In [ ]:
```
#This notebook aims to analyze user data, highlighting any potential i
#The initial step involves reading the data and restructuring the nest
```

In [*]:
```
# import pandas as pd
import json

# Open the file and read lines
with open('/Users/project/users.json', 'r') as file:
    data = file.readlines()

# Parse JSON and convert to DataFrame
users_df = pd.json_normalize([json.loads(line) for line in data])
```

users_df.info()

```
users_df.head(5)
```

In [ ]:
```
# converting dates into standard format
```

In [46]:
```
users_df['lastLoginDate']   = pd.to_datetime(users_df['lastLogin.$date
users_df['createdDate'] = pd.to_datetime(users_df['createdDate.$date']
```

In [ ]:
```
# Removing the column from the original dataframe since we have alread
```

In [37]:
```
users_df.head(5)
```

Out[37]:

	active	role	signUpSource	state	_id.$oid	createdDate.$date	lastLog
0	True	consumer	Email	WI	5ff1e194b6a9d73a3a9f1052	1609687444800	1.609
1	True	consumer	Email	WI	5ff1e194b6a9d73a3a9f1052	1609687444800	1.609
2	True	consumer	Email	WI	5ff1e194b6a9d73a3a9f1052	1609687444800	1.609
3	True	consumer	Email	WI	5ff1e1eacfcf6c399c274ae6	1609687530554	1.609
4	True	consumer	Email	WI	5ff1e194b6a9d73a3a9f1052	1609687444800	1.609

In [47]:
```
users_df.drop(columns=['createdDate.$date'], inplace=True)
users_df.drop(columns=['lastLogin.$date'], inplace=True)
```

In [48]:
```
users_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 495 entries, 0 to 494
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   active         495 non-null    bool          
 1   role           495 non-null    object        
 2   signUpSource   447 non-null    object        
 3   state          439 non-null    object        
 4   _id.$oid       495 non-null    object        
 5   lastLoginDate  433 non-null    datetime64[ns]
 6   createdDate    495 non-null    datetime64[ns]
dtypes: bool(1), datetime64[ns](2), object(4)
memory usage: 23.8+ KB
```

In [ ]:
```
# The dataset indeed contains null values
```

In [49]:
```
users_df.rename(columns={'_id.$oid': 'userId'}, inplace=True)
users_df.rename(columns={'createdDate.$date': 'createdDate'}, inplace=
users_df.rename(columns={'lastLogin.$date': 'lastLogin'}, inplace=True
```

In [51]:
```
users_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 495 entries, 0 to 494
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   active         495 non-null    bool          
 1   role           495 non-null    object        
 2   signUpSource   447 non-null    object        
 3   state          439 non-null    object        
 4   userId         495 non-null    object        
 5   lastLoginDate  433 non-null    datetime64[ns]
 6   createdDate    495 non-null    datetime64[ns]
dtypes: bool(1), datetime64[ns](2), object(4)
memory usage: 23.8+ KB
```

In [52]:
```
users_df.isnull().sum()
```

Out[52]:
```
active           0
role             0
signUpSource    48
state           56
userId           0
lastLoginDate   62
createdDate      0
dtype: int64
```

In [ ]:
```
# There are duplicate values with them
```

In [53]:
```
duplicate_rows = users_df[users_df.duplicated()]
duplicate_rows.count()
```

Out[53]:
```
active          283
role            283
signUpSource    240
state           233
userId          283
lastLoginDate   261
createdDate     283
dtype: int64
```

In [55]:
```
def has_special_characters(text):
    import re
    # Check if the value is NaN
    if pd.isna(text):
        return False
    # Define a regular expression pattern to match special characters
    pattern = r'[^a-zA-Z0-9\s]'
    # Use re.search to check if the pattern matches any part of the te
    if re.search(pattern, str(text)):
        return True
    else:
        return False

# Apply the function to the column
users_df['has_special_char'] = users_df['userId'].apply(has_special_ch

# Filter rows with special characters
rows_with_special_chars = users_df[users_df['has_special_char']]

print(rows_with_special_chars)
```

```
Empty DataFrame
Columns: [active, role, signUpSource, state, userId, lastLoginDate, c
reatedDate, has_special_char]
Index: []
```

In [56]:
```
# Extract the year from the 'lastLogin' column
users_df['cyears'] = users_df['createdDate'].dt.year

print(users_df)
```

```
     active         role signUpSource state                    userId
\
0      True     consumer        Email    WI  5ff1e194b6a9d73a3a9f1052
1      True     consumer        Email    WI  5ff1e194b6a9d73a3a9f1052
2      True     consumer        Email    WI  5ff1e194b6a9d73a3a9f1052
3      True     consumer        Email    WI  5ff1e1eacfcf6c399c274ae6
4      True     consumer        Email    WI  5ff1e194b6a9d73a3a9f1052
..      ...          ...          ...   ...                       ...
490    True  fetch-staff          NaN   NaN  54943462e4b07e684157a532
491    True  fetch-staff          NaN   NaN  54943462e4b07e684157a532
492    True  fetch-staff          NaN   NaN  54943462e4b07e684157a532
493    True  fetch-staff          NaN   NaN  54943462e4b07e684157a532
494    True  fetch-staff          NaN   NaN  54943462e4b07e684157a532

                    lastLoginDate             createdDate  has_specia
l_char  \
0   2021-01-03 15:25:37.857999872 2021-01-03 15:24:04.800
False
```

```
1   2021-01-03 15:25:37.857999872 2021-01-03 15:24:04.800
False
2   2021-01-03 15:25:37.857999872 2021-01-03 15:24:04.800
False
3   2021-01-03 15:25:30.596999936 2021-01-03 15:25:30.554
False
4   2021-01-03 15:25:37.857999872 2021-01-03 15:24:04.800
False
..                          ...                    ...
...
490 2021-03-05 16:52:23.204000000 2014-12-19 14:21:22.381
False
491 2021-03-05 16:52:23.204000000 2014-12-19 14:21:22.381
False
492 2021-03-05 16:52:23.204000000 2014-12-19 14:21:22.381
False
493 2021-03-05 16:52:23.204000000 2014-12-19 14:21:22.381
False
494 2021-03-05 16:52:23.204000000 2014-12-19 14:21:22.381
False

     cyears
0      2021
1      2021
2      2021
3      2021
4      2021
..      ...
490    2014
491    2014
492    2014
493    2014
494    2014

[495 rows x 9 columns]
```

In [57]:
```
users_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 495 entries, 0 to 494
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype         
---  ------            --------------  -----         
 0   active            495 non-null    bool          
 1   role              495 non-null    object        
 2   signUpSource      447 non-null    object        
 3   state             439 non-null    object        
 4   userId            495 non-null    object        
 5   lastLoginDate     433 non-null    datetime64[ns]
 6   createdDate       495 non-null    datetime64[ns]
 7   has_special_char  495 non-null    bool          
 8   cyears            495 non-null    int32         
dtypes: bool(2), datetime64[ns](2), int32(1), object(4)
memory usage: 26.2+ KB
```

In [67]:
```
columns_to_delete = ['has_special_char','cyears']
```

In [68]:
```
users_df = users_df.drop(columns=columns_to_delete)
```

In [58]:
```
value_counts = users_df['signUpSource'].value_counts()

# Print the result
print(value_counts)
```

```
signUpSource
Email     443
Google      4
Name: count, dtype: int64
```

In [59]:
```
value_counts_state = users_df['state'].value_counts()

# Print the result
print(value_counts_state)
```

```
state
WI    396
NH     20
AL     12
OH      5
IL      3
KY      1
CO      1
SC      1
Name: count, dtype: int64
```

In [60]:
```
value_counts_role = users_df['role'].value_counts()

# Print the result
print(value_counts_role)
```

```
role
consumer       413
fetch-staff     82
Name: count, dtype: int64
```

In [ ]:
```
# Issues or Anamolies present in the 'Users' dataset

# userId contains duplicate values [283/494]. This would impact when c
# There are no special characters present in any of the columns
# 'signUpSource' contains null values when the creation date years wer
```

In [72]:
```
users_df.head(1)
```

Out[72]:

	active	role	signUpSource	state	userId	lastLoginDate	create
0	True	consumer	Email	WI	5ff1e194b6a9d73a3a9f1052	2021-01-03 15:25:37.857999872	2021 15:24:

In [ ]: