.com/fake_data/2020-07-01_17%3A11%3A00/cons.csv)

● **Constituent Email Addresses:** https://als-hiring.s3.amazonaws.com/fake_data/2020-07-01_17%3A11%3A00/cons_email.csv (https://als-hiring.s3.amazonaws.com/fake_data/2020-07-01_17%3A11%3A00/cons_email.csv)

- Boolean columns (including is_primary) in all of these datasets are 1/0 numeric values. 1 means True, 0 means False.

● **Constituent Subscription Status:** https://als-hiring.s3.amazonaws.com/fake_data/2020-0701_17%3A11%3A00/cons_email_chapter_subscription.csv (https://als-hiring.s3.amazonaws.com/fake_data/2020-0701_17%3A11%3A00/cons_email_chapter_subscription.csv)

- We only care about subscription statuses where chapter_id is 1.
- If an email is not present in this table, it is assumed to still be subscribed where chapter_id is 1.

'the script is wrote without consideration of memory'.

In [1]:
```
# need to install this package if not in the machine for this just uncoment it
'''
!pip install queries
!pip install sql-query
!pip install ipython-sql
!pip install --upgrade ipython
!pip install duckdb
!pip install pandasql

'''
```

Out[1]:
```
'\n!pip install queries\n!pip install sql-query\n!pip install ipython-sql\n!p
ip install --upgrade ipython\n!pip install duckdb\n!pip install pandasql\n\n'
```

In [2]:
```
#Import required Libraries
print("Import required Libraries...")
import pandas as pd
import requests, os
import pandasql as ps
import duckdb
import datetime
from datetime import datetime
from sklearn.impute import SimpleImputer
print("done import required Libraries")
```

```
Import required Libraries...
done import required Libraries
```

In [4]:
```
# Get data file from downloaded path
print("Import required dataframe...")
cons_csv = pd.read_csv("https://als-hiring.s3.amazonaws.com/fake_data/2020-07-(
cons_email_csv = pd.read_csv("https://als-hiring.s3.amazonaws.com/fake_data/20
cons_email_chp_csv = pd.read_csv("https://als-hiring.s3.amazonaws.com/fake_dat
print("Done import required dataframe")
```

```
Import required dataframe...
Done import required dataframe
```

check the shape of the Data Recall the number of Row and Columns

In [5]:

```
print(f"the cons.csv  Row is   {cons_csv.shape[0]}   and the number of Columns
print()
print(f"the cons_email.csv  Row is   {cons_email_csv.shape[0]}   and the numbe
print()
print(f"the cons_email_chapter_subscription.csv  Row is   {cons_email_chp_csv.
```

```
the cons.csv  Row is   700000   and the number of Columns is   29

the cons_email.csv  Row is   1400000   and the number of Columns is   16

the cons_email_chapter_subscription.csv  Row is   350000   and the number of
Columns is   6
```

In [6]:
```
cons_csv.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 700000 entries, 0 to 699999
Data columns (total 29 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   cons_id                     700000 non-null  int64  
 1   prefix                      350304 non-null  object 
 2   firstname                   350244 non-null  object 
 3   middlename                  560213 non-null  object 
 4   lastname                    349314 non-null  object 
 5   suffix                      349541 non-null  object 
 6   salutation                  350021 non-null  object 
 7   gender                      349891 non-null  object 
 8   birth_dt                    349954 non-null  object 
 9   title                       350082 non-null  object 
 10  employer                    349228 non-null  object 
 11  occupation                  350239 non-null  object 
 12  income                      350637 non-null  float64
 13  source                      350026 non-null  object 
 14  subsource                   350315 non-null  object 
 15  userid                      700000 non-null  int64  
 16  password                    700000 non-null  object 
 17  is_validated                700000 non-null  int64  
 18  is_banned                   700000 non-null  int64  
 19  change_password_next_login  700000 non-null  int64  
 20  consent_type_id             700000 non-null  int64  
 21  create_dt                   700000 non-null  object 
 22  create_app                  700000 non-null  int64  
 23  create_user                 700000 non-null  int64  
 24  modified_dt                 700000 non-null  object 
 25  modified_app                700000 non-null  int64  
 26  modified_user               700000 non-null  int64  
 27  status                      700000 non-null  int64  
 28  note                        69884 non-null   object 
dtypes: float64(1), int64(11), object(17)
memory usage: 154.9+ MB
```

In [7]:
```
cons_email_csv.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1400000 entries, 0 to 1399999
Data columns (total 16 columns):
 #   Column                Non-Null Count    Dtype
---  ------                --------------    -----
 0   cons_email_id         1400000 non-null  int64
 1   cons_id               1400000 non-null  int64
 2   cons_email_type_id    1400000 non-null  int64
 3   is_primary            1400000 non-null  int64
 4   email                 1400000 non-null  object
 5   canonical_local_part  700029 non-null   object
 6   domain                1400000 non-null  object
 7   double_validation     699825 non-null   object
 8   create_dt             1400000 non-null  object
 9   create_app            1400000 non-null  int64
 10  create_user           1400000 non-null  int64
 11  modified_dt           1400000 non-null  object
 12  modified_app          1400000 non-null  int64
 13  modified_user         1400000 non-null  int64
 14  status                1400000 non-null  int64
 15  note                  139535 non-null   object
dtypes: int64(9), object(7)
memory usage: 170.9+ MB
```

In [8]:
```
cons_email_chp_csv.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 350000 entries, 0 to 349999
Data columns (total 6 columns):
 #   Column                              Non-Null Count   Dtype
---  ------                              --------------   -----
 0   cons_email_chapter_subscription_id  350000 non-null  int64
 1   cons_email_id                       350000 non-null  int64
 2   chapter_id                          350000 non-null  int64
 3   isunsub                             350000 non-null  int64
 4   unsub_dt                            350000 non-null  object
 5   modified_dt                         350000 non-null  object
dtypes: int64(4), object(2)
memory usage: 16.0+ MB
```

recall the first 2 lines of each Dataframes

In [9]:
```
cons_csv = cons_csv.sort_values(by=['cons_id'], ascending=True)
print("table from Dataframe cons.csv")
print()
cons_csv.head(2)
```

```
table from Dataframe cons.csv
```

Out[9]:

	cons_id	prefix	firstname	middlename	lastname	suffix	salutation	gender	birt
0	1	NaN	NaN	Lee	NaN	MD	NaN	E	
1	2	NaN	NaN	NaN	NaN	II	boFqBKgLlSgEZsFrgCZd	E	2 1

2 rows × 29 columns

In [10]:
```
cons_email_csv = cons_email_csv.sort_values(by=['cons_email_id'], ascending=Tru
print("table from Dataframe cons_email.csv")
print()
cons_email_csv.head(2)
```

```
table from Dataframe cons_email.csv
```

Out[10]:

	cons_email_id	cons_id	cons_email_type_id	is_primary	email	canonical_lo
0	1	548198	3361	1	xmartinez@vincent.com	
1	2	491137	2474	1	hmiller@haynes.biz	jqCyozTDojYuyl

In [11]:
```
cons_email_chp_csv = cons_email_chp_csv.sort_values(by=['cons_email_id'], ascer
print("table from Dataframe cons_email_chapter_subscription.csv")
print()
cons_email_chp_csv.head(2)
```

```
table from Dataframe cons_email_chapter_subscription.csv
```

Out[11]:

	cons_email_chapter_subscription_id	cons_email_id	chapter_id	isunsub	unsub_dt	modi
108533	108534	3	1	1	Mon, 1973-08-20 02:16:04	Sun 1:
296065	296066	4	1	1	Tue, 1974-07-09 06:29:34	Wed 0:

Join the Dataframe from cons.csv with cons_email.csv

In [12]:
```
df = pd.merge(cons_csv , cons_email_csv ,left_on="cons_id", right_on="cons_id"
print(f"the cons.csv + cons_email.csv  row nbr is   {df.shape[0]}   and the nur
```

```
the cons.csv + cons_email.csv  row nbr is   1494361   and the number of colum
ns is   44
```

In [13]:
```
df2 = pd.merge(df , cons_email_chp_csv ,left_on="cons_email_id", right_on="cons
print(f"the cons.csv + cons_email.csv + cons_email_chp_csv row nbr is   {df2.sh
```

```
the cons.csv + cons_email.csv + cons_email_chp_csv row nbr is   1568877   and
the number of columns is   49
```

query the DataFrame cons.csv + cons_email.csv + cons_email_chp_csv according to this filter

- We only care about subscription statuses where chapter_id is 1.
- If an email is not present in this table, it is assumed to stil
l be subscribed where chapter_id is 1.

In [14]:
```
#call the engine from duckdb
def dbrun(query: str) -> pd.DataFrame:
    result = dbcon.query(query).to_df()
    return result
#connect to engine
dbcon = duckdb.connect()
#run the query and save data as df3
df3 = dbrun('''SELECT * FROM df2 WHERE email IS NULL OR email = '' OR chapter_

'''
#without usesing the step to join or merge the table first we can run the query
df3 = dbrun("""SELECT *FROM cons_csv a LEFT JOIN cons_email_csv b ON a.cons_id
'''
# Filter the Data on "email" ,"source", "isunsub" , "create_dt_csv2", "modifie
Table_exercise1 = df3[["email" ,"source", "isunsub" , "is_primary" , "create_d
```

In [15]:
```
# check duplicate on column email
boolea = Table_exercise1["email"].duplicated().any() # True
print(f"{boolea}  there is duplicate on email column and the Dataframe")

# cheack missing value
print()
print()
print("this is the statistique of missing value in the data")
print()
percentage_missing_each_col = pd.DataFrame(Table_exercise1.isnull().sum())
percentage_missing_each_col.rename(columns = {0:"number of missing value"}, in

percentage_missing_each_col["percentage(%)"] = (percentage_missing_each_col["n

percentage_missing_each_col
```

```
True  there is duplicate on email column and the Dataframe


this is the statistique of missing value in the data
```

Out[15]:

	number of missing value	percentage(%)
email	94361	6.0
source	184493	12.0
isunsub	94361	6.0
is_primary	94361	6.0
create_dt_csv2	94361	6.0
modified_dt	94361	6.0

Treat the missing value on the source column here we will use sklearn imputer on most frequent to replace the missing value

In [16]:
```
# let call the sklean imputer on  most frequent
the_imputer = SimpleImputer(strategy='most_frequent')

# select the column to impute
#column_to_impute = 'source'

# now let impute missing values in selected column source
imputed_column = the_imputer.fit_transform(Table_exercise1[['source']])

# use the value and replace the original column
Table_exercise1 = Table_exercise1.assign(source=lambda d: pd.DataFrame(imputed_

#print the statistic of missing value and recall that the source column is now

print()
print()
print("this is the statistique of missing value in the data with treated source
print()
percentage_missing_each_col2 = pd.DataFrame(Table_exercise1.isnull().sum())
percentage_missing_each_col2.rename(columns = {0:"number of missing value"}, i

percentage_missing_each_col2["percentage(%)"] = (percentage_missing_each_col2[

percentage_missing_each_col2
```

```
this is the statistique of missing value in the data with treated source colu
mn
```

Out[16]:

	number of missing value	percentage(%)
email	94361	6.0
source	0	0.0
isunsub	94361	6.0
is_primary	94361	6.0
create_dt_csv2	94361	6.0
modified_dt	94361	6.0

In [17]:
```
# check duplicate again on column email
boole = Table_exercise1["email"].duplicated().any() # True
print(f"{boole}  there is duplicate on email column and the Dataframe and we w

#since the duplicate here are not impacting our analysis, we will not remove i
#Table_exercise1.drop_duplicates(subset="email", keep="last")

# From the Dataframe select all column  except the column where email is missi
Table_exercise1 = Table_exercise1[Table_exercise1['email'].notna()]

print()
print("Now there is no duplicate on email and the other duplicate are reasonab
```

```
True  there is duplicate on email column and the Dataframe and we will take t
hem out treament...

Now there is no duplicate on email and the other duplicate are reasonable,we
procede to treat duplicate on email only
```

In [18]:
```
# change the 0 and 1 in the table to False and True boolean
for i in range(len(Table_exercise1.columns)):
    if set(Table_exercise1.iloc[:,i].unique()) == set([0,1]):
        Table_exercise1.iloc[:,i]= Table_exercise1.iloc[:,i].replace([1,0],['T

# Rename column with proper name
Table_exercise1.rename(columns = {"source":"code", "isunsub":"is_unsub", "crea
                                  "modified_dt":"updated_dt"}, inplace = True)

#filter the table and name it Email_table
Email_table = Table_exercise1[["email" ,"code", "is_unsub" , "created_dt", "up

# Set value on the Dataframe to proper type
Email_table.loc[ : , "email"] = Email_table["email"].convert_dtypes(convert_st
Email_table.loc[ : , "code"]  = Email_table["code"].convert_dtypes(convert_str
Email_table.loc[ : , "is_unsub"] = Email_table["is_unsub"].convert_dtypes(conv
Email_table.is_unsub = Email_table.is_unsub.astype('bool')
Email_table.loc[ : , "created_dt"] = pd.to_datetime(Email_table["created_dt"])
Email_table.loc[ : , "updated_dt"] = pd.to_datetime(Email_table["updated_dt"])
```

In [19]:

```
# Define a funtion to create folder in the working directory the name of the fo

# Folder 1
def create_folder(FolderName):
    date = datetime.now()
    now = date.strftime("%Y-%m-%d %H-%M")
    print (f'creating folder {FolderName}....')
    print (f'this table will be in {FolderName} that is in Email_table folder'
    newpath0 = os.path.join(os.getcwd(),FolderName+now)
    if not os.path.exists(newpath0):
        os.makedirs(newpath0)
    Path0 = newpath0.rstrip('\n')
    filenameCreated0 = Path0
    return filenameCreated0

# Folder 2
def create_folder2(FolderName):
    date = datetime.now()
    now = date.strftime("%Y-%m-%d %H-%M")
    print (f'creating folder {FolderName}....')
    print (f'this table will be in {FolderName} that is in Date_table_aggregate
    newpath0 = os.path.join(os.getcwd(),FolderName+now)
    if not os.path.exists(newpath0):
        os.makedirs(newpath0)
    Path0 = newpath0.rstrip('\n')
    filenameCreated0 = Path0
    return filenameCreated0
```

Save the Dataframe Email_table in the folder in working directory this is the table for ETL excercie 1

In [20]:
```
print("This is the recall of all Dataframe column type data from Email_table.cs
print()
Email_table.info()
print()
print()
# Create folder and file name
FolderName1 = "ETL_exercise_1_"
FileName1 = "Email_table.csv"
# Create folder in working directory giving the defien funtion.
create_folder(FolderName1)
# Set the path of the file to save
path123 = os.path.join(create_folder(FolderName1), FileName1)
# Save the file
Email_table.to_csv(path123)
print()
print()
print("This is the recall of first 6 row and columns of Email_table.csv")
Email_table.head(6)
```

```
This is the recall of all Dataframe column type data from Email_table.csv

<class 'pandas.core.frame.DataFrame'>
Int64Index: 275484 entries, 0 to 369844
Data columns (total 5 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   email       275484 non-null  string
 1   code        275484 non-null  string
 2   is_unsub    275484 non-null  bool
 3   created_dt  275484 non-null  datetime64[ns]
 4   updated_dt  275484 non-null  datetime64[ns]
dtypes: bool(1), datetime64[ns](2), string(2)
memory usage: 10.8 MB


creating folder ETL_exercise_1_....
this table will be in ETL_exercise_1_ that is in Email_table folder
creating folder ETL_exercise_1_....
this table will be in ETL_exercise_1_ that is in Email_table folder


This is the recall of first 6 row and columns of Email_table.csv
```

Out[20]:

	email	code	is_unsub	created_dt	updated_dt
0	caustin@spears-carson.com	facebook	True	1987-01-23 13:51:26	1988-12-19 12:14:02
1	deborah57@shaffer-reed.org	facebook	True	1995-02-03 05:29:21	2009-12-10 06:34:18
2	klewis@ford.biz	organic	True	2013-05-02 09:20:11	1985-07-16 03:09:10
94364	stephenhamilton@gmail.com	organic	True	1976-05-23 12:23:38	1991-03-17 04:25:12
94365	inovak@barnett-wise.com	organic	True	2004-01-11 14:35:11	1978-11-05 23:14:09
94366	norr@donovan.com	twitter	True	1973-06-22 22:47:25	2011-02-08 12:49:10

Save the Dataframe Date_table_aggregate in the folder in working directory. the table is for ETL excercie2

In [21]:
```
# Create the Datafome for excercise 2 from ETL excercise
Date_table_aggr = Email_table[[ "created_dt"]].copy()
# Rename the column to acquisition_date
Date_table_aggr.rename(columns = {"created_dt":"acquisition_date"}, inplace = 
# Create new column acquisitions that will show aggragate date count
Date_table_aggr = Date_table_aggr.groupby(['acquisition_date']).size().reset_i
# Sort the value from the column acquisitions as descending
Date_table_aggr = Date_table_aggr.sort_values("acquisitions",axis=0,ascending=

Date_table_aggr[ "acquisition_date"] = pd.to_datetime(Date_table_aggr[ "acquis
Date_table_aggr.reset_index(drop=True, inplace=True)
print("This is the recall of all Dataframe column type data in Date_table_aggr
print()
print(f"{Date_table_aggr.info()}")
print()
print()
# Create folder and file name
FolderName2 = "ETL_exercise_2_"
FileName2 = "Date_table_aggregate.csv"
# Create folder in working directory giving the defien funtion.
create_folder2(FolderName2)
# Set the path of the file to save
path123 = os.path.join(create_folder(FolderName2), FileName2)
# Save the file
Email_table.to_csv(path123)
print()
print()
print("This is the recall of first 6 row and columns of Date_table_aggr.csv")
Date_table_aggr.head(6)
```

```
This is the recall of all Dataframe column type data in Date_table_aggr.csv

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 275458 entries, 0 to 275457
Data columns (total 2 columns):
 #   Column            Non-Null Count   Dtype         
---  ------            --------------   -----         
 0   acquisition_date  275458 non-null  datetime64[ns]
 1   acquisitions      275458 non-null  int64         
dtypes: datetime64[ns](1), int64(1)
memory usage: 4.2 MB
None


creating folder ETL_exercise_2_....
this table will be in ETL_exercise_2_ that is in Date_table_aggregate folder
creating folder ETL_exercise_2_....
this table will be in ETL_exercise_2_ that is in Email_table folder


This is the recall of first 6 row and columns of Date_table_aggr.csv
```

Out[21]:

	acquisition_date	acquisitions
0	2018-09-25	2
1	2012-12-29	2
2	1970-06-22	2
3	1998-02-28	2
4	2015-03-03	2
5	2012-11-01	2

In [ ]: