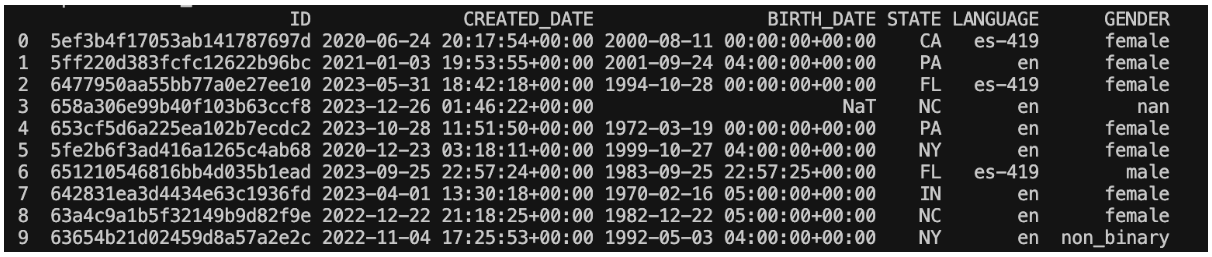
Exploration of Users Table

Displaying first few rows of Users Table:

Query:

```
# Display first few rows of each dataset
print("\Users Data:")
print(users_df.head(10))
```

Output:



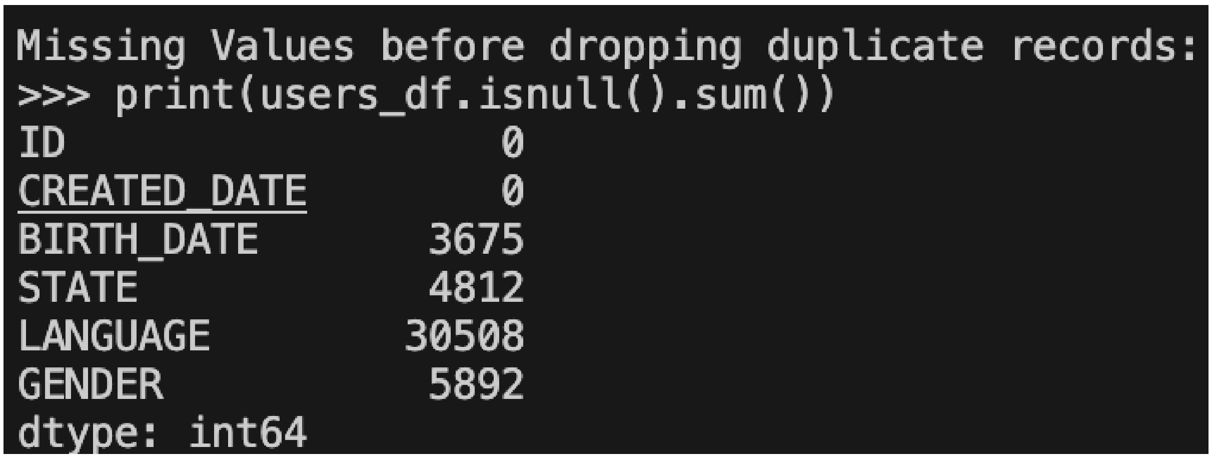
```
                         ID              CREATED_DATE                BIRTH_DATE STATE LANGUAGE      GENDER
0  5ef3b4f17053ab141787697d 2020-06-24 20:17:54+00:00 2000-08-11 00:00:00+00:00    CA   es-419      female
1  5ff220d383fcfc12622b96bc 2021-01-03 19:53:55+00:00 2001-09-24 04:00:00+00:00    PA       en      female
2  6477950aa55bb77a0e27ee10 2023-05-31 18:42:18+00:00 1994-10-28 00:00:00+00:00    FL   es-419      female
3  658a306e99b40f103b63ccf8 2023-12-26 01:46:22+00:00                       NaT    NC       en         nan
4  653cf5d6a225ea102b7ecdc2 2023-10-28 11:51:50+00:00 1972-03-19 00:00:00+00:00    PA       en      female
5  5fe2b6f3ad416a1265c4ab68 2020-12-23 03:18:11+00:00 1999-10-27 04:00:00+00:00    NY       en      female
6  651210546816bb4d035b1ead 2023-09-25 22:57:24+00:00 1983-09-25 22:57:25+00:00    FL   es-419        male
7  642831ea3d4434e63c1936fd 2023-04-01 13:30:18+00:00 1970-02-16 05:00:00+00:00    IN       en      female
8  63a4c9a1b5f32149b9d82f9e 2022-12-22 21:18:25+00:00 1982-12-22 05:00:00+00:00    NC       en      female
9  63654b21d02459d8a57a2e2c 2022-11-04 17:25:53+00:00 1992-05-03 04:00:00+00:00    NY       en  non_binary
```

Displaying Missing Values from the Users table:

Query:

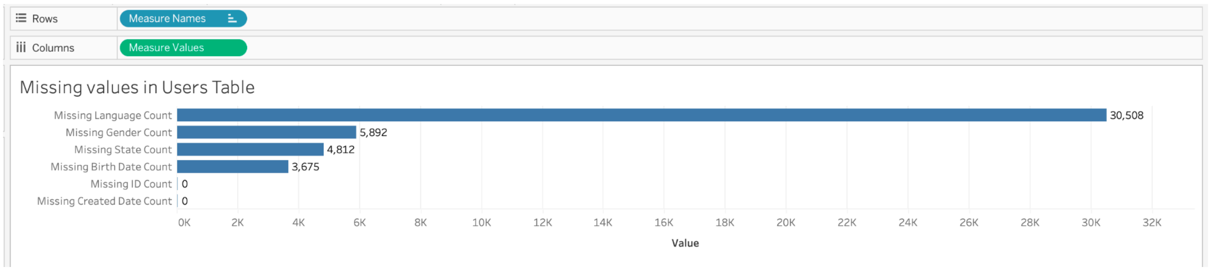
```
print("\nMissing Values before dropping duplicate records:")
print(users_df.isnull().sum())
```

Output:



```
Missing Values before dropping duplicate records:
>>> print(users_df.isnull().sum())
ID                  0
CREATED_DATE        0
BIRTH_DATE       3675
STATE            4812
LANGUAGE        30508
GENDER           5892
dtype: int64
```

Graphical view:

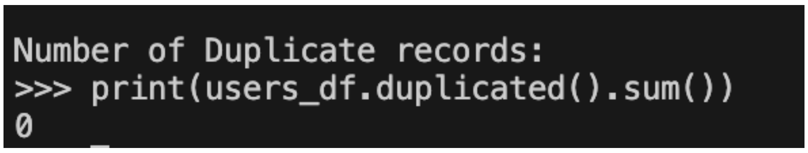


Displaying Duplicate Records:

Query:

```
# Duplicate records in products table
print("\nNumber of Duplicate records:")
print(users_df.duplicated().sum())
```

Output:



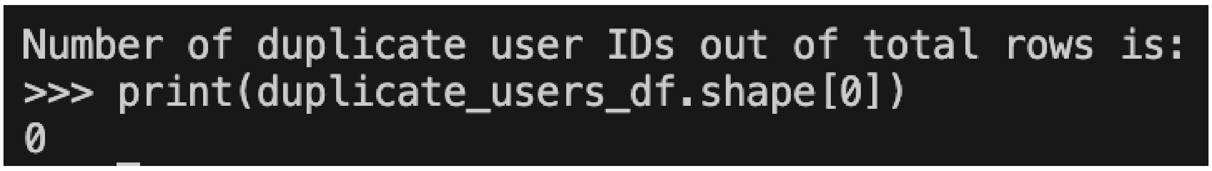
```
Number of Duplicate records:
>>> print(users_df.duplicated().sum())
0
```

Displaying Duplicate User ID:

Query:

```
# Filter the dataframe to include only duplicate IDs
duplicate_users_df = users_df[users_df['ID'].duplicated(keep=False)]

# The percentage of duplicate ID rows out of the total rows
print("\Number of duplicate user IDs out of total rows is: ")
print(duplicate_users_df.shape[0])
```

Output:



```
Number of duplicate user IDs out of total rows is:
>>> print(duplicate_users_df.shape[0])
0
```

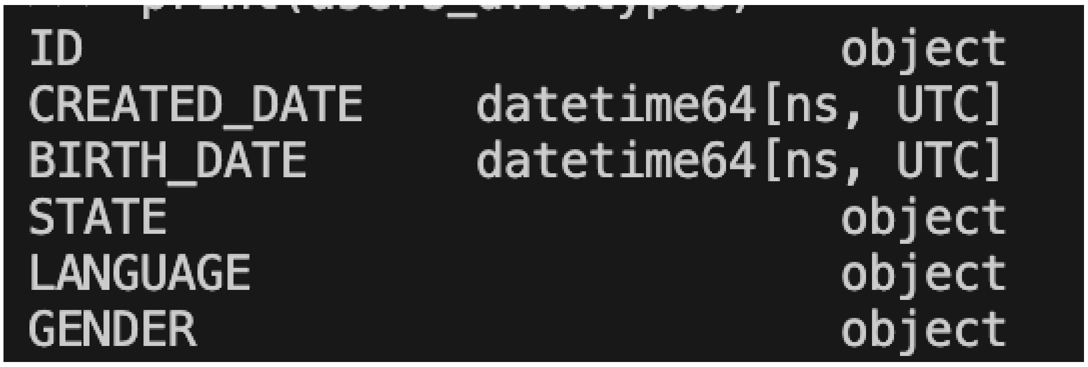
Converting the columns of Users table into their appropriate data types:

Query:

```
# Convert date columns to datetime format:
```

```
users_df['CREATED_DATE'] = pd.to_datetime(users_df['CREATED_DATE'],
errors='coerce')
users_df['BIRTH_DATE'] = pd.to_datetime(users_df['BIRTH_DATE'], errors='coerce')

# Convert ID, STATE, LANGUAGE, GENDER to varchar
users_df['ID'] = users_df['ID'].astype(str)
users_df['STATE'] = users_df['STATE'].astype(str)
users_df['LANGUAGE'] = users_df['LANGUAGE'].astype(str)
users_df['GENDER'] = users_df['GENDER'].astype(str)

# Check the data types
print(users_df.dtypes)
```

Output:



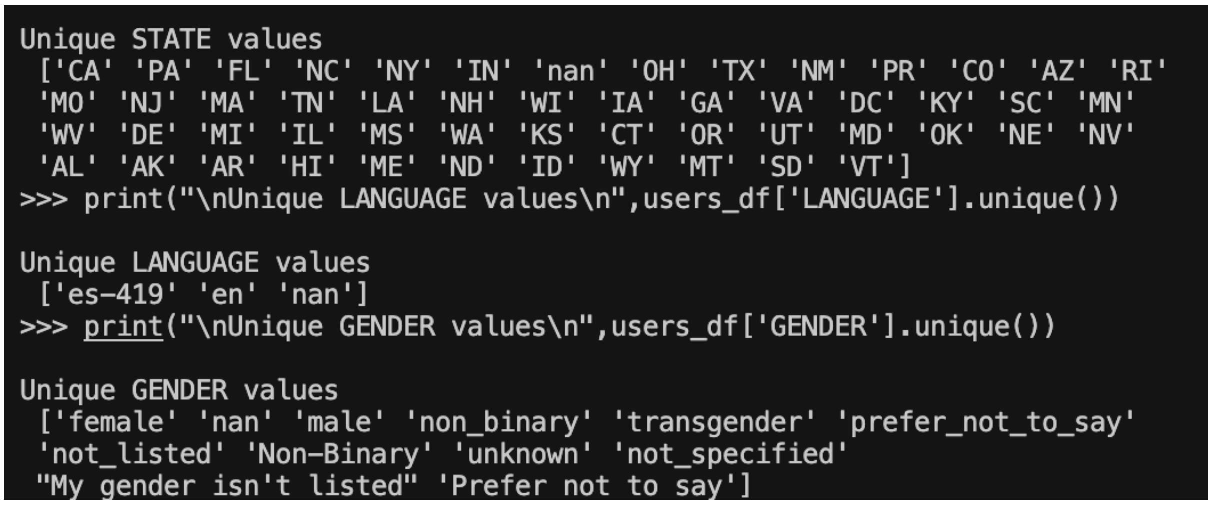
```
ID                              object
CREATED_DATE     datetime64[ns, UTC]
BIRTH_DATE       datetime64[ns, UTC]
STATE                           object
LANGUAGE                        object
GENDER                          object
```

Checking the unique values in ‘STATE’, ‘LANGUAGE’, and ‘GENDER’ columns:

Query:

```
# Check the unique values in each column
print("\nUnique STATE values\n",users_df['STATE'].unique())
print("\nUnique LANGUAGE values\n",users_df['LANGUAGE'].unique())
print("\nUnique GENDER values\n",users_df['GENDER'].unique())
```

Output:



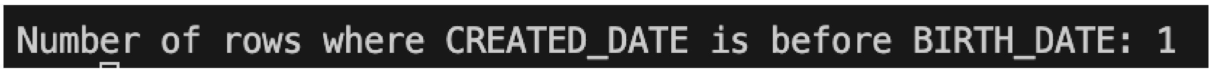
```
Unique STATE values
 ['CA' 'PA' 'FL' 'NC' 'NY' 'IN' 'nan' 'OH' 'TX' 'NM' 'PR' 'CO' 'AZ' 'RI'
 'MO' 'NJ' 'MA' 'TN' 'LA' 'NH' 'WI' 'IA' 'GA' 'VA' 'DC' 'KY' 'SC' 'MN'
 'WV' 'DE' 'MI' 'IL' 'MS' 'WA' 'KS' 'CT' 'OR' 'UT' 'MD' 'OK' 'NE' 'NV'
 'AL' 'AK' 'AR' 'HI' 'ME' 'ND' 'ID' 'WY' 'MT' 'SD' 'VT']
>>> print("\nUnique LANGUAGE values\n",users_df['LANGUAGE'].unique())

Unique LANGUAGE values
 ['es-419' 'en' 'nan']
>>> print("\nUnique GENDER values\n",users_df['GENDER'].unique())

Unique GENDER values
 ['female' 'nan' 'male' 'non_binary' 'transgender' 'prefer_not_to_say'
 'not_listed' 'Non-Binary' 'unknown' 'not_specified'
 "My gender isn't listed" 'Prefer not to say']
```

Checking for inconsistency in CREATED_DATE and BIRTH_DATE in USERS table:

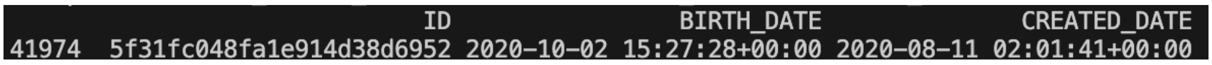
Query:

```
# Check where CREATED_DATE is before BIRTH_DATE
created_before_birthdate = users_df[users_df['CREATED_DATE'] <
users_df['BIRTH_DATE']]

# Count occurrences
count_created_before_birthdate = created_before_birthdate.shape[0]

# Display results
print(f"\nNumber of rows where CREATED_DATE is before BIRTH_DATE:
{count_created_before_birthdate}")

# Display some of these rows
print(created_before_birthdate[['ID','BIRTH_DATE', 'CREATED_DATE']].head(10))
```

Output:



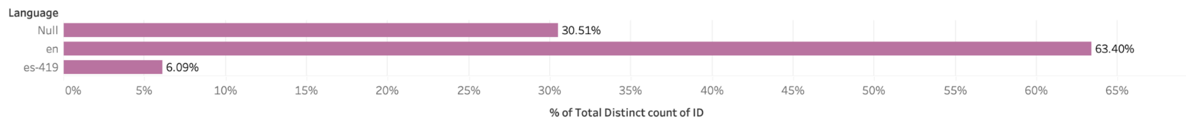
```
Number of rows where CREATED_DATE is before BIRTH_DATE: 1
```



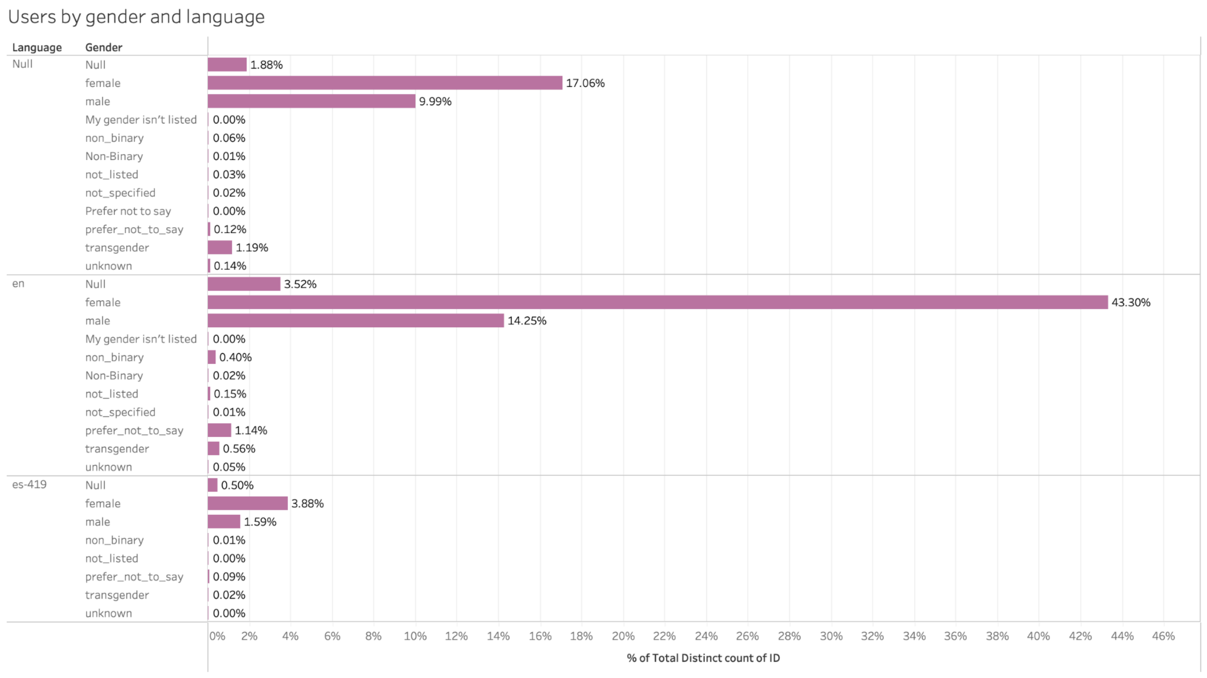
```
                             ID                 BIRTH_DATE              CREATED_DATE
41974  5f31fc048fa1e914d38d6952 2020-10-02 15:27:28+00:00 2020-08-11 02:01:41+00:00
```

Showing some interesting user demographics in graphical format:

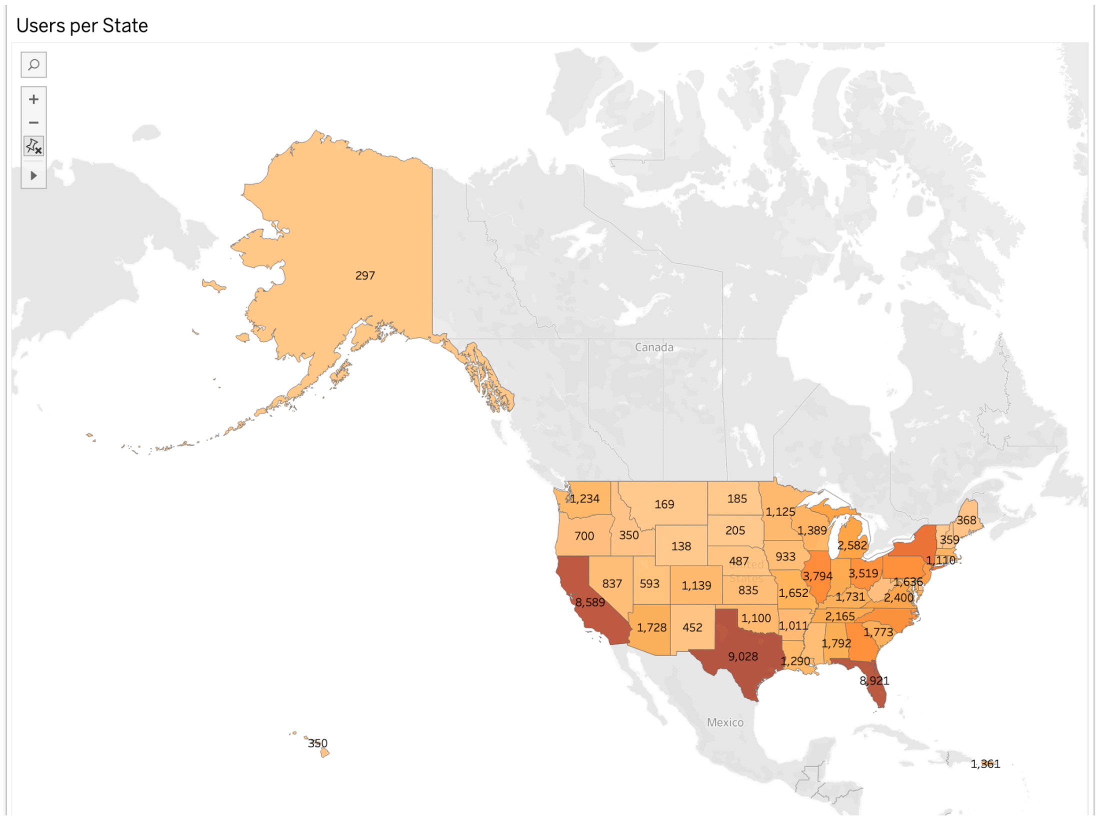
Graphical view of Language spoken vs Percent of User IDs:



Graphical view of Percent of Users by Language and Gender:



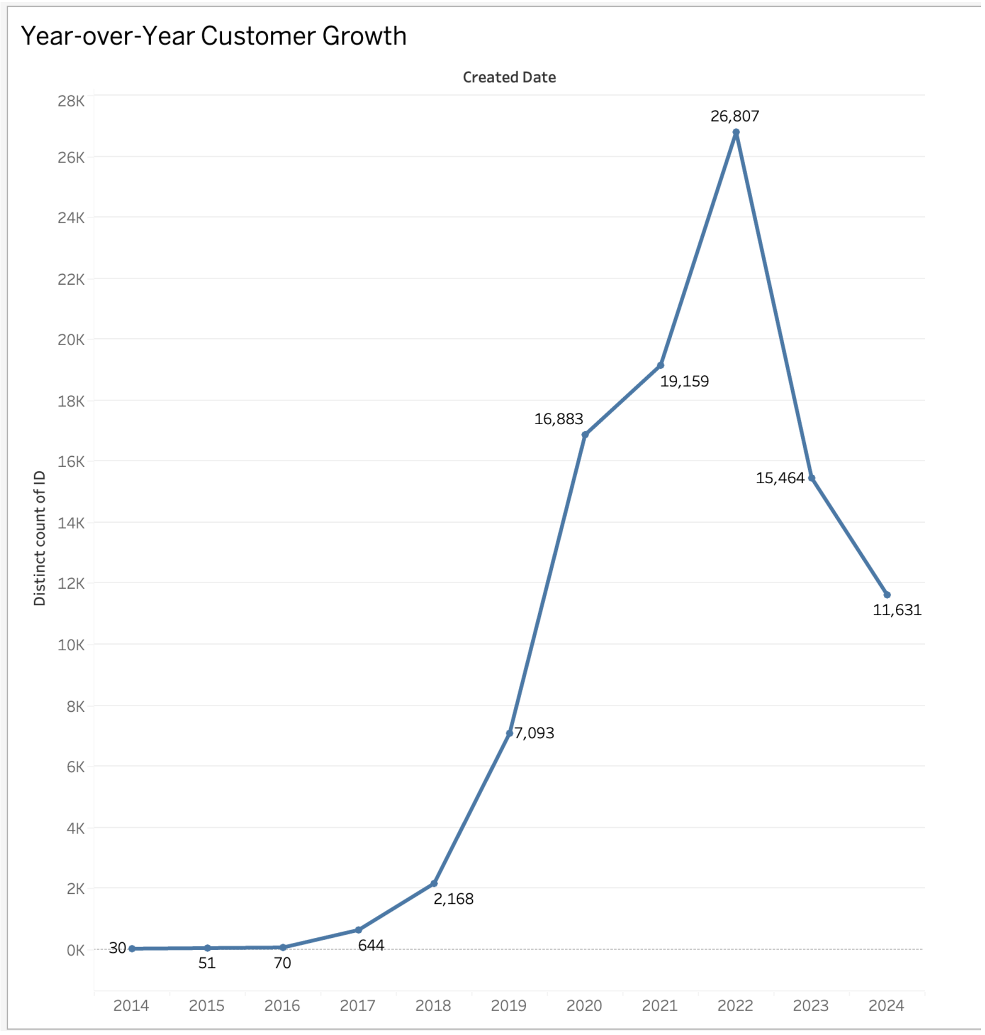
Graphical view of Users by States:



Foundings:

1. The visuals clearly indicate that the majority of users, approximately 63% of the total 100,000, prefer English as their language. Additionally, 43% of the total users who prefer English are female.
2. Majority of our users are from California, Texas and Florida

Graphical view of Year-over-Year Customer Growth:



Foundings:

a. Steady Growth Until 2018
b. Rapid Expansion from 2018 to 2022
c. Decline in Growth from 2023 Onward

Assessment Answers: Data Quality & Challenges in the Users Table

Q. Are there any data quality issues present?

Following data quality issues were found in the users table:

Potential Data Inconsistencies:

CREATED_DATE occurs before BIRTH_DATE in 1 case: There is 1 field where **CREATED_DATE** is before **BIRTH_DATE,** which does not make sense.

Q. Are there any fields that are challenging to understand?

There are no fields in the Users Table that are challenging to understand. The exploration process included displaying data, identifying duplicates, handling missing values, converting data types, and checking inconsistencies. Each column, such as ID, STATE, LANGUAGE, GENDER, CREATED_DATE, and BIRTH_DATE, has clear meanings and expected values. The queries and visualizations provide a straightforward understanding of the dataset, making all fields comprehensible.