

## Importing libraries

In [2]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
# for the Q-Q plots
# import scipy.stats as stats
%matplotlib inline
import pandas as pd
pd.options.display.float_format = '{:.2f}'.format
# from pandas.io.json import json_normalize
```

## Loading dataset for Users

In [3]:

```
users = pd.read_excel("users.xlsx")
```

In [4]:

```
users.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 495 entries, 0 to 494
Data columns (total 7 columns):
 _id/$oid      495 non-null object
 active        495 non-null bool
 createdAt     495 non-null int64
 lastLoginDate 433 non-null float64
 role          495 non-null object
 signUpSource   447 non-null object
 state         439 non-null object
dtypes: bool(1), float64(1), int64(1), object(4)
memory usage: 23.8+ KB
```

In [5]:

```
users["lastLoginDate"] = users["lastLoginDate"].astype(str)
```

In [6]:

```
users.head()
```

Out[6]:

	_id/\$oid	active	createdDate	lastLoginDate	role	signUpSource	s
0	5ff1e194b6a9d73a3a9f1052	True	1609687444800	1609687537858.0	consumer	Email	
1	5ff1e194b6a9d73a3a9f1052	True	1609687444800	1609687537858.0	consumer	Email	
2	5ff1e194b6a9d73a3a9f1052	True	1609687444800	1609687537858.0	consumer	Email	
3	5ff1e1eacfcf6c399c274ae6	True	1609687530554	1609687530597.0	consumer	Email	
4	5ff1e194b6a9d73a3a9f1052	True	1609687444800	1609687537858.0	consumer	Email	



## Identifying numerical and categorical variables

In [7]:

```
# make lists of variable types

temporal = [var for var in users.columns if 'date' in var or 'Date' in var]

discrete = [
    var for var in users.columns if users[var].dtype != 'O'
    and len(users[var].unique()) < 20 and var not in temporal
]

continuous = [
    var for var in users.columns if users[var].dtype != 'O'
    if var not in discrete and var != '_id'
    and var not in temporal
]

categorical = [var for var in users.columns if users[var].dtype == 'O'
    and var not in temporal and var not in discrete]

print(f'There are {len(continuous)} continuous variables')
print(f'There are {len(discrete)} discrete variables')
print(f'There are {len(temporal)} temporal variables')
print(f'There are {len(categorical)} categorical variables')
```

```
There are 0 continuous variables
There are 1 discrete variables
There are 2 temporal variables
There are 4 categorical variables
```

In [8]:

discrete

Out[8]:

['active']

In [9]:

temporal

Out[9]:

['createdDate', 'lastLoginDate']

In [10]:

categorical

Out[10]:

['\_id/\$oid', 'role', 'signUpSource', 'state']

## Quantifying missing data

In [11]:

users.isnull().sum()

Out[11]:

_id/\$oid	0
active	0
createdDate	0
lastLoginDate	0
role	0
signUpSource	48
state	56
dtype:	int64

### *percentage of missing values in variables*

In [12]:

```
# alternatively, we can use the mean() method after isnull() to visualise the percentage of
# missing values for each variable
percentage_null_values= users.isnull().mean()
for key,value in percentage_null_values.items():
    if value >0:
        print(key,":",value*100)
```

```
signUpSource : 9.696969696969697
state : 11.313131313131313
```

## Checking for redundant records

In [13]:

```
duplicateRowsDF = users[users.duplicated()]
print("Duplicate Rows except first occurrence based on all columns are :")
print(duplicateRowsDF)
```

```
Duplicate Rows except first occurrence based on all columns are :
   _id/$oid  active  createdAt  lastLoginDate \
1  5ff1e194b6a9d73a3a9f1052  True  1609687444800  1609687537858.0
2  5ff1e194b6a9d73a3a9f1052  True  1609687444800  1609687537858.0
4  5ff1e194b6a9d73a3a9f1052  True  1609687444800  1609687537858.0
5  5ff1e194b6a9d73a3a9f1052  True  1609687444800  1609687537858.0
8  5ff1e194b6a9d73a3a9f1052  True  1609687444800  1609687537858.0
..      ...      ...      ...      ...
490  54943462e4b07e684157a532  True  1418998882381  1614963143204.0
491  54943462e4b07e684157a532  True  1418998882381  1614963143204.0
492  54943462e4b07e684157a532  True  1418998882381  1614963143204.0
493  54943462e4b07e684157a532  True  1418998882381  1614963143204.0
494  54943462e4b07e684157a532  True  1418998882381  1614963143204.0

   role signUpSource state
1  consumer      Email  WI
2  consumer      Email  WI
4  consumer      Email  WI
5  consumer      Email  WI
8  consumer      Email  WI
..      ...      ...      ...
490  fetch-staff      NaN  NaN
491  fetch-staff      NaN  NaN
492  fetch-staff      NaN  NaN
493  fetch-staff      NaN  NaN
494  fetch-staff      NaN  NaN
```

[283 rows x 7 columns]

We have a considerably large number of redundant(duplicated) user records.

## Unique values of categorical variables

Unique values:

In [14]:

```
users["role"].unique()
```

Out[14]:

```
array(['consumer', 'fetch-staff'], dtype=object)
```

In [15]:

```
users["signUpSource"].unique()
```

Out[15]:

```
array(['Email', 'Google', nan], dtype=object)
```

In [16]:

```
users["state"].unique()
```

Out[16]:

```
array(['WI', 'KY', 'AL', 'CO', 'IL', nan, 'OH', 'SC', 'NH'], dtype=object)
```

## Examining percentage of different category values for categorical variables

In [17]:

```
freq_source = 100*(users['signUpSource'].value_counts() / len(users))
print(freq_source.map('{:,.2f} %'.format))
```

```
Email      89.49 %
Google      0.81 %
Name: signUpSource, dtype: object
```

Dominant sign up source is Email.

In [18]:

```
freq_state = 100*(users['state'].value_counts() / len(users))
print(freq_state.map('{:,.2f} %'.format))
```

```
WI      80.00 %
NH       4.04 %
AL       2.42 %
OH       1.01 %
IL       0.61 %
CO       0.20 %
KY       0.20 %
SC       0.20 %
Name: state, dtype: object
```

As per this data, majority of the users that have signed up reside in Wisconsin.

Data quality issues found:

1. More than half of the user records(out of the total 495) are redundant.

Another, less pressing issue:

1. small percentage of missing values in signUpSource and state columns