Airbnb New User Bookings_1

December 30, 2024

========== Data Cleaning, Analysis, and Exploratory Data Insight ==========

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

1 Downloading Dataset from Kaggle Using API

[2]:
```
import json
import os
from kaggle.api.kaggle_api_extended import KaggleApi

# Load the kaggle.json file manually
with open('F:/Andalosia/kaggle.json') as f:
    kaggle_json = json.load(f)

# Set up the API client
os.environ['KAGGLE_USERNAME'] = kaggle_json['username']
os.environ['KAGGLE_KEY'] = kaggle_json['key']

# Authenticate the API client
api = KaggleApi()
api.authenticate()

#  download the dataset
api.competition_download_files('airbnb-recruiting-new-user-bookings', path='F:/
 ↪Andalosia')
```

[3]:
```
df_sample_submission_NDF = pd.read_csv('F:/Andalosia/
 ↪airbnb-recruiting-new-user-bookings/sample_submission_NDF.csv')
df_sessions = pd.read_csv('F:/Andalosia/airbnb-recruiting-new-user-bookings/
 ↪sessions.csv')
df_countries = pd.read_csv('F:/Andalosia/airbnb-recruiting-new-user-bookings/
 ↪countries.csv')
df_age_gender = pd.read_csv('F:/Andalosia/airbnb-recruiting-new-user-bookings/
 ↪age_gender_bkts.csv')
```

[120]:
```
df_train = pd.read_csv('F:/Andalosia/airbnb-recruiting-new-user-bookings/
↪train_users_2.csv')
df_test = pd.read_csv('F:/Andalosia/airbnb-recruiting-new-user-bookings/
↪test_users.csv')
```

2 Data Exploration

[5]:
```
df_train
```

[5]:

```
                id date_account_created  timestamp_first_active  \
0       gxn3p5htnn           2010-06-28          20090319043255
1       820tgsjxq7           2011-05-25          20090523174809
2       4ft3gnwmtx           2010-09-28          20090609231247
3       bjjt8pjhuk           2011-12-05          20091031060129
4       87mebub9p4           2010-09-14          20091208061105
…              …                    …                       …
213446  zxodksqpep           2014-06-30          20140630235636
213447  mhewnxesx9           2014-06-30          20140630235719
213448  6o3arsjbb4           2014-06-30          20140630235754
213449  jh95kwisub           2014-06-30          20140630235822
213450  nw9fwlyb5f           2014-06-30          20140630235824

       date_first_booking     gender   age signup_method  signup_flow  \
0                     NaN  -unknown-   NaN      facebook            0
1                     NaN       MALE  38.0      facebook            0
2              2010-08-02     FEMALE  56.0         basic            3
3              2012-09-08     FEMALE  42.0      facebook            0
4              2010-02-18  -unknown-  41.0         basic            0
…                     …        …   …           …          …
213446                NaN       MALE  32.0         basic            0
213447                NaN  -unknown-   NaN         basic            0
213448                NaN  -unknown-  32.0         basic            0
213449                NaN  -unknown-   NaN         basic           25
213450                NaN  -unknown-   NaN         basic           25

       language affiliate_channel affiliate_provider first_affiliate_tracked  \
0            en            direct             direct                untracked
1            en               seo             google                untracked
2            en            direct             direct                untracked
3            en            direct             direct                untracked
4            en            direct             direct                untracked
…          …                 …                  …                        …
213446       en       sem-brand             google                      omg
213447       en            direct             direct                   linked
213448       en            direct             direct                untracked
213449       en             other              other           tracked-other
```

```
213450       en            direct              direct                untracked

       signup_app first_device_type  first_browser country_destination
0             Web       Mac Desktop         Chrome                 NDF
1             Web       Mac Desktop         Chrome                 NDF
2             Web   Windows Desktop             IE                  US
3             Web       Mac Desktop        Firefox               other
4             Web       Mac Desktop         Chrome                  US
…             …                 …              …                   …
213446        Web       Mac Desktop         Safari                 NDF
213447        Web   Windows Desktop         Chrome                 NDF
213448        Web       Mac Desktop        Firefox                 NDF
213449        iOS            iPhone  Mobile Safari                 NDF
213450        iOS            iPhone      -unknown-                 NDF

[213451 rows x 16 columns]
```

[6]:
```
print(f'shape of data is {df_train.shape[0]} , {df_train.shape[1]}')
```

```
shape of data is 213451 , 16
```

[7]:
```
df_train.dtypes
```

[7]:
```
id                         object
date_account_created       object
timestamp_first_active      int64
date_first_booking         object
gender                     object
age                       float64
signup_method              object
signup_flow                 int64
language                   object
affiliate_channel          object
affiliate_provider         object
first_affiliate_tracked    object
signup_app                 object
first_device_type          object
first_browser              object
country_destination        object
dtype: object
```

[8]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 213451 entries, 0 to 213450
Data columns (total 16 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
```

```
 0   id                       213451 non-null  object
 1   date_account_created     213451 non-null  object
 2   timestamp_first_active   213451 non-null  int64
 3   date_first_booking       88908 non-null   object
 4   gender                   213451 non-null  object
 5   age                      125461 non-null  float64
 6   signup_method            213451 non-null  object
 7   signup_flow              213451 non-null  int64
 8   language                 213451 non-null  object
 9   affiliate_channel        213451 non-null  object
 10  affiliate_provider       213451 non-null  object
 11  first_affiliate_tracked  207386 non-null  object
 12  signup_app               213451 non-null  object
 13  first_device_type        213451 non-null  object
 14  first_browser            213451 non-null  object
 15  country_destination      213451 non-null  object
dtypes: float64(1), int64(2), object(13)
memory usage: 26.1+ MB
```

[9]:
```
for col in df_train.columns:
    print(f' {col}:  \n number of unique value for each column {df_train[col].
 ↪nunique()} , \n unique values is {df_train[col].unique()}')
    print('='*100)
```

```
 id:
 number of unique value for each column 213451 ,
 unique values is ['gxn3p5htnn' '820tgsjxq7' '4ft3gnwmtx' … '6o3arsjbb4'
'jh95kwisub'
 'nw9fwlyb5f']
====================================================================================
====================
 date_account_created:
 number of unique value for each column 1634 ,
 unique values is ['2010-06-28' '2011-05-25' '2010-09-28' … '2014-06-27'
'2014-06-29'
 '2014-06-30']
====================================================================================
====================
 timestamp_first_active:
 number of unique value for each column 213451 ,
 unique values is [20090319043255 20090523174809 20090609231247 …
20140630235754
 20140630235822 20140630235824]
====================================================================================
====================
 date_first_booking:
 number of unique value for each column 1976 ,
 unique values is [nan '2010-08-02' '2012-09-08' … '2015-06-25' '2015-06-29'
```

```
'2015-06-28']
================================================================================
====================
 gender:
 number of unique value for each column 4 ,
 unique values is ['-unknown-' 'MALE' 'FEMALE' 'OTHER']
================================================================================
====================
 age:
 number of unique value for each column 127 ,
 unique values is [      nan 3.800e+01 5.600e+01 4.200e+01 4.100e+01 4.600e+01
4.700e+01
 5.000e+01 3.600e+01 3.700e+01 3.300e+01 3.100e+01 2.900e+01 3.000e+01
 4.000e+01 2.600e+01 3.200e+01 3.500e+01 5.900e+01 4.900e+01 4.400e+01
 3.400e+01 2.800e+01 1.900e+01 5.300e+01 5.200e+01 3.900e+01 5.700e+01
 2.500e+01 5.400e+01 6.900e+01 6.300e+01 4.300e+01 5.500e+01 6.500e+01
 5.800e+01 6.100e+01 1.800e+01 5.000e+00 2.700e+01 4.500e+01 6.000e+01
 4.800e+01 5.100e+01 6.400e+01 7.200e+01 7.000e+01 6.700e+01 7.300e+01
 2.014e+03 1.040e+02 6.600e+01 1.050e+02 6.800e+01 9.500e+01 2.400e+01
 9.400e+01 7.500e+01 7.900e+01 6.200e+01 2.013e+03 1.600e+01 4.000e+00
 2.300e+01 1.010e+02 9.800e+01 7.600e+01 7.400e+01 8.700e+01 9.200e+01
 1.150e+02 7.100e+01 8.400e+01 1.070e+02 7.800e+01 9.900e+01 1.100e+02
 1.020e+02 8.200e+01 7.700e+01 2.200e+01 8.900e+01 2.100e+01 1.500e+01
 2.000e+01 1.030e+02 2.000e+00 1.700e+01 8.600e+01 9.700e+01 8.100e+01
 9.000e+01 8.800e+01 8.000e+01 1.000e+02 9.100e+01 1.060e+02 8.300e+01
 8.500e+01 1.080e+02 1.130e+02 1.090e+02 9.300e+01 9.600e+01 1.949e+03
 1.110e+02 1.931e+03 1.932e+03 1.120e+02 1.928e+03 1.000e+00 1.936e+03
 1.933e+03 1.935e+03 1.925e+03 1.952e+03 1.500e+02 1.927e+03 1.320e+02
 1.953e+03 1.942e+03 1.995e+03 2.008e+03 1.924e+03 1.929e+03 1.947e+03
 1.938e+03 1.926e+03]
================================================================================
====================
 signup_method:
 number of unique value for each column 3 ,
 unique values is ['facebook' 'basic' 'google']
================================================================================
====================
 signup_flow:
 number of unique value for each column 17 ,
 unique values is [ 0  3  2  1 24  8  6  5 10 25 12  4 16 15 20 21 23]
================================================================================
====================
 language:
 number of unique value for each column 25 ,
 unique values is ['en' 'fr' 'de' 'es' 'it' 'pt' 'zh' 'ko' 'ja' 'ru' 'pl' 'el'
'sv' 'nl'
 'hu' 'da' 'id' 'fi' 'no' 'tr' 'th' 'cs' 'hr' 'ca' 'is']
================================================================================
```

```
====================
 affiliate_channel:
 number of unique value for each column 8 ,
 unique values is ['direct' 'seo' 'other' 'sem-non-brand' 'content' 'sem-brand'
 'remarketing' 'api']
================================================================================
====================
 affiliate_provider:
 number of unique value for each column 18 ,
 unique values is ['direct' 'google' 'other' 'craigslist' 'facebook' 'vast'
'bing' 'meetup'
 'facebook-open-graph' 'email-marketing' 'yahoo' 'padmapper' 'gsp' 'wayn'
 'naver' 'baidu' 'yandex' 'daum']
================================================================================
====================
 first_affiliate_tracked:
 number of unique value for each column 7 ,
 unique values is ['untracked' 'omg' nan 'linked' 'tracked-other' 'product'
'marketing'
 'local ops']
================================================================================
====================
 signup_app:
 number of unique value for each column 4 ,
 unique values is ['Web' 'Moweb' 'iOS' 'Android']
================================================================================
====================
 first_device_type:
 number of unique value for each column 9 ,
 unique values is ['Mac Desktop' 'Windows Desktop' 'iPhone' 'Other/Unknown'
 'Desktop (Other)' 'Android Tablet' 'iPad' 'Android Phone'
 'SmartPhone (Other)']
================================================================================
====================
 first_browser:
 number of unique value for each column 52 ,
 unique values is ['Chrome' 'IE' 'Firefox' 'Safari' '-unknown-' 'Mobile Safari'
 'Chrome Mobile' 'RockMelt' 'Chromium' 'Android Browser' 'AOL Explorer'
 'Palm Pre web browser' 'Mobile Firefox' 'Opera' 'TenFourFox' 'IE Mobile'
 'Apple Mail' 'Silk' 'Camino' 'Arora' 'BlackBerry Browser' 'SeaMonkey'
 'Iron' 'Sogou Explorer' 'IceWeasel' 'Opera Mini' 'SiteKiosk' 'Maxthon'
 'Kindle Browser' 'CoolNovo' 'Conkeror' 'wOSBrowser' 'Google Earth'
 'Crazy Browser' 'Mozilla' 'OmniWeb' 'PS Vita browser' 'NetNewsWire'
 'CometBird' 'Comodo Dragon' 'Flock' 'Pale Moon' 'Avant Browser'
 'Opera Mobile' 'Yandex.Browser' 'TheWorld Browser' 'SlimBrowser' 'Epic'
 'Stainless' 'Googlebot' 'Outlook 2007' 'IceDragon']
================================================================================
====================
```

```
 country_destination:
 number of unique value for each column 12 ,
 unique values is ['NDF' 'US' 'other' 'FR' 'CA' 'GB' 'ES' 'IT' 'PT' 'NL' 'DE'
'AU']
================================================================================
====================
```

[10]:
```
df_train.isnull().sum()#check for null values
```

[10]:
```
id                              0
date_account_created            0
timestamp_first_active          0
date_first_booking         124543
gender                          0
age                         87990
signup_method                   0
signup_flow                     0
language                        0
affiliate_channel               0
affiliate_provider              0
first_affiliate_tracked      6065
signup_app                      0
first_device_type               0
first_browser                   0
country_destination             0
dtype: int64
```

[11]:
```
df_train.duplicated().sum()#check duplicates
```

[11]:
```
0
```

Convert the columns to datetime

[12]:
```
df_train['timestamp_first_active'] = df_train['timestamp_first_active'].
 ↪astype(str)
df_train['timestamp_first_active'] = pd.
 ↪to_datetime(df_train['timestamp_first_active'], format='%Y%m%d%H%M%S')
print(df_train['timestamp_first_active'].head())
```

```
0   2009-03-19 04:32:55
1   2009-05-23 17:48:09
2   2009-06-09 23:12:47
3   2009-10-31 06:01:29
4   2009-12-08 06:11:05
Name: timestamp_first_active, dtype: datetime64[ns]
```

[13]:
```
# Convert the columns to datetime
df_train['date_account_created'] = pd.
 ↪to_datetime(df_train['date_account_created'])
```

```
df_train['date_first_booking'] = pd.to_datetime(df_train['date_first_booking'])
```

[121]:
```
df_train['date_first_booking'].value_counts()
```

[121]:
```
date_first_booking
2014-05-22    248
2014-06-11    231
2014-06-24    226
2014-05-21    225
2014-06-10    223
              ...
2010-01-31      1
2010-02-09      1
2010-06-10      1
2010-02-04      1
2015-06-28      1
Name: count, Length: 1976, dtype: int64
```

[15]:
```
df_train['date_first_booking'] = df_train['date_first_booking'].
↪fillna(df_train['date_first_booking'].median())
```

3 Extracting After Transformation

[16]:
```
#  day, month, and year from the datetime columns
df_train['day_account_created'] = df_train['date_account_created'].dt.day
df_train['month_account_created'] = df_train['date_account_created'].dt.month
df_train['year_account_created'] = df_train['date_account_created'].dt.year

df_train['day_first_active'] = df_train['timestamp_first_active'].dt.day
df_train['month_first_active'] = df_train['timestamp_first_active'].dt.month
df_train['year_first_active'] = df_train['timestamp_first_active'].dt.year
# df_train['hour_first_active'] = df_train['timestamp_first_active'].dt.hour

df_train['day_first_booking'] = df_train['date_first_booking'].dt.day
df_train['month_first_booking'] = df_train['date_first_booking'].dt.month
df_train['year_first_booking'] = df_train['date_first_booking'].dt.year
```

[17]:
```
df_train.drop(columns=['date_account_created', 'timestamp_first_active'],␣
↪inplace=True)
```

4 Irrelevant Features

1-replace

2-drop

[18]:
```
df_train['gender'] = df_train['gender'].replace('-unknown-', 'Unknown')
df_train['first_browser'] = df_train['first_browser'].replace('-unknown-',␣
↪'Unknown')
```

[19]:
```
df_train['age'].value_counts()
```

[19]:
```
age
30.0      6124
31.0      6016
29.0      5963
28.0      5939
32.0      5855
          …
1925.0       1
1935.0       1
1933.0       1
112.0        1
1926.0       1
Name: count, Length: 127, dtype: int64
```

[20]:
```
print("Mean Age:", df_train['age'].mean())
print("Median Age:", df_train['age'].median())
```

```
Mean Age: 49.66833517985669
Median Age: 34.0
```

Replaced ages (outside 18–100) with the mean age

[21]:
```
min_age = 18
max_age = 100

df_train['age'] = df_train['age'].apply(lambda x: x if min_age <= x <= max_age␣
↪else df_train['age'].median())
df_train['age']
```

[21]:
```
0         34.0
1         38.0
2         56.0
3         42.0
4         41.0
          …
213446    32.0
213447    34.0
213448    32.0
213449    34.0
213450    34.0
Name: age, Length: 213451, dtype: float64
```

Categorize the 'age' column into age groups using bins

[22]:
```
bins = [18, 25, 35, 45, 55, 65, 75, 85, 100]
labels = ['18-25', '26-35', '36-45', '46-55', '56-65', '66-75', '76-85',␣
 ↪'86-100']

df_train['age_group'] = pd.cut(df_train['age'], bins=bins, labels=labels,␣
 ↪right=False)
print(df_train['age_group'].value_counts())
```

```
age_group
26-35     146187
36-45      30759
46-55      14521
18-25      10677
56-65       7599
66-75       2902
76-85        481
86-100       299
Name: count, dtype: int64
```

[23]:
```
df_train.drop(columns=['age'], inplace=True)
```

5 check for null values

[24]:
```
# df_train['age_group'].isnull().sum()
df_train.isnull().sum()
```

[24]:
```
id                          0
date_first_booking          0
gender                      0
signup_method               0
signup_flow                 0
language                    0
affiliate_channel           0
affiliate_provider          0
first_affiliate_tracked    6065
signup_app                  0
first_device_type           0
first_browser               0
country_destination         0
day_account_created         0
month_account_created       0
year_account_created        0
day_first_active            0
month_first_active          0
year_first_active           0
day_first_booking           0
month_first_booking         0
```

```
year_first_booking            0
age_group                    26
dtype: int64
```

5.0.1 Handling Missing Data with SimpleImputer

[25]:
```
from sklearn.impute import SimpleImputer

# Create imputers
mode_imputer = SimpleImputer(strategy='most_frequent')

categorical_columns = [ 'first_affiliate_tracked','age_group']
df_train[categorical_columns] = mode_imputer.
 ↪fit_transform(df_train[categorical_columns])
```

[26]:
```
df_train.isnull().any().any()
```

[26]:
```
False
```

Finally No missing Values

Dtypes After Transform

[27]:
```
df_train.dtypes
```

[27]:
```
id                              object
date_first_booking      datetime64[ns]
gender                          object
signup_method                   object
signup_flow                      int64
language                        object
affiliate_channel               object
affiliate_provider              object
first_affiliate_tracked         object
signup_app                      object
first_device_type               object
first_browser                   object
country_destination             object
day_account_created              int32
month_account_created            int32
year_account_created             int32
day_first_active                 int32
month_first_active               int32
year_first_active                int32
day_first_booking                int32
month_first_booking              int32
year_first_booking               int32
age_group                       object
```

```
dtype: object
```

[28]:
```
for col in df_train.columns:
    print(f' {col}:  \n number of unique value for each column {df_train[col].
 ↪nunique()} , \n unique values is {df_train[col].unique()}')
    print('='*100)
```

```
 id:
 number of unique value for each column 213451 ,
 unique values is ['gxn3p5htnn' '820tgsjxq7' '4ft3gnwmtx' … '6o3arsjbb4'
'jh95kwisub'
 'nw9fwlyb5f']
====================================================================================
====================
 date_first_booking:
 number of unique value for each column 1976 ,
 unique values is <DatetimeArray>
['2013-09-11 00:00:00', '2010-08-02 00:00:00', '2012-09-08 00:00:00',
 '2010-02-18 00:00:00', '2010-01-02 00:00:00', '2010-01-05 00:00:00',
 '2010-01-13 00:00:00', '2010-07-29 00:00:00', '2010-01-04 00:00:00',
 '2010-01-06 00:00:00',
 …
 '2015-06-06 00:00:00', '2015-06-09 00:00:00', '2015-06-22 00:00:00',
 '2015-06-23 00:00:00', '2015-06-18 00:00:00', '2015-06-14 00:00:00',
 '2015-06-26 00:00:00', '2015-06-25 00:00:00', '2015-06-29 00:00:00',
 '2015-06-28 00:00:00']
Length: 1976, dtype: datetime64[ns]
====================================================================================
====================
 gender:
 number of unique value for each column 4 ,
 unique values is ['Unknown' 'MALE' 'FEMALE' 'OTHER']
====================================================================================
====================
 signup_method:
 number of unique value for each column 3 ,
 unique values is ['facebook' 'basic' 'google']
====================================================================================
====================
 signup_flow:
 number of unique value for each column 17 ,
 unique values is [ 0  3  2  1 24  8  6  5 10 25 12  4 16 15 20 21 23]
====================================================================================
====================
 language:
 number of unique value for each column 25 ,
 unique values is ['en' 'fr' 'de' 'es' 'it' 'pt' 'zh' 'ko' 'ja' 'ru' 'pl' 'el'
'sv' 'nl'
```

```
 'hu' 'da' 'id' 'fi' 'no' 'tr' 'th' 'cs' 'hr' 'ca' 'is']
================================================================================
====================
 affiliate_channel:
 number of unique value for each column 8 ,
 unique values is ['direct' 'seo' 'other' 'sem-non-brand' 'content' 'sem-brand'
 'remarketing' 'api']
================================================================================
====================
 affiliate_provider:
 number of unique value for each column 18 ,
 unique values is ['direct' 'google' 'other' 'craigslist' 'facebook' 'vast'
'bing' 'meetup'
 'facebook-open-graph' 'email-marketing' 'yahoo' 'padmapper' 'gsp' 'wayn'
 'naver' 'baidu' 'yandex' 'daum']
================================================================================
====================
 first_affiliate_tracked:
 number of unique value for each column 7 ,
 unique values is ['untracked' 'omg' 'linked' 'tracked-other' 'product'
'marketing'
 'local ops']
================================================================================
====================
 signup_app:
 number of unique value for each column 4 ,
 unique values is ['Web' 'Moweb' 'iOS' 'Android']
================================================================================
====================
 first_device_type:
 number of unique value for each column 9 ,
 unique values is ['Mac Desktop' 'Windows Desktop' 'iPhone' 'Other/Unknown'
 'Desktop (Other)' 'Android Tablet' 'iPad' 'Android Phone'
 'SmartPhone (Other)']
================================================================================
====================
 first_browser:
 number of unique value for each column 52 ,
 unique values is ['Chrome' 'IE' 'Firefox' 'Safari' 'Unknown' 'Mobile Safari'
 'Chrome Mobile' 'RockMelt' 'Chromium' 'Android Browser' 'AOL Explorer'
 'Palm Pre web browser' 'Mobile Firefox' 'Opera' 'TenFourFox' 'IE Mobile'
 'Apple Mail' 'Silk' 'Camino' 'Arora' 'BlackBerry Browser' 'SeaMonkey'
 'Iron' 'Sogou Explorer' 'IceWeasel' 'Opera Mini' 'SiteKiosk' 'Maxthon'
 'Kindle Browser' 'CoolNovo' 'Conkeror' 'wOSBrowser' 'Google Earth'
 'Crazy Browser' 'Mozilla' 'OmniWeb' 'PS Vita browser' 'NetNewsWire'
 'CometBird' 'Comodo Dragon' 'Flock' 'Pale Moon' 'Avant Browser'
 'Opera Mobile' 'Yandex.Browser' 'TheWorld Browser' 'SlimBrowser' 'Epic'
 'Stainless' 'Googlebot' 'Outlook 2007' 'IceDragon']
```

```
================================================================================
====================
 country_destination:
 number of unique value for each column 12 ,
 unique values is ['NDF' 'US' 'other' 'FR' 'CA' 'GB' 'ES' 'IT' 'PT' 'NL' 'DE'
'AU']
================================================================================
====================
 day_account_created:
 number of unique value for each column 31 ,
 unique values is [28 25  5 14  1  2  3  4  7  8 10 11 12 13 15 16 19 21 23 24
26 27 29 30
 31  6  9 17 18 20 22]
================================================================================
====================
 month_account_created:
 number of unique value for each column 12 ,
 unique values is [ 6  5  9 12  1  2  3  4 11  7  8 10]
================================================================================
====================
 year_account_created:
 number of unique value for each column 5 ,
 unique values is [2010 2011 2014 2012 2013]
================================================================================
====================
 day_first_active:
 number of unique value for each column 31 ,
 unique values is [19 23  9 31  8  1  2  3  4  5  7 10 11 12 13 14 15 16 21 24
25 26 27 28
 29 30  6 17 18 20 22]
================================================================================
====================
 month_first_active:
 number of unique value for each column 12 ,
 unique values is [ 3  5  6 10 12  1  2  4  7  8  9 11]
================================================================================
====================
 year_first_active:
 number of unique value for each column 6 ,
 unique values is [2009 2010 2011 2012 2013 2014]
================================================================================
====================
 day_first_booking:
 number of unique value for each column 31 ,
 unique values is [11  2  8 18  5 13 29  4  6  9 10 15 22 19 24 21  3 27 25 26
30 28 31 12
 20  7  1 16 17 23 14]
================================================================================
```

```
====================
 month_first_booking:
 number of unique value for each column 12 ,
 unique values is [ 9  8  2  1  7 12  3  6  4  5 10 11]
================================================================================
====================
 year_first_booking:
 number of unique value for each column 6 ,
 unique values is [2013 2010 2012 2011 2014 2015]
================================================================================
====================
 age_group:
 number of unique value for each column 8 ,
 unique values is ['26-35' '36-45' '56-65' '46-55' '18-25' '66-75' '86-100'
'76-85']
================================================================================
====================
```

6 EDA (Exploratory Data Analysis)

1-Univariate Analysis

2-Bivariate Analysis

3-Multivariate Analysis

[29]:
```
df_train.describe().T
```

[29]:

	count	mean	\
date_first_booking	213451	2013-08-13 13:37:17.487760896	
signup_flow	213451.0	3.267387	
day_account_created	213451.0	15.86923	
month_account_created	213451.0	6.022459	
year_account_created	213451.0	2013.023846	
day_first_active	213451.0	15.869071	
month_first_active	213451.0	6.022385	
year_first_active	213451.0	2013.023218	
day_first_booking	213451.0	12.934561	
month_first_booking	213451.0	7.796487	
year_first_booking	213451.0	2013.017845	

	min	25%	\
date_first_booking	2010-01-02 00:00:00	2013-09-11 00:00:00	
signup_flow	0.0	0.0	
day_account_created	1.0	8.0	
month_account_created	1.0	3.0	
year_account_created	2010.0	2012.0	
day_first_active	1.0	8.0	

```
month_first_active                    1.0                  3.0
year_first_active                  2009.0               2012.0
day_first_booking                     1.0                 11.0
month_first_booking                   1.0                  7.0
year_first_booking                 2010.0               2013.0

                                      50%                  75%  \
date_first_booking     2013-09-11 00:00:00  2013-09-11 00:00:00
signup_flow                           0.0                  0.0
day_account_created                  16.0                 23.0
month_account_created                 6.0                  9.0
year_account_created               2013.0               2014.0
day_first_active                     16.0                 23.0
month_first_active                    6.0                  9.0
year_first_active                  2013.0               2014.0
day_first_booking                    11.0                 13.0
month_first_booking                   9.0                  9.0
year_first_booking                 2013.0               2013.0

                                      max       std
date_first_booking     2015-06-29 00:00:00       NaN
signup_flow                          25.0  7.637707
day_account_created                  31.0  8.740107
month_account_created                12.0   3.23669
year_account_created               2014.0  0.938489
day_first_active                     31.0  8.739582
month_first_active                   12.0  3.236501
year_first_active                  2014.0  0.939039
day_first_booking                    31.0  6.079097
month_first_booking                  12.0  2.498683
year_first_booking                 2015.0  0.656313
```

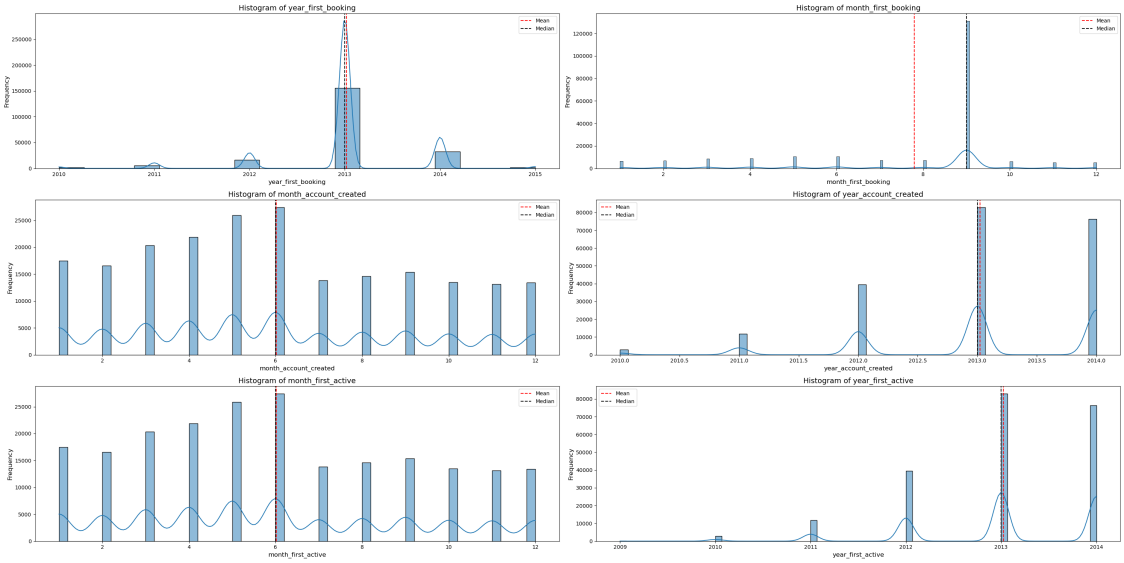
[30]:
```
df_train.columns
```

[30]:
```
Index(['id', 'date_first_booking', 'gender', 'signup_method', 'signup_flow',
       'language', 'affiliate_channel', 'affiliate_provider',
       'first_affiliate_tracked', 'signup_app', 'first_device_type',
       'first_browser', 'country_destination', 'day_account_created',
       'month_account_created', 'year_account_created', 'day_first_active',
       'month_first_active', 'year_first_active', 'day_first_booking',
       'month_first_booking', 'year_first_booking', 'age_group'],
      dtype='object')
```

1-univariate analysis

[31]:
```
numeric_columns = ['year_first_booking', 'month_first_booking',
       'month_account_created', 'year_account_created',
       'month_first_active', 'year_first_active']
```

```
plt.figure(figsize=(30, 15))

for i, col in enumerate(numeric_columns):
    plt.subplot(3,2, i + 1)
    ax = sns.histplot(df_train[col], kde=True)
    ax.axvline(df_train[col].mean(), color='red', linestyle='--', label='Mean')
    ax.axvline(df_train[col].median(), color='black', linestyle='--',␣
 ↪label='Median')
    plt.title(f'Histogram of {col}', fontsize=14)
    plt.xlabel(col, fontsize=12)
    plt.ylabel('Frequency', fontsize=12)
    plt.legend()
    plt.tight_layout()

plt.show()
```



1. **Year of First Booking**:
   - Most bookings happened in **2013** (mean and median both around **2013**).
2. **Month of First Booking**:
   - Bookings peaked in **(9)**.
3. **Month Account Created**:
   - Most accounts were created in **June (6)** and **July (7)**.
4. **Year Account Created**:
   - Majority of accounts were created in **2013**.
5. **First Active (Month & Year)**:
   - Accounts became active mostly in **June (6)** and **2013**.

Key Insight:

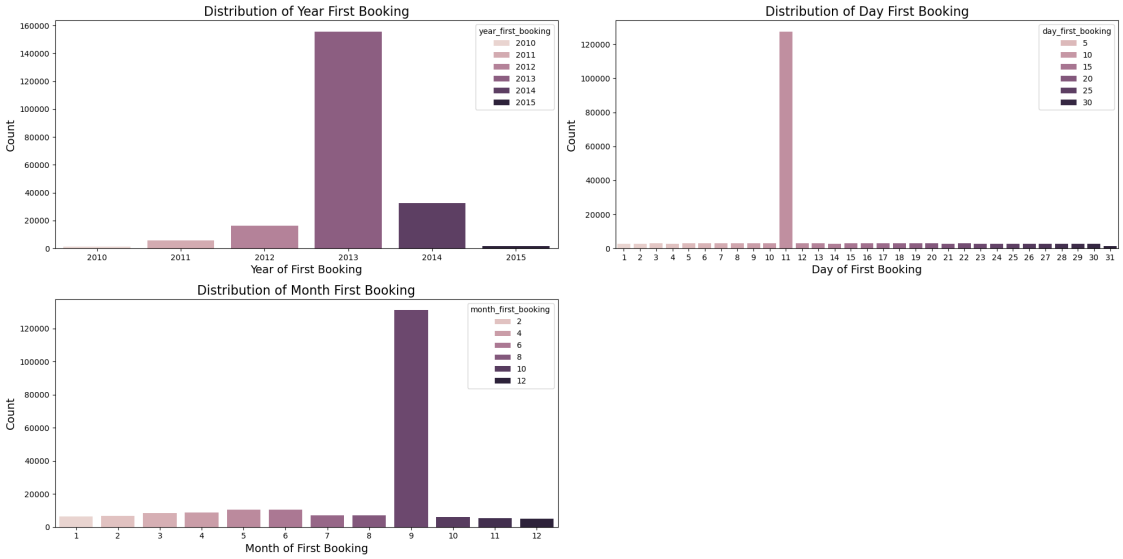
Most user activity happened in mid-year (June & July) and during 2013, showing seasonal patterns.

[32]:
```
# Count values for 'month_first_booking' and 'year_first_booking'
month_counts = df_train['month_first_booking'].value_counts().
 ↪sort_index(ascending=False)
year_counts = df_train['year_first_booking'].value_counts().
 ↪sort_index(ascending=False)

# Print the counts in descending order
print("Counts for 'month_first_booking' in descending order:")
print(month_counts.sort_values(ascending=False))
print("\nCounts for 'year_first_booking' in descending order:")
print(year_counts.sort_values(ascending=False))
```

```
Counts for 'month_first_booking' in descending order:
month_first_booking
9     131141
6      10509
5      10478
4       8813
3       8391
7       7249
8       7055
2       6790
1       6491
10      6184
11      5264
12      5086
Name: count, dtype: int64

Counts for 'year_first_booking' in descending order:
year_first_booking
2013    155802
2014     32419
2012     16241
2011      5738
2015      1772
2010      1479
Name: count, dtype: int64
```

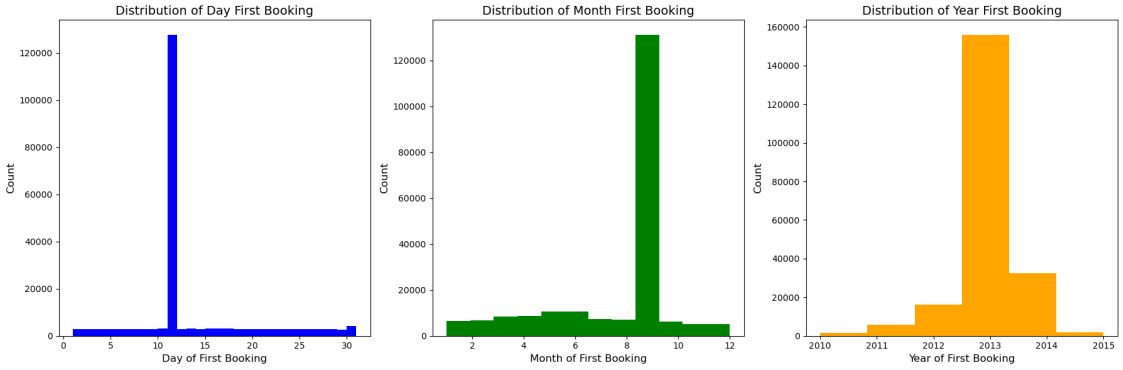
[33]:
```
fig, axes = plt.subplots(2, 2, figsize=(20, 10))

# Visualization for 'year_first_booking'
sns.countplot(x='year_first_booking', data=df_train,hue='year_first_booking',␣
 ↪ax=axes[0, 0])
axes[0, 0].set_title('Distribution of Year First Booking', fontsize=16)
axes[0, 0].set_xlabel('Year of First Booking', fontsize=14)
axes[0, 0].set_ylabel('Count', fontsize=14)
```

```
# Visualization for 'day_first_booking'
sns.countplot(x='day_first_booking', data=df_train,hue='day_first_booking',␣
 ↪ax=axes[0, 1])
axes[0, 1].set_title('Distribution of Day First Booking', fontsize=16)
axes[0, 1].set_xlabel('Day of First Booking', fontsize=14)
axes[0, 1].set_ylabel('Count', fontsize=14)

# Visualization for 'month_first_booking'
sns.countplot(x='month_first_booking', data=df_train,␣
 ↪hue='month_first_booking', ax=axes[1, 0])
axes[1, 0].set_title('Distribution of Month First Booking', fontsize=16)
axes[1, 0].set_xlabel('Month of First Booking', fontsize=14)
axes[1, 0].set_ylabel('Count', fontsize=14)

axes[1, 1].axis('off')

plt.tight_layout()
plt.show()
```



[34]:
```
fig, axes = plt.subplots(1, 3, figsize=(18, 6))

df_train['day_first_booking'].plot(kind='hist', ax=axes[0], bins=30,␣
 ↪color='blue')
axes[0].set_title('Distribution of Day First Booking', fontsize=14)
axes[0].set_xlabel('Day of First Booking', fontsize=12)
axes[0].set_ylabel('Count', fontsize=12)
```

```
df_train['month_first_booking'].plot(kind='hist', ax=axes[1], bins=12,␣
 ↪color='green')
axes[1].set_title('Distribution of Month First Booking', fontsize=14)
axes[1].set_xlabel('Month of First Booking', fontsize=12)
axes[1].set_ylabel('Count', fontsize=12)

df_train['year_first_booking'].plot(kind='hist', ax=axes[2], bins=6,␣
 ↪color='orange')
axes[2].set_title('Distribution of Year First Booking', fontsize=14)
axes[2].set_xlabel('Year of First Booking', fontsize=12)
axes[2].set_ylabel('Count', fontsize=12)

plt.tight_layout()

plt.show()
```

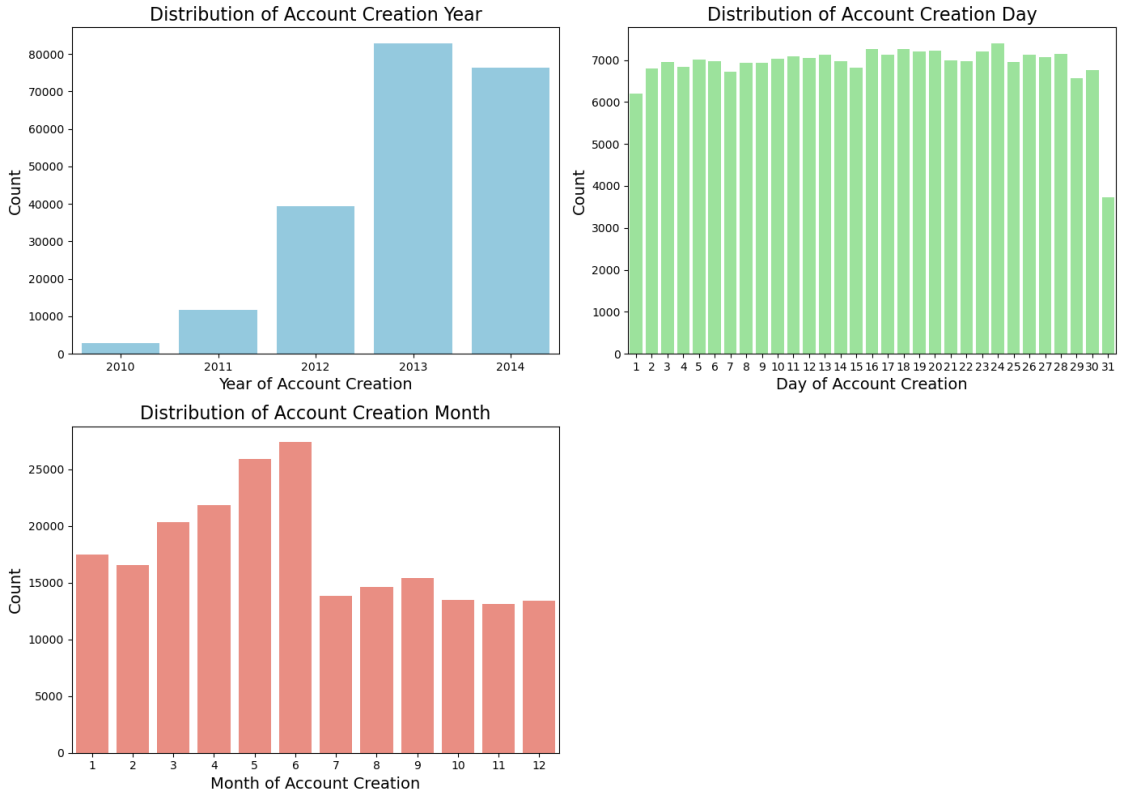


This is similar to the histogram I plotted earlier 1. **Day of First Booking (Left Chart)**: - Most first bookings happened on **one specific day** (likely the 11th), while very few bookings happened on other days.

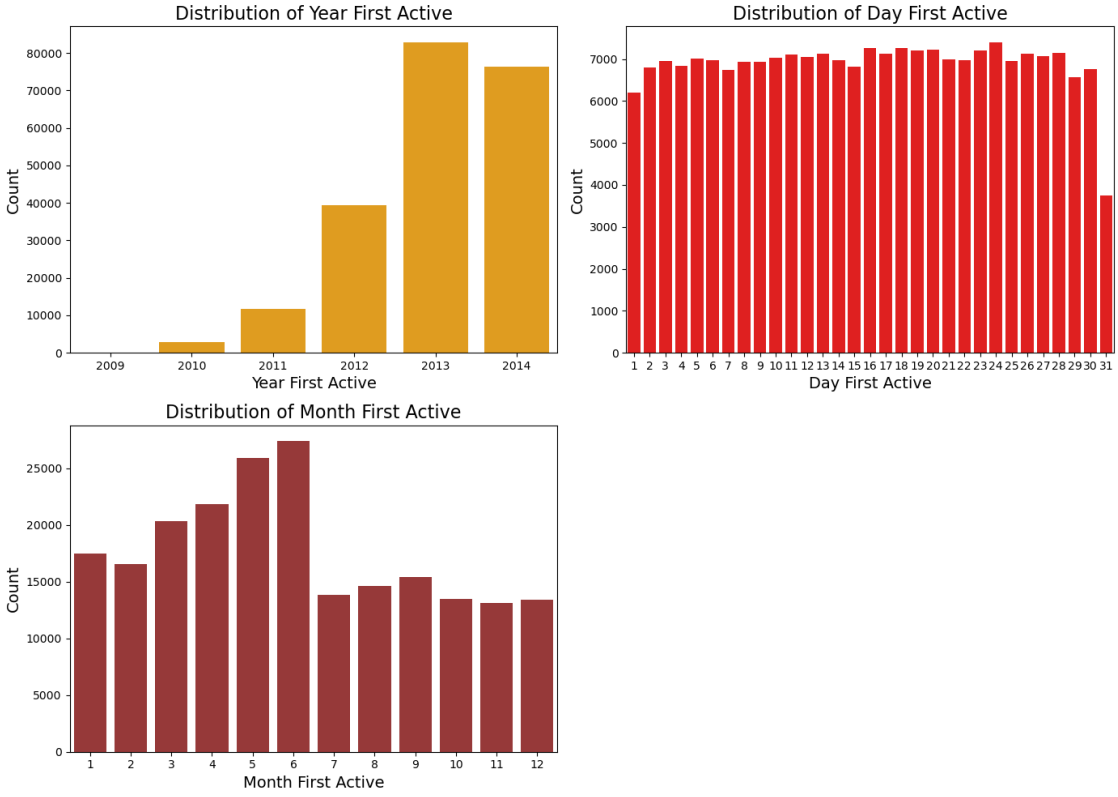
2. **Month of First Booking (Middle Chart)**:
   - A **specific month** (september) had the highest number of first bookings compared to other months.
3. **Year of First Booking (Right Chart)**:
   - Most first bookings were made in **one year** (likely 2013), and bookings in other years were much fewer.

[35]:
```
fig, axes = plt.subplots(2, 2, figsize=(14, 10))

# Visualization for 'year_account_created'
sns.countplot(x='year_account_created', data=df_train, color='skyblue',␣
 ↪ax=axes[0, 0])
axes[0, 0].set_title('Distribution of Account Creation Year', fontsize=16)
axes[0, 0].set_xlabel('Year of Account Creation', fontsize=14)
```

```
axes[0, 0].set_ylabel('Count', fontsize=14)

# Visualization for 'day_account_created'
sns.countplot(x='day_account_created', data=df_train, color='lightgreen',␣
 ↪ax=axes[0, 1])
axes[0, 1].set_title('Distribution of Account Creation Day', fontsize=16)
axes[0, 1].set_xlabel('Day of Account Creation', fontsize=14)
axes[0, 1].set_ylabel('Count', fontsize=14)

# Visualization for 'month_account_created'
sns.countplot(x='month_account_created', data=df_train, color='salmon',␣
 ↪ax=axes[1, 0])
axes[1, 0].set_title('Distribution of Account Creation Month', fontsize=16)
axes[1, 0].set_xlabel('Month of Account Creation', fontsize=14)
axes[1, 0].set_ylabel('Count', fontsize=14)

axes[1, 1].axis('off')

plt.tight_layout()
plt.show()
```

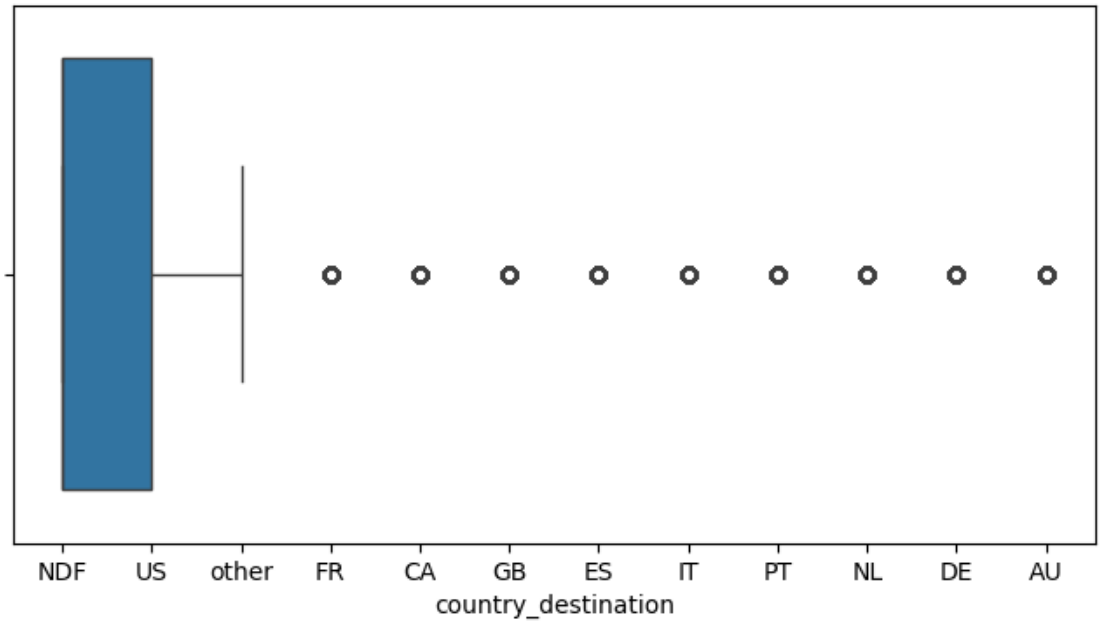


[36]:
```
fig, axes = plt.subplots(2, 2, figsize=(14, 10))

# Visualization for 'year_first_active'
sns.countplot(x='year_first_active', data=df_train, color='orange', ax=axes[0,␣
 ↪0])
axes[0, 0].set_title('Distribution of Year First Active', fontsize=16)
axes[0, 0].set_xlabel('Year First Active', fontsize=14)
axes[0, 0].set_ylabel('Count', fontsize=14)

# Visualization for 'day_first_active'
sns.countplot(x='day_first_active', data=df_train, color='red', ax=axes[0, 1])
axes[0, 1].set_title('Distribution of Day First Active', fontsize=16)
axes[0, 1].set_xlabel('Day First Active', fontsize=14)
axes[0, 1].set_ylabel('Count', fontsize=14)

# Visualization for 'month_first_active'
sns.countplot(x='month_first_active', data=df_train, color='brown', ax=axes[1,␣
 ↪0])
axes[1, 0].set_title('Distribution of Month First Active', fontsize=16)
axes[1, 0].set_xlabel('Month First Active', fontsize=14)
axes[1, 0].set_ylabel('Count', fontsize=14)


axes[1, 1].axis('off')

plt.tight_layout()
plt.show()
```



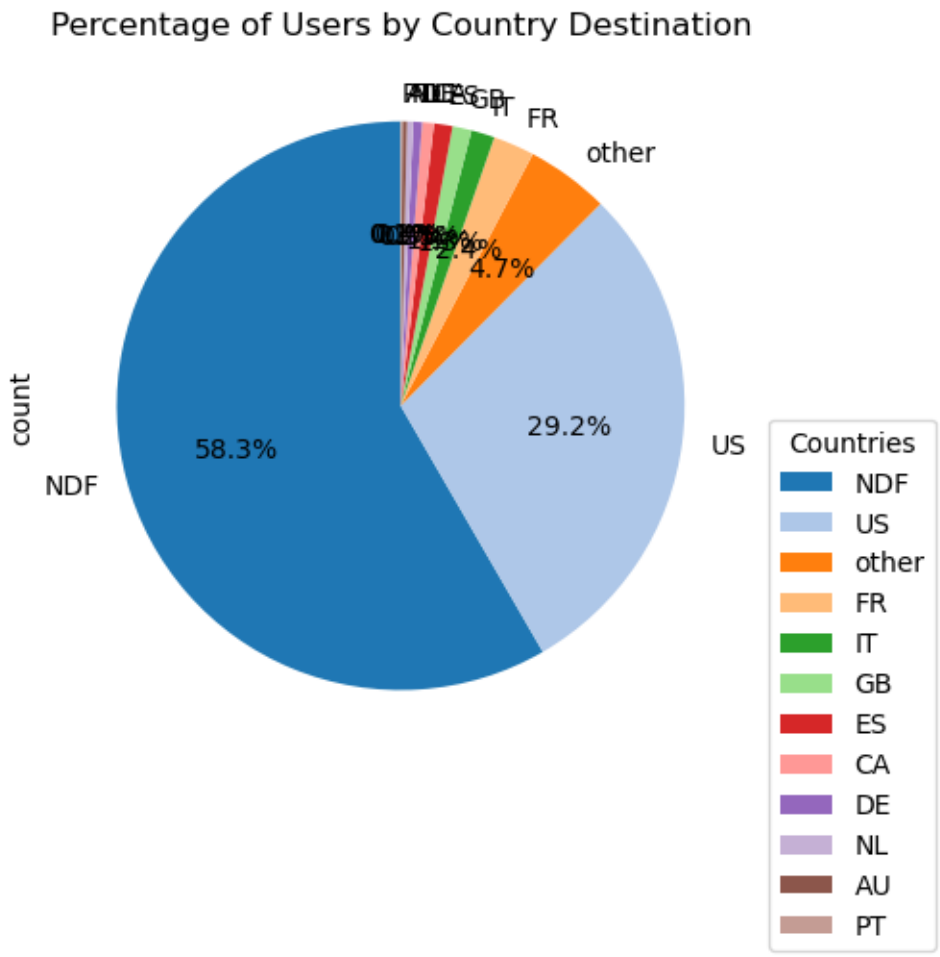
[37]:
```
plt.figure(figsize=(8, 4))
sns.boxplot(data=df_train,x='country_destination')
print(df_train['country_destination'].value_counts())
```

```
country_destination
NDF      124543
US        62376
other     10094
FR         5023
IT         2835
GB         2324
ES         2249
CA         1428
DE         1061
NL          762
AU          539
PT          217
Name: count, dtype: int64
```



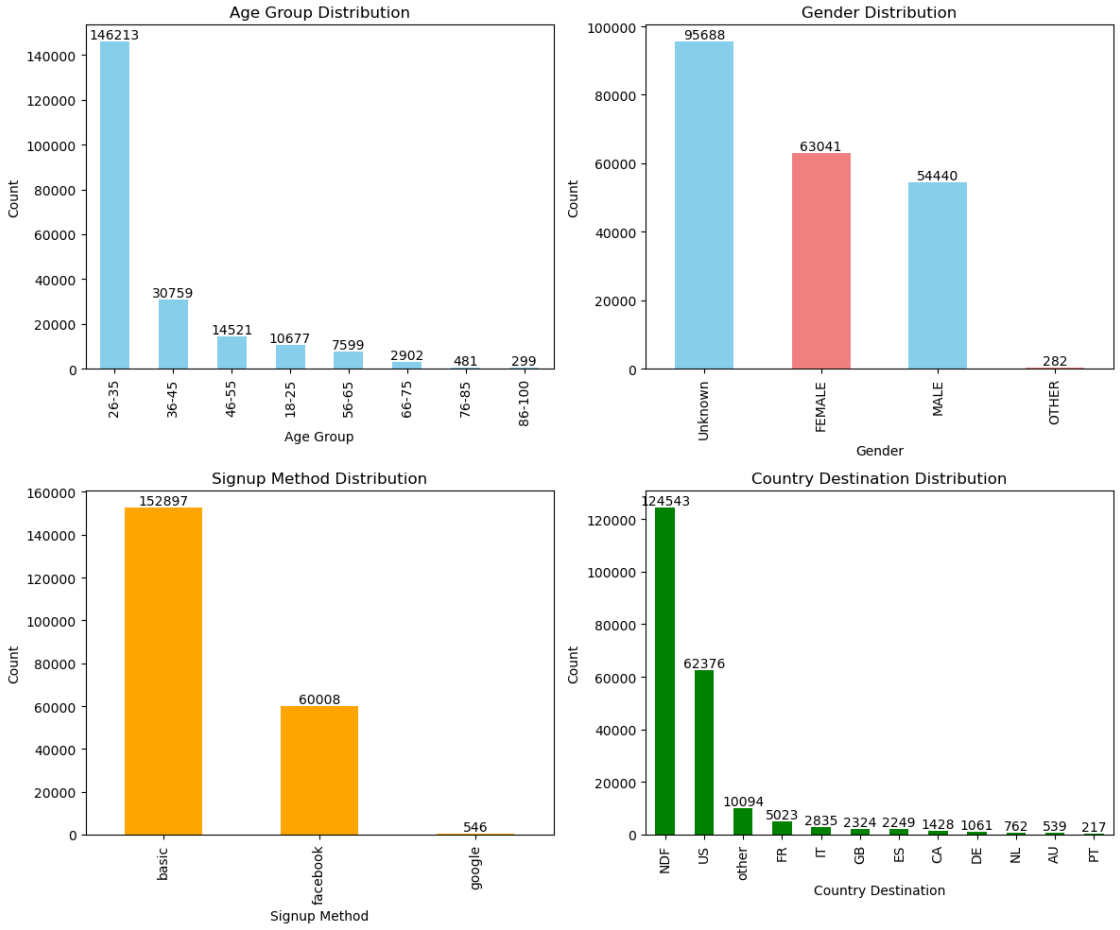
The boxplot shows that most users' destination is **"NDF"** (No Destination Found), followed by the **"US"** and **"Other"** destinations.
Destinations like **France (FR)**, **Canada (CA)**, and **Great Britain (GB)** have very few users, which shows a big imbalance in the data.

[38]:
```
df_train['country_destination'].value_counts().plot(
    kind='pie',
    autopct='%1.1f%%',
    startangle=90,
    colors=plt.cm.tab20.colors,
)
plt.legend( title="Countries", loc="best", bbox_to_anchor=(1, 0.5))
plt.title("Percentage of Users by Country Destination")
plt.show()
```



[39]:
```
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

df_train['age_group'].value_counts().plot(kind='bar', ax=axes[0, 0],␣
 ↪color='skyblue')
axes[0, 0].set_title('Age Group Distribution')
axes[0, 0].set_xlabel('Age Group')
axes[0, 0].set_ylabel('Count')
axes[0, 0].bar_label(axes[0, 0].containers[0])

df_train['gender'].value_counts().plot(kind='bar', ax=axes[0, 1],␣
 ↪color=['skyblue', 'lightcoral'])
axes[0, 1].set_title('Gender Distribution')
axes[0, 1].set_xlabel('Gender')
axes[0, 1].set_ylabel('Count')
axes[0, 1].bar_label(axes[0, 1].containers[0])
```

```
df_train['signup_method'].value_counts().plot(kind='bar', ax=axes[1, 0],␣
 ↪color='orange')
axes[1, 0].set_title('Signup Method Distribution')
axes[1, 0].set_xlabel('Signup Method')
axes[1, 0].set_ylabel('Count')
axes[1, 0].bar_label(axes[1, 0].containers[0])

df_train['country_destination'].value_counts().plot(kind='bar', ax=axes[1, 1],␣
 ↪color='green')
axes[1, 1].set_title('Country Destination Distribution')
axes[1, 1].set_xlabel('Country Destination')
axes[1, 1].set_ylabel('Count')
axes[1, 1].bar_label(axes[1, 1].containers[0])

plt.tight_layout()
plt.show()
```

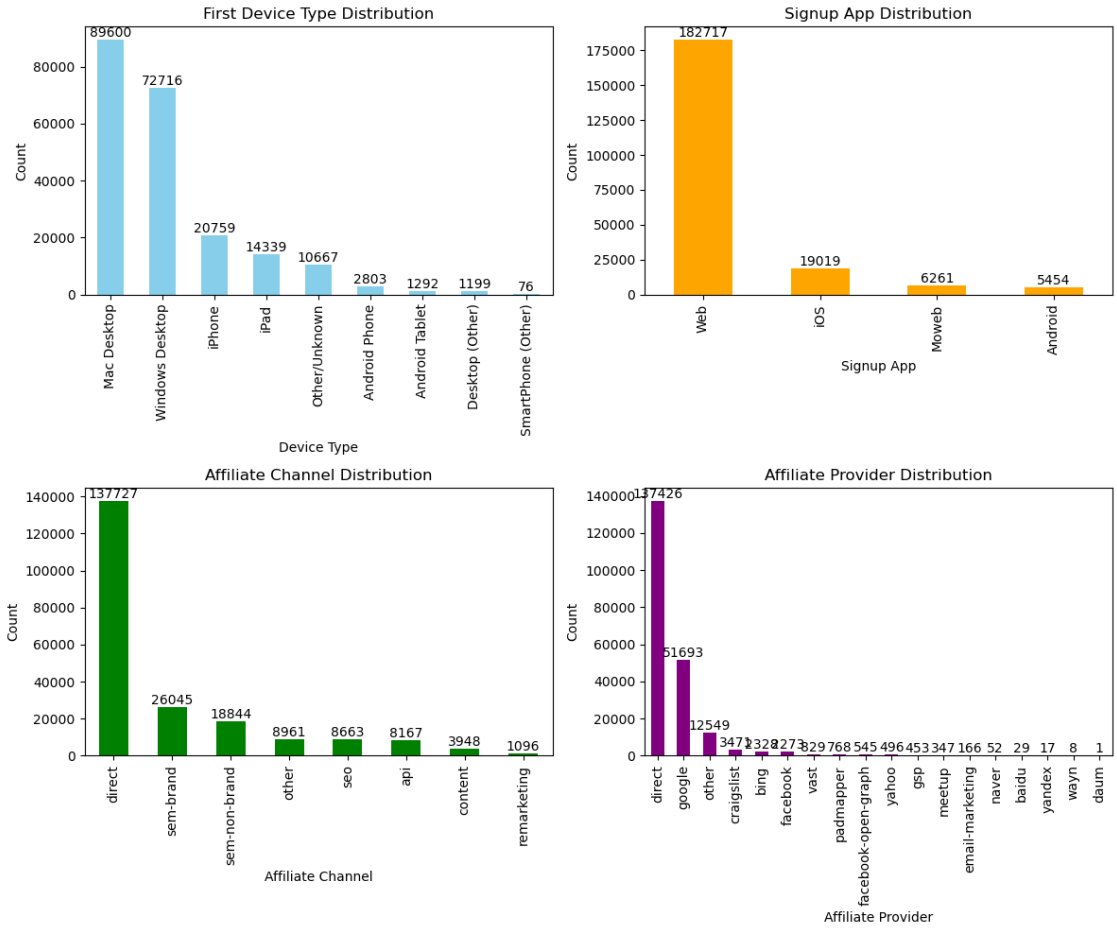


1. **Age Group:** Most users are aged **46-55**, followed by **26-35** and **36-45**. Few users are

younger than 25 or older than 55.

2. **Gender:** Many users have an **Unknown** gender. Among known genders, **Female** slightly outnumber **Male**.
3. **Signup Method:** Most users use **Basic** signup, followed by **Facebook**. Very few use **Google**.
4. **Country Destination: NDF (No Destination Found)** is the largest group. , **US** is the most popular, followed by **Other** and **France**.

[40]:
```
fig, axes = plt.subplots(2, 2, figsize=(12, 10))  # 3 rows and 2 columns

df_train['first_device_type'].value_counts().plot(kind='bar', ax=axes[0, 0],␣
 ↪color='skyblue')
axes[0, 0].set_title('First Device Type Distribution')
axes[0, 0].set_xlabel('Device Type')
axes[0, 0].set_ylabel('Count')
axes[0, 0].bar_label(axes[0, 0].containers[0])


df_train['signup_app'].value_counts().plot(kind='bar', ax=axes[0, 1],␣
 ↪color='orange')
axes[0, 1].set_title('Signup App Distribution')
axes[0, 1].set_xlabel('Signup App')
axes[0, 1].set_ylabel('Count')
axes[0, 1].bar_label(axes[0, 1].containers[0])

df_train['affiliate_channel'].value_counts().plot(kind='bar', ax=axes[1, 0],␣
 ↪color='green')
axes[1, 0].set_title('Affiliate Channel Distribution')
axes[1, 0].set_xlabel('Affiliate Channel')
axes[1, 0].set_ylabel('Count')
axes[1, 0].bar_label(axes[1, 0].containers[0])

df_train['affiliate_provider'].value_counts().plot(kind='bar', ax=axes[1, 1],␣
 ↪color='purple')
axes[1, 1].set_title('Affiliate Provider Distribution')
axes[1, 1].set_xlabel('Affiliate Provider')
axes[1, 1].set_ylabel('Count')
axes[1, 1].bar_label(axes[1, 1].containers[0])

plt.tight_layout()
plt.show()
```



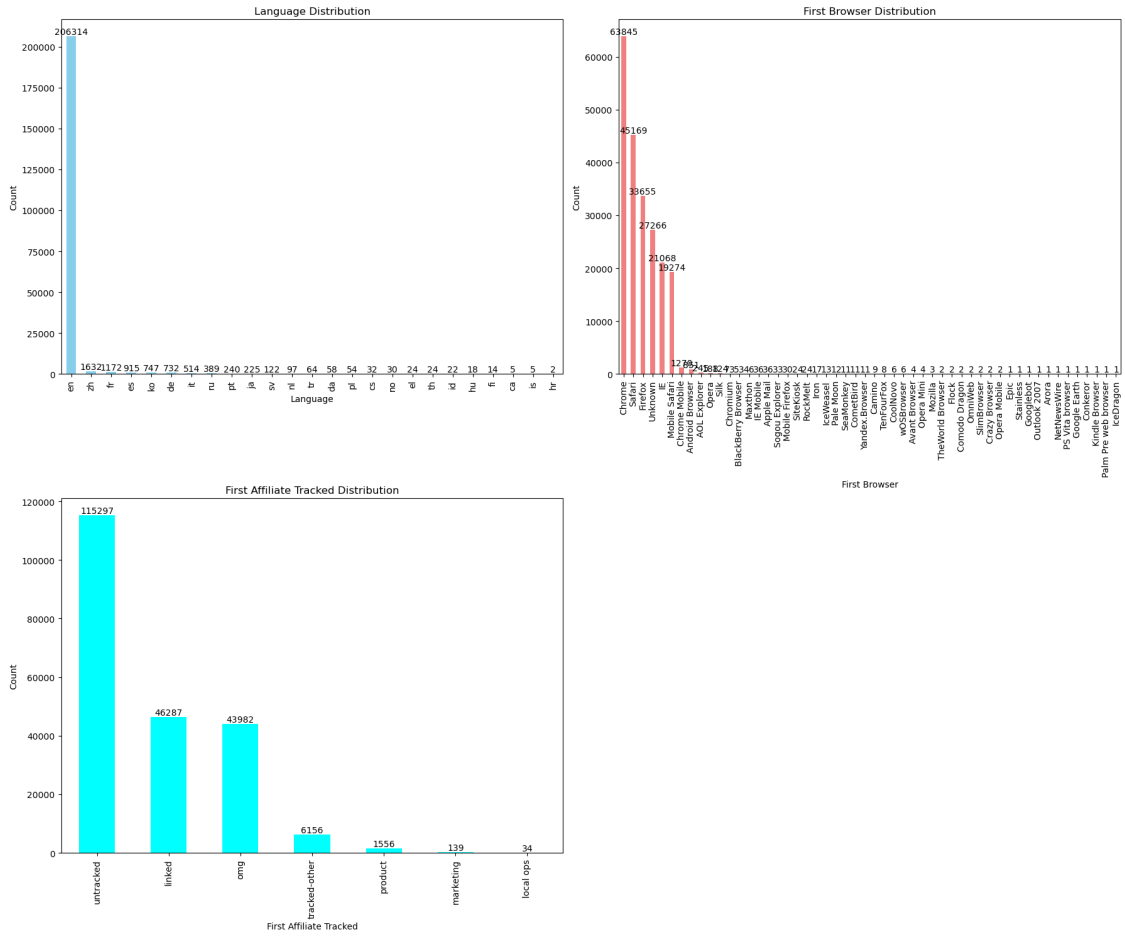
1. **First Device Type**:
   - Most people use **Mac Desktop** or **Windows Desktop**, followed by **iPhone** and **iPad**. Android devices are used much less.
2. **Signup App**:
   - Most signups happen on **Web**, followed by **iOS**. Very few use **Moweb** or **Android**.
3. **Affiliate Channel**:
   - **Direct** is the most common channel, with fewer users coming from **sem-brand** or **sem-non-brand**. Channels like **content** and **remarketing** are rarely used.
4. **Affiliate Provider**:
   - **Direct** and **Google** are the biggest providers

[41]:
```
df_train.columns
```

[41]:
```
Index(['id', 'date_first_booking', 'gender', 'signup_method', 'signup_flow',
       'language', 'affiliate_channel', 'affiliate_provider',
       'first_affiliate_tracked', 'signup_app', 'first_device_type',
       'first_browser', 'country_destination', 'day_account_created',
       'month_account_created', 'year_account_created', 'day_first_active',
```

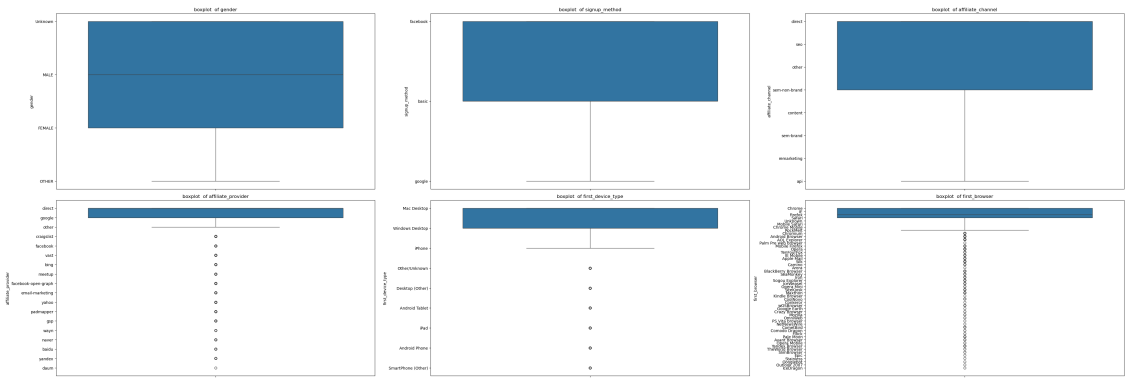
```
       'month_first_active', 'year_first_active', 'day_first_booking',
       'month_first_booking', 'year_first_booking', 'age_group'],
      dtype='object')
```

[42]:
```
fig, axes = plt.subplots(2, 2, figsize=(18, 15))  # Create 2 rows and 2 columns␣
 ↪of subplots

# Plot 1: Language Distribution
df_train['language'].value_counts().plot(kind='bar', ax=axes[0, 0],␣
 ↪color='skyblue')
axes[0, 0].set_title('Language Distribution')
axes[0, 0].set_xlabel('Language')
axes[0, 0].set_ylabel('Count')
axes[0, 0].bar_label(axes[0, 0].containers[0])

# Plot 2: First Browser Distribution
df_train['first_browser'].value_counts().plot(kind='bar', ax=axes[0, 1],␣
 ↪color='lightcoral')
axes[0, 1].set_title('First Browser Distribution')
axes[0, 1].set_xlabel('First Browser')
axes[0, 1].set_ylabel('Count')
axes[0, 1].bar_label(axes[0, 1].containers[0])

# Plot 3: First Affiliate Tracked Distribution
df_train['first_affiliate_tracked'].value_counts().plot(kind='bar', ax=axes[1,␣
 ↪0], color='cyan')
axes[1, 0].set_title('First Affiliate Tracked Distribution')
axes[1, 0].set_xlabel('First Affiliate Tracked')
axes[1, 0].set_ylabel('Count')
axes[1, 0].bar_label(axes[1, 0].containers[0])

# Hide the unused subplot
axes[1, 1].axis('off')  # Disable the fourth subplot

plt.tight_layout()
plt.show()
```



1. **Language Distribution**:
   - Most users (206,314) use the main language (likely English). Other languages like French, Spanish, and German have very few users.
2. **First Browser Distribution**:
   - **Chrome** is the most used browser (63,845), followed by **Safari** (45,169) and **Firefox** (33,655). Other browsers are rarely used.

[43]:
```
plt.figure(figsize=(40,20))
category_columns =['gender', 'signup_method', 'affiliate_channel',␣
 ↪'affiliate_provider', 'first_device_type',
       'first_browser']
for i , col in enumerate(category_columns):
    plt.subplot(3,3,i+1)
    ax=sns.boxplot(df_train[col])
    plt.title(f'boxplot  of {col}')
    plt.tight_layout()
plt.show()
```



1. **Gender:**
   Most users are categorized as "Unknown," followed by "Male" and "Female." A small portion is marked as "Other."
2. **Signup Method:**
   The majority of users registered using the "basic" method.
3. **Affiliate Channel:**
   Most users came through the "direct" channel, with fewer using "seo" or "other" channels.
4. **Affiliate Provider:**
   "Direct" is the dominant provider, followed by "Google" Other providers show minimal usage.
5. **First Device Type:**
   Desktops (Mac and Windows) are the most used devices
6. **First Browser:**
   There is significant diversity in browsers, but "Chrome" and "Safari" are the most commonly used.

2-univariate analysis

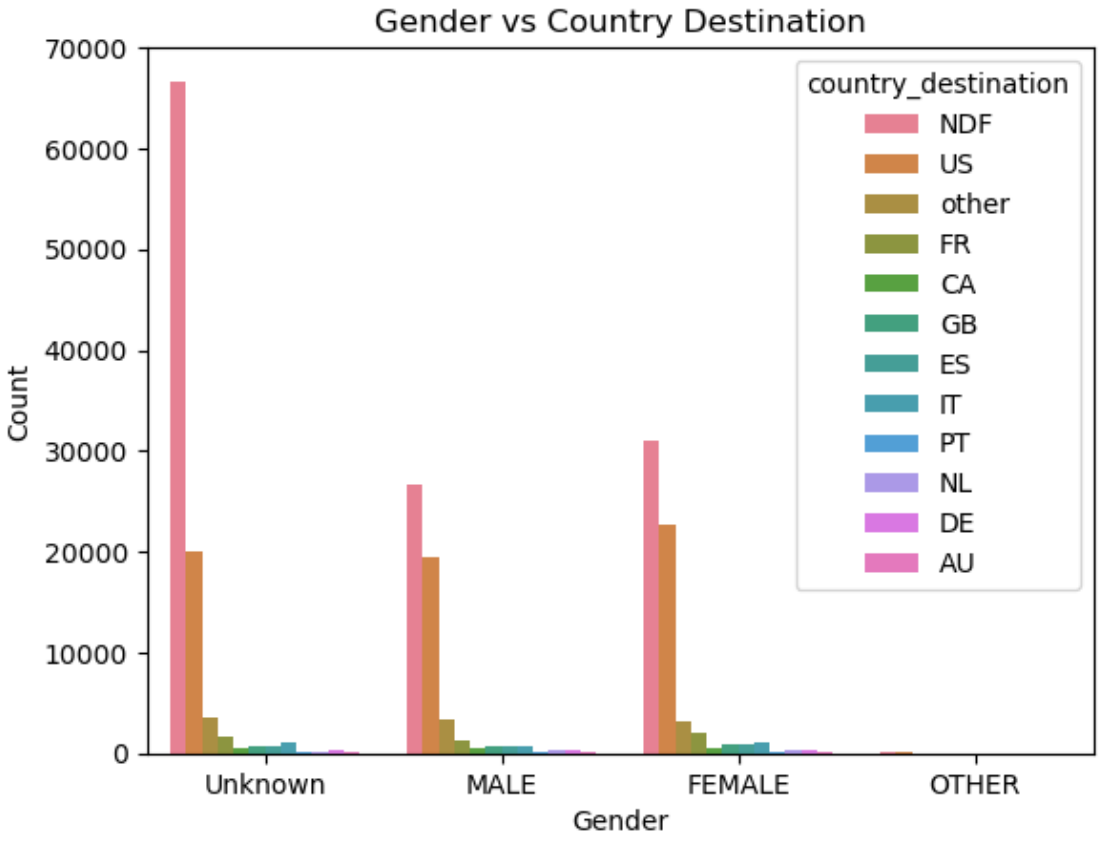
[44]:
```
gender_country = df_train.groupby('gender')['country_destination'].
  ↪value_counts().unstack()
gender_country
```

[44]:
```
country_destination    AU   CA   DE   ES    FR   GB    IT    NDF   NL  PT  \
gender
FEMALE                207  455  358  853  1962  881  1091  31048  254  78
MALE                  188  477  416  677  1335  682   699  26719  278  69
OTHER                   1    5    3    4    13    3     5    106    3   1
Unknown               143  491  284  715  1713  758  1040  66670  227  69

country_destination     US  other
gender
FEMALE               22694   3160
MALE                 19457   3443
```

```
OTHER                   116      22
Unknown                 20109    3469
```

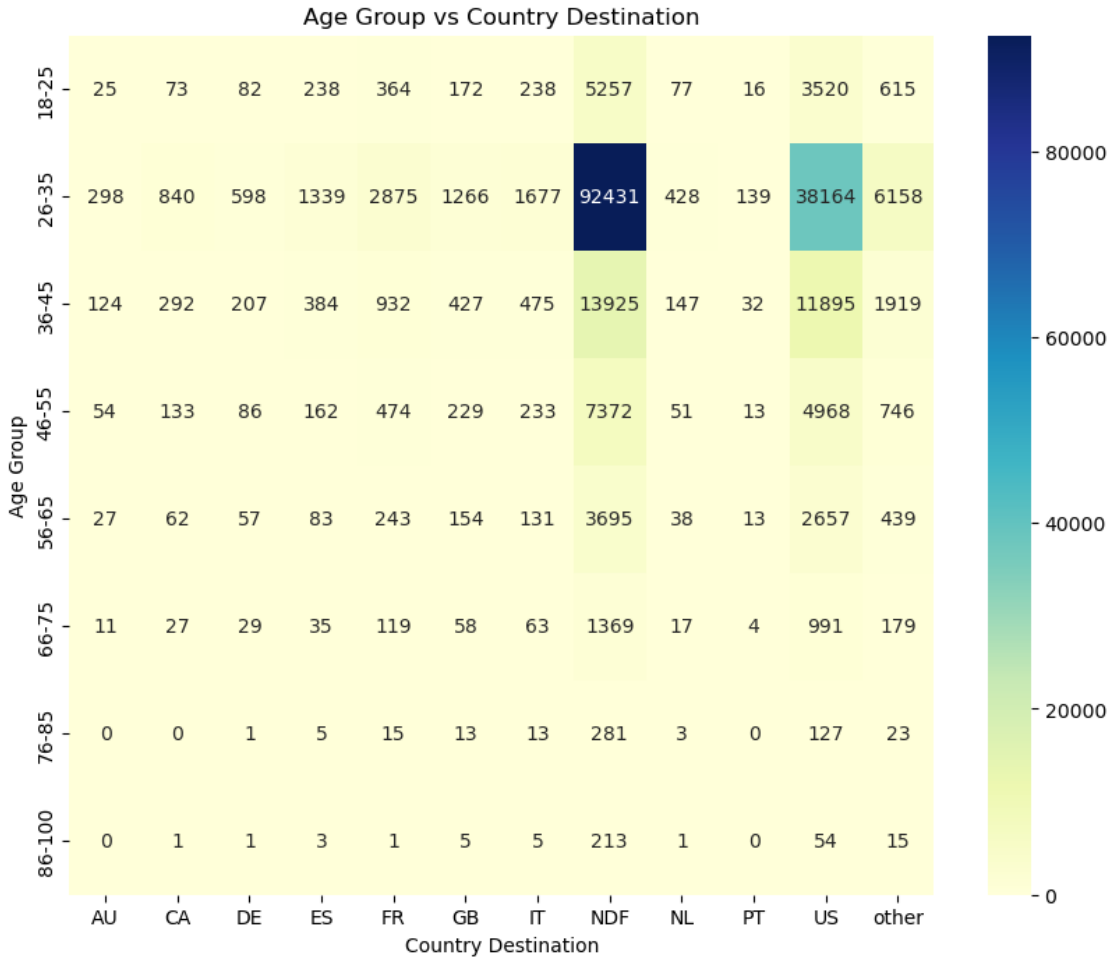
1. **NDF (No Destination Found)** has the highest number of people across all genders. Most people did not select a destination.
2. For **Females**, the most popular destination (after NDF) is the **US** (22,694), followed by **France** (1,964) and **Spain** (853).
3. For **Males**, the most popular destination (after NDF) is the **US** (19,457), followed by **France** (1,333) and **Germany** (677).
4. **Other genders** have very few numbers. The most common destination for them is the **US** (116).
5. For people with **Unknown gender**, the most common choice is **NDF** (66,670), followed by the **US** (20,109).
6. Overall, the **US** is the most popular destination for people who choose to travel.

[45]:
```
sns.countplot(data=df_train, x='gender', hue='country_destination')
plt.title('Gender vs Country Destination')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()
```



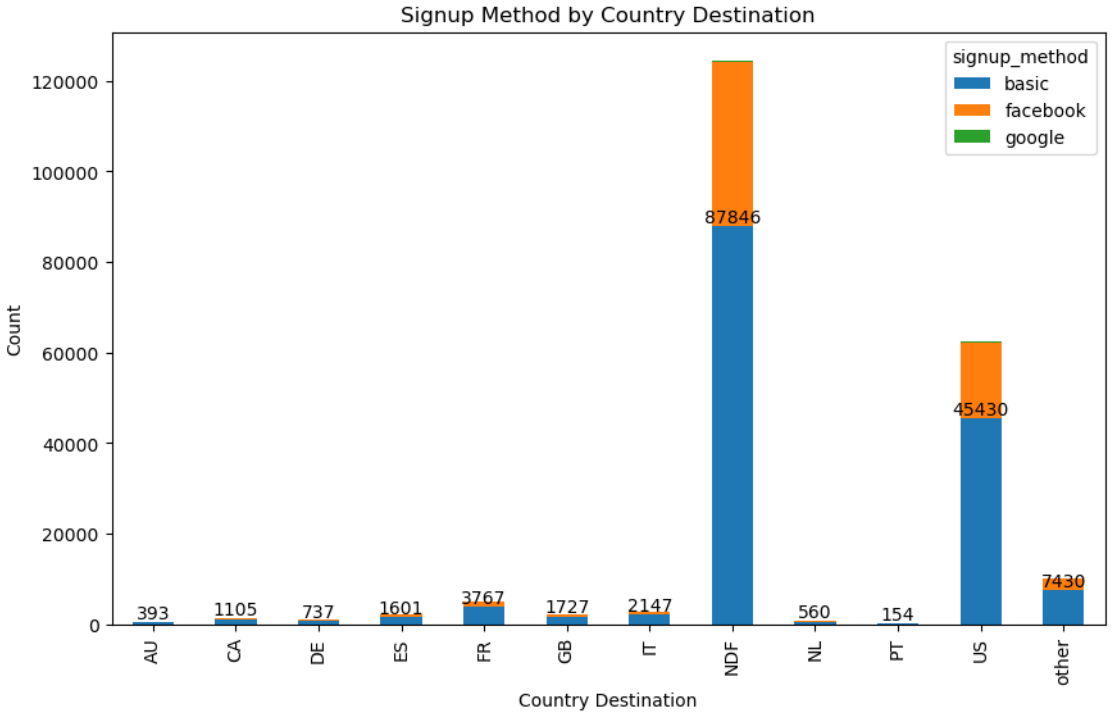
- Most users, regardless of gender, belong to the **NDF** group.
- The **US** is the second most common destination for all genders.
- Users with "Other" gender make up a very small portion.

[46]:
```
# Cross-tabulation of age_group and country_destination
cross_tab = pd.crosstab(df_train['age_group'], df_train['country_destination'])

plt.figure(figsize=(10, 8))
sns.heatmap(cross_tab, annot=True, fmt="d", cmap='YlGnBu')

plt.title('Age Group vs Country Destination')
plt.xlabel('Country Destination')
plt.ylabel('Age Group')

plt.show()
```



[47]:
```
plt.figure(figsize=(10,10))
ax=df_train.groupby('country_destination')['signup_method'].value_counts().
  ↪unstack().plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('Signup Method by Country Destination')
plt.xlabel('Country Destination')
plt.ylabel('Count')
ax.bar_label(ax.containers[0])
plt.show()
```

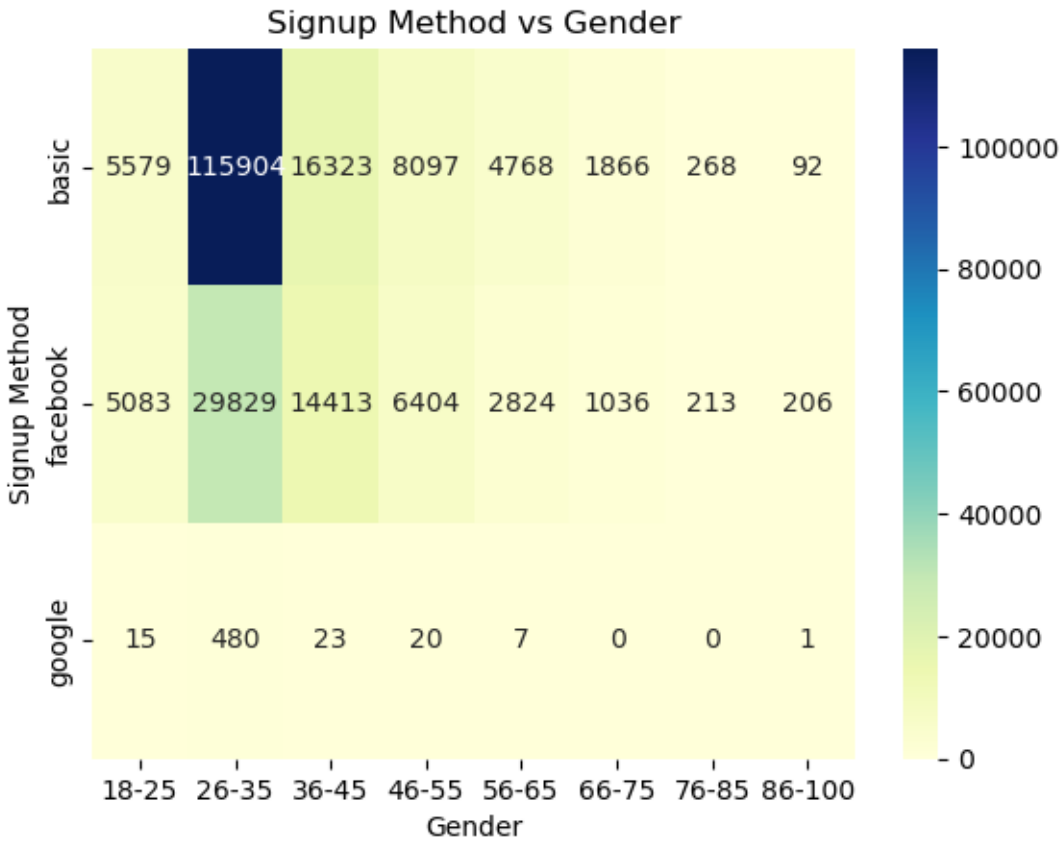
```
<Figure size 1000x1000 with 0 Axes>
```



- Most users signed up using **basic**, especially for **NDF** and the **US**.
- **Facebook** is the second most common signup method, used more for smaller destinations
- **Google** is the least used signup method across all destinations.

[48]:
```
cross_tab = pd.crosstab(df_train['signup_method'], df_train['age_group'])
sns.heatmap(cross_tab, annot=True, fmt="d", cmap='YlGnBu')
plt.title('Signup Method vs Gender')
plt.xlabel('Gender')
plt.ylabel('Signup Method')
```

```
plt.show()
```



==========**Sessions Data**==========

[49]:
```
df_sessions.head()
```

[49]:
```
      user_id          action action_type        action_detail  \
0  d1mm9tcy42          lookup         NaN                  NaN
1  d1mm9tcy42  search_results       click  view_search_results
2  d1mm9tcy42          lookup         NaN                  NaN
3  d1mm9tcy42  search_results       click  view_search_results
4  d1mm9tcy42          lookup         NaN                  NaN

       device_type  secs_elapsed
0  Windows Desktop         319.0
1  Windows Desktop       67753.0
2  Windows Desktop         301.0
3  Windows Desktop       22141.0
4  Windows Desktop         435.0
```

[50]:
```
print(f'shape of data is {df_sessions.shape[0]} , {df_sessions.shape[1]}')
```

```
shape of data is 10567737 , 6
```

[51]:
```
df_sessions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10567737 entries, 0 to 10567736
Data columns (total 6 columns):
 #   Column         Dtype
---  ------         -----
 0   user_id        object
 1   action         object
 2   action_type    object
 3   action_detail  object
 4   device_type    object
 5   secs_elapsed   float64
dtypes: float64(1), object(5)
memory usage: 483.8+ MB
```

[52]:
```
for col in df_sessions.columns:
    print(f' {col}:  \n number of unique value for each column␣
 ↪{df_sessions[col].nunique()} , \n unique values is {df_sessions[col].
 ↪unique()}')
    print('='*100)
```

```
 user_id:
 number of unique value for each column 135483 ,
 unique values is ['d1mm9tcy42' 'yo8nz8bqcq' '4grx6yxeby' … 'fa6260ziny'
'87k0fy4ugm'
 '9uqfg8txu3']
====================================================================================
====================
 action:
 number of unique value for each column 359 ,
 unique values is ['lookup' 'search_results' 'personalize' 'index'
'similar_listings'
 'ajax_refresh_subtotal' 'show' 'header_userpic' 'ask_question' nan
 'other_hosting_reviews_first' 'hosting_social_proof' 'decision_tree'
 'recent_reservations' 'faq_experiment_ids' 'multi' 'active' 'dashboard'
 'create' 'confirm_email' 'show_personalize' 'verify' 'pending'
 'requested' 'concierge' 'faq' 'clear_reservation' 'cancellation_policies'
 'track_page_view' 'update' 'my' 'campaigns' 'notifications' 'listings'
 'unavailabilities' 'ajax_lwlb_contact' 'ajax_check_dates' 'qt2'
 'request_new_confirm_email' 'ajax_photo_widget_form_iframe'
 'facebook_auto_login' 'identity' 'qt_reply_v2' 'travel_plans_current'
 'complete_status' 'populate_from_facebook' 'kba_update' 'kba' 'login'
 'authenticate' 'calendar_tab_inner2' 'other_hosting_reviews'
 'social_connections' 'relationship' '15' 'collections' '12'
```

'jumio_redirect' 'jumio_token' 'login_modal' 'domains'
'toggle_archived_thread' 'search' 'edit_verification' 'edit' 'ajax_ldp'
'connect' 'account' 'delete' 'phone_number_widget' 'callback'
'signup_modal' '10' 'open_graph_setting' 'reviews' 'signup_login'
'payment_instruments' 'payment_methods' 'pay' 'unread' 'at_checkpoint'
'push_notification_callback' 'faq_category' 'localization_settings'
'update_notifications' 'manage_listing' 'set_user' 'references'
'languages_multiselect' 'salute' 'rentals' 'currencies' 'new' 'position'
'populate_help_dropdown' 'popular' 'popular_listing' 'listing'
'available' 'glob' 'this_hosting_reviews' 'widget' 'complete'
'profile_pic' 'signature' 'apply_reservation' 'ajax_statsd' 'travel'
'tos_confirm' 'uptodate' 'ajax_payout_options_by_country'
'payout_preferences' 'payout_update' 'ajax_payout_edit' 'pending_tickets'
'issue' 'contact_new' 'itinerary' 'receipt'
'update_hide_from_search_engines' 'settings' 'privacy' 'reviews_new'
'add_guests' 'ajax_image_upload' 'qt_with' 'webcam_upload' 'friends'
'ajax_google_translate_description' 'trust' 'guarantee' 'supported'
'countries' 'status' 'upload' 'authorize' 'rate' 'remove_dashboard_alert'
'ajax_get_results' 'recommend' 'change_currency'
'transaction_history_paginated' 'transaction_history' 'set_password'
'cancel' 'phone_verification_modal' 'submit_contact' 'detect_fb_session'
'tell_a_friend' 'change' 'clickthrough' 'multi_message_attributes'
'update_friends_display' 'my_listings' 'email_itinerary_colorbox'
'mobile_landing_page' 'create_ach' 'country_options' '11' 'host_2013'
'terms' 'multi_message' 'ajax_send_message' 'airbnb_picks' 'friends_new'
'reputation' 'complete_redirect' 'toggle_starred_thread' 'email_share'
'email_wishlist' 'destroy' 'add_note' 'overview' 'requirements'
'update_reservation_requirements' 'image_order' 'review_page'
'ajax_worth' 'place_worth' 'google_importer' 'change_availability'
'hospitality' 'change_password' 'feed' 'host_summary'
'ajax_price_and_availability' 'jumio' 'guest_booked_elsewhere' 'about_us'
'founders' 'travel_plans_previous' 'why_host' 'hospitality_standards'
'social' 'recommendations' 'update_cached' 'become_user' 'departments'
'department' 'office_location' 'photography' 'preapproval'
'maybe_information' 'toggle_availability' 'payoneer_account_redirect'
'forgot_password' 'new_host' 'payoneer_signup_complete'
'ajax_google_translate' 'onenight' 'approve' 'nyan' 'booking'
'payout_delete' 'change_default_payout' 'envoy_bank_details_redirect'
'respond' 'request_photography' 'zendesk_login_jwt' 'ajax_photo_widget'
'message' 'southern-europe' 'life' 'press_release' 'media_resources'
'press_news' 'referrer_status' 'create_multiple' 'load_more'
'create_paypal' 'locale_from_host' 'terms_and_conditions'
'invalid_action' 'photography_update' 'badge' 'apply' 'redirect'
'pricing' 'apply_code' 'slideshow' 'locations'
'ajax_google_translate_reviews' 'new_session' 'has_profile_pic' 'sublets'
'wishlists' 'reservation' 'localized' 'home_safety_landing' 'click'
'how_it_works' 'ajax_get_referrals_amt' 'phone_verification' 'satisfy'
'city_count' 'recommendation_page' 'press_content'

```
 'guest_billing_receipt' 'ajax_payout_split_edit' 'print_confirmation'
 'envoy_form' 'ajax_special_offer_dates_available'
 'ajax_referral_banner_experiment_type' 'patch' 'questions'
 'home_safety_terms' 'track_activity' 'check' 'sldf'
 'recommended_listings' 'mobile_oauth_callback' 'show_code'
 'signed_out_modal' 'plaxo_cb' 'views' 'friend_listing'
 'ajax_referral_banner_type' 'deactivate' 'sync' 'social-media'
 'united-states' 'email_by_key' 'disaster_action' 'views_campaign'
 'update_message' 'spoken_languages' 'use_mobile_site' 'deauthorize'
 'special_offer' 'top_destinations' 'create_airbnb' 'handle_vanity_url'
 'impressions' 'message_to_host_focus' 'cancellation_policy_click'
 'message_to_host_change' 'agree_terms_check' 'read_policy_click'
 'phone_verification_success'
 'phone_verification_number_sucessfully_submitted'
 'phone_verification_number_submitted_for_sms'
 'phone_verification_phone_number_removed' 'endpoint_error'
 'p4_refund_policy_terms' 'apply_coupon_error_type' 'apply_coupon_error'
 'apply_coupon_click' 'coupon_field_focus' 'coupon_code_click'
 'agree_terms_uncheck' 'p4_terms'
 'phone_verification_call_taking_too_long'
 'phone_verification_number_submitted_for_call' 'phone_verification_error'
 'apply_coupon_click_success' 'set_default' 'update_country_of_residence'
 'open_hard_fallback_modal' 'tos_2014' 'views_campaign_rules'
 'weibo_signup_referral_finish' 'signup_weibo_referral'
 'similar_listings_v2' 'confirmation' 'signup_weibo'
 'acculynk_load_pin_pad' 'acculynk_bin_check_success'
 'acculynk_session_obtained' 'acculynk_pin_pad_inactive' 'reactivate'
 'airbrb' 'desks' 'sandy' 'unsubscribe' 'host_cancel'
 'acculynk_bin_check_failed' 'acculynk_pin_pad_error'
 'custom_recommended_destinations' 'this_hosting_reviews_3000'
 'reset_calendar' 'events' 'business_travel' 'add_guest_colorbox'
 'hard_fallback_submit' 'add_business_address_colorbox' 'my_reservations'
 'report' 'book' 'revert_to_admin' 'acculynk_pin_pad_success'
 'south-america' 'braintree_client_token' 'view' 'stpcv'
 'set_minimum_payout_amount' 'support_phone_numbers'
 'refund_guest_cancellation' 'accept_decline' 'deactivated'
 'rest-of-world']
================================================================================
====================
 action_type:
 number of unique value for each column 10 ,
 unique values is [nan 'click' 'data' 'view' 'submit' 'message_post' '-unknown-'
 'booking_request' 'partner_callback' 'booking_response' 'modify']
================================================================================
====================
 action_detail:
 number of unique value for each column 155 ,
 unique values is [nan 'view_search_results' 'wishlist_content_update'
```

```
'similar_listings'
 'change_trip_characteristics' 'p3' 'header_userpic' 'contact_host'
 'message_post' '-unknown-' 'dashboard' 'create_user' 'confirm_email_link'
 'user_profile_content_update' 'user_profile' 'pending' 'p5'
 'create_phone_numbers' 'cancellation_policies' 'user_wishlists'
 'change_contact_host_dates' 'wishlist' 'message_thread'
 'request_new_confirm_email' 'send_message' 'your_trips' 'login_page'
 'login' 'login_modal' 'toggle_archived_thread' 'p1'
 'profile_verifications' 'edit_profile' 'oauth_login'
 'post_checkout_action' 'account_notification_settings'
 'update_user_profile' 'oauth_response' 'signup_modal' 'signup_login_page'
 'at_checkpoint' 'manage_listing' 'create_listing' 'your_listings'
 'profile_references' 'list_your_space' 'popular_wishlists'
 'listing_reviews_page' 'apply_coupon' 'user_tax_forms'
 'account_payout_preferences' 'guest_itinerary' 'guest_receipt'
 'account_privacy_settings' 'lookup_message_thread' 'friends_wishlists'
 'host_guarantee' 'delete_phone_numbers' 'account_transaction_history'
 'set_password' 'guest_cancellation' 'change_or_alter' 'your_reservations'
 'terms_and_privacy' 'airbnb_picks_wishlists' 'toggle_starred_thread'
 'email_wishlist' 'email_wishlist_button' 'wishlist_note'
 'calculate_worth' 'place_worth' 'change_password' 'alteration_field'
 'previous_trips' 'update_listing' 'update_listing_description'
 'user_reviews' 'update_user' 'notifications' 'user_social_connections'
 'unavailable_dates' 'reservations' 'listing_reviews' 'user_listings'
 'signup' 'message_inbox' 'trip_availability' 'payment_instruments'
 'admin_templates' 'host_home' 'translations' 'forgot_password' 'homepage'
 'remove_dashboard_alert' 'user_friend_recommendations' 'confirm_email'
 'host_respond' 'booking' 'respond_to_alteration_request'
 'alteration_request' 'create_alteration_request' 'delete_listing'
 'set_password_page' 'delete_listing_description'
 'translate_listing_reviews' 'book_it' 'instant_book' 'request_to_book'
 'complete_booking' 'change_availability' 'special_offer_field'
 'listing_recommendations' 'view_listing' 'listing_descriptions'
 'user_languages' 'p4' 'message_to_host_focus' 'cancellation_policy_click'
 'message_to_host_change' 'read_policy_click' 'phone_verification_success'
 'p4_refund_policy_terms' 'apply_coupon_error' 'apply_coupon_click'
 'coupon_field_focus' 'coupon_code_click' 'p4_terms'
 'apply_coupon_click_success' 'tos_2014' 'view_reservations'
 'view_locations' 'modify_users' 'view_security_checks' 'phone_numbers'
 'profile_reviews' 'modify_reservations' 'view_resolutions'
 'account_payment_methods' 'create_payment_instrument'
 'set_default_payment_instrument' 'delete_payment_instrument' 'photos'
 'click_reviews' 'move_map' 'share' 'cancellation_policy'
 'click_about_host' 'click_amenities' 'host_refund_guest'
 'host_respond_page' 'view_user_real_names' 'view_identity_verifications'
 'view_ghosting_reasons' 'view_ghostings' 'host_standard_suspension'
 'deactivate_user_account']
================================================================================
```

```
====================
 device_type:
 number of unique value for each column 14 ,
 unique values is ['Windows Desktop' '-unknown-' 'Mac Desktop' 'Android Phone'
'iPhone'
 'iPad Tablet' 'Android App Unknown Phone/Tablet' 'Linux Desktop' 'Tablet'
 'Chromebook' 'Blackberry' 'iPodtouch' 'Windows Phone' 'Opera Phone']
================================================================================
====================
 secs_elapsed:
 number of unique value for each column 337661 ,
 unique values is [3.190000e+02 6.775300e+04 3.010000e+02 … 2.870570e+05
1.551558e+06
 1.752436e+06]
================================================================================
====================
```

[53]:
```
print(df_sessions.isnull().sum())
```

```
user_id            34496
action             79626
action_type      1126204
action_detail    1126204
device_type            0
secs_elapsed      136031
dtype: int64
```

[54]:
```
df_sessions.describe().T
```

[54]:

	count	mean	std	min	25%	50%	75%	max
secs_elapsed	10431706.0	19405.810751	88884.243208	0.0	229.0	1147.0	8444.0	1799977.0

Handle Missing Value

[55]:
```
df_sessions['action'] = df_sessions['action'].fillna('unknown')
df_sessions['action_type'] = df_sessions['action_type'].fillna('unknown')
df_sessions['action_detail'] = df_sessions['action_detail'].fillna('unknown')
df_sessions['secs_elapsed'] = df_sessions['secs_elapsed'].
  ↪fillna(df_sessions['secs_elapsed'].median())

print(df_sessions.isnull().sum())
```

```
user_id          34496
action               0
action_type          0
action_detail        0
```

```
device_type          0
secs_elapsed         0
dtype: int64
```

[56]:
```
df_sessions['device_type'] = df_sessions['device_type'].replace('-unknown-',␣
↪'Unknown')
df_sessions['action_type'] = df_sessions['action_type'].replace('-unknown-',␣
↪'Unknown')
```

7 Merging Cleaned Training Data with Sessions Data

[57]:
```
df_sessions.rename(columns={'user_id': 'id'}, inplace=True)
```

[58]:
```
sessions_summary = df_sessions.groupby('id').agg({
    'action': 'count',
    'device_type': 'nunique',
    'secs_elapsed': 'sum'
}).reset_index()

sessions_summary.rename(columns={
    'action': 'num_actions',
    'device_type': 'num_devices',
    'secs_elapsed': 'total_secs'
}, inplace=True)


df_train = df_train.merge(sessions_summary, on='id', how='left')
```

Grouped session data by user ID to calculate total actions, unique devices, and time spent:
- `action` → `num_actions` (total actions performed by the user),
- `device_type` → `num_devices` (number of unique devices used),
- `secs_elapsed` → `total_secs` (total time spent).

Merged this summary with `df_train` to enrich it with user activity details for better predictions."

[59]:
```
df_train.isnull().sum()
```

[59]:
```
id                          0
date_first_booking          0
gender                      0
signup_method               0
signup_flow                 0
language                    0
affiliate_channel           0
affiliate_provider          0
first_affiliate_tracked     0
signup_app                  0
```

```
first_device_type              0
first_browser                  0
country_destination            0
day_account_created            0
month_account_created          0
year_account_created           0
day_first_active               0
month_first_active             0
year_first_active              0
day_first_booking              0
month_first_booking            0
year_first_booking             0
age_group                      0
num_actions               139636
num_devices               139636
total_secs                139636
dtype: int64
```

[60]:
```
df_train.describe().T
```

[60]:

	count	mean	\
date_first_booking	213451	2013-08-13 13:37:17.487760896	
signup_flow	213451.0	3.267387	
day_account_created	213451.0	15.86923	
month_account_created	213451.0	6.022459	
year_account_created	213451.0	2013.023846	
day_first_active	213451.0	15.869071	
month_first_active	213451.0	6.022385	
year_first_active	213451.0	2013.023218	
day_first_booking	213451.0	12.934561	
month_first_booking	213451.0	7.796487	
year_first_booking	213451.0	2013.017845	
num_actions	73815.0	75.024819	
num_devices	73815.0	1.313663	
total_secs	73815.0	1515381.958992	

	min	25%	\
date_first_booking	2010-01-02 00:00:00	2013-09-11 00:00:00	
signup_flow	0.0	0.0	
day_account_created	1.0	8.0	
month_account_created	1.0	3.0	
year_account_created	2010.0	2012.0	
day_first_active	1.0	8.0	
month_first_active	1.0	3.0	
year_first_active	2009.0	2012.0	
day_first_booking	1.0	11.0	
month_first_booking	1.0	7.0	

```
year_first_booking                   2010.0                2013.0
num_actions                             1.0                  13.0
num_devices                             1.0                   1.0
total_secs                           1147.0              258067.5

                                        50%                   75%  \
date_first_booking     2013-09-11 00:00:00   2013-09-11 00:00:00
signup_flow                             0.0                   0.0
day_account_created                    16.0                  23.0
month_account_created                   6.0                   9.0
year_account_created                 2013.0                2014.0
day_first_active                       16.0                  23.0
month_first_active                      6.0                   9.0
year_first_active                    2013.0                2014.0
day_first_booking                      11.0                  13.0
month_first_booking                     9.0                   9.0
year_first_booking                   2013.0                2013.0
num_actions                            38.0                  91.0
num_devices                             1.0                   2.0
total_secs                         874009.0             2044634.5

                                        max            std
date_first_booking     2015-06-29 00:00:00            NaN
signup_flow                            25.0       7.637707
day_account_created                    31.0       8.740107
month_account_created                  12.0        3.23669
year_account_created                 2014.0       0.938489
day_first_active                       31.0       8.739582
month_first_active                     12.0       3.236501
year_first_active                    2014.0       0.939039
day_first_booking                      31.0       6.079097
month_first_booking                    12.0       2.498683
year_first_booking                   2015.0       0.656313
num_actions                          2644.0     112.543174
num_devices                             6.0       0.558021
total_secs                       38222510.0  1913191.47547
```

[61]:
```
df_train['num_actions'] = df_train['num_actions'].
  ↪fillna(df_train['num_actions'].median())
df_train['num_devices'] = df_train['num_devices'].
  ↪fillna(df_train['num_devices'].median())
df_train['total_secs'] = df_train['total_secs'].fillna(df_train['total_secs'].
  ↪median())
```

[62]:
```
df_train.drop(columns=['id'], inplace=True)
```

[63]:
```
df_train
```

[63]:

	date_first_booking	gender	signup_method	signup_flow	language
0	2013-09-11	Unknown	facebook	0	en
1	2013-09-11	MALE	facebook	0	en
2	2010-08-02	FEMALE	basic	3	en
3	2012-09-08	FEMALE	facebook	0	en
4	2010-02-18	Unknown	basic	0	en
…	…	…	…	…	…
213446	2013-09-11	MALE	basic	0	en
213447	2013-09-11	Unknown	basic	0	en
213448	2013-09-11	Unknown	basic	0	en
213449	2013-09-11	Unknown	basic	25	en
213450	2013-09-11	Unknown	basic	25	en

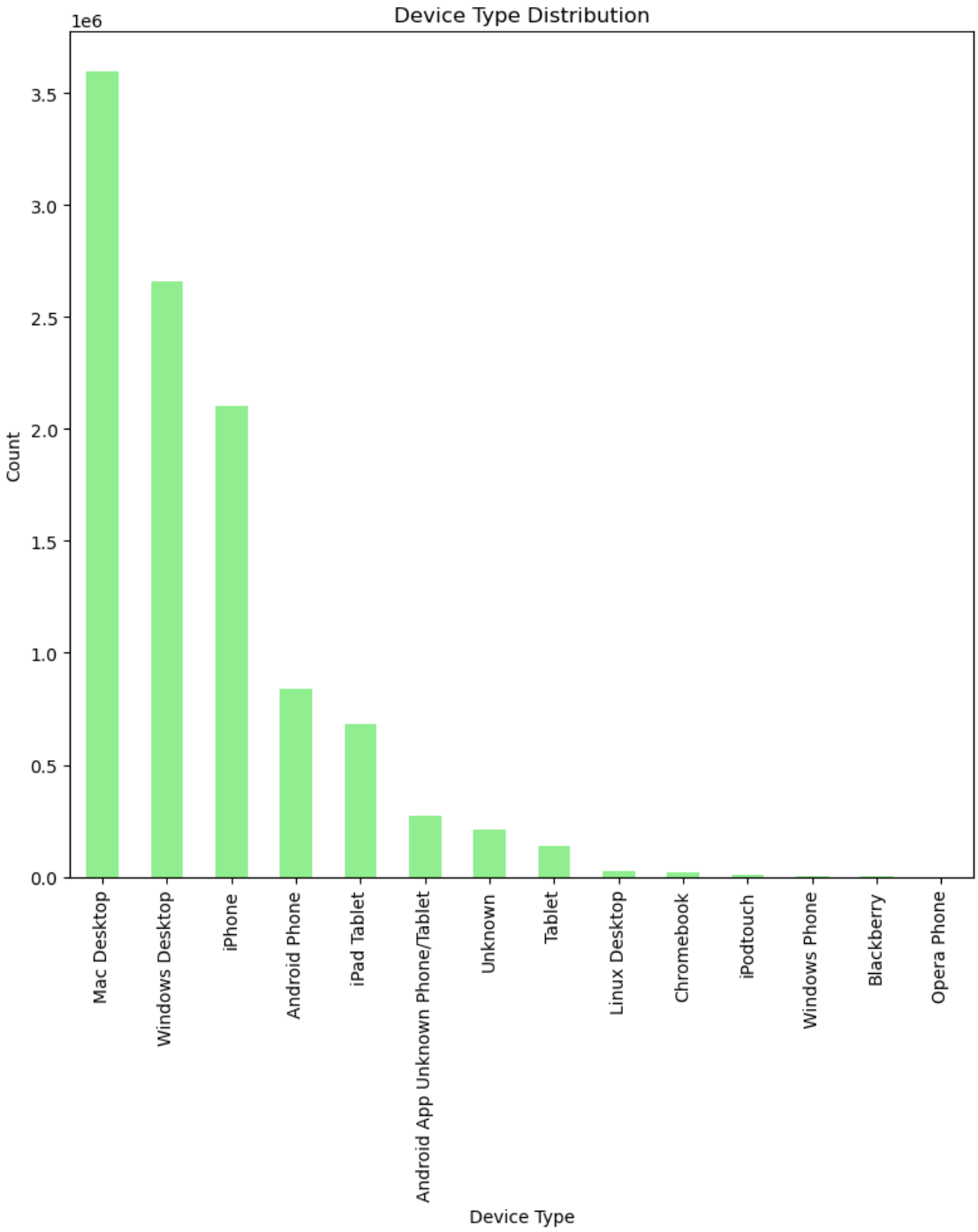
	affiliate_channel	affiliate_provider	first_affiliate_tracked
0	direct	direct	untracked
1	seo	google	untracked
2	direct	direct	untracked
3	direct	direct	untracked
4	direct	direct	untracked
…	…	…	…
213446	sem-brand	google	omg
213447	direct	direct	linked
213448	direct	direct	untracked
213449	other	other	tracked-other
213450	direct	direct	untracked

	signup_app	first_device_type	…	day_first_active	month_first_active
0	Web	Mac Desktop	…	19	3
1	Web	Mac Desktop	…	23	5
2	Web	Windows Desktop	…	9	6
3	Web	Mac Desktop	…	31	10
4	Web	Mac Desktop	…	8	12
…	…	…	…	…	…
213446	Web	Mac Desktop	…	30	6
213447	Web	Windows Desktop	…	30	6
213448	Web	Mac Desktop	…	30	6
213449	iOS	iPhone	…	30	6
213450	iOS	iPhone	…	30	6

	year_first_active	day_first_booking	month_first_booking
0	2009	11	9
1	2009	11	9
2	2009	2	8
3	2009	8	9
4	2009	18	2
…	…	…	…
213446	2014	11	9

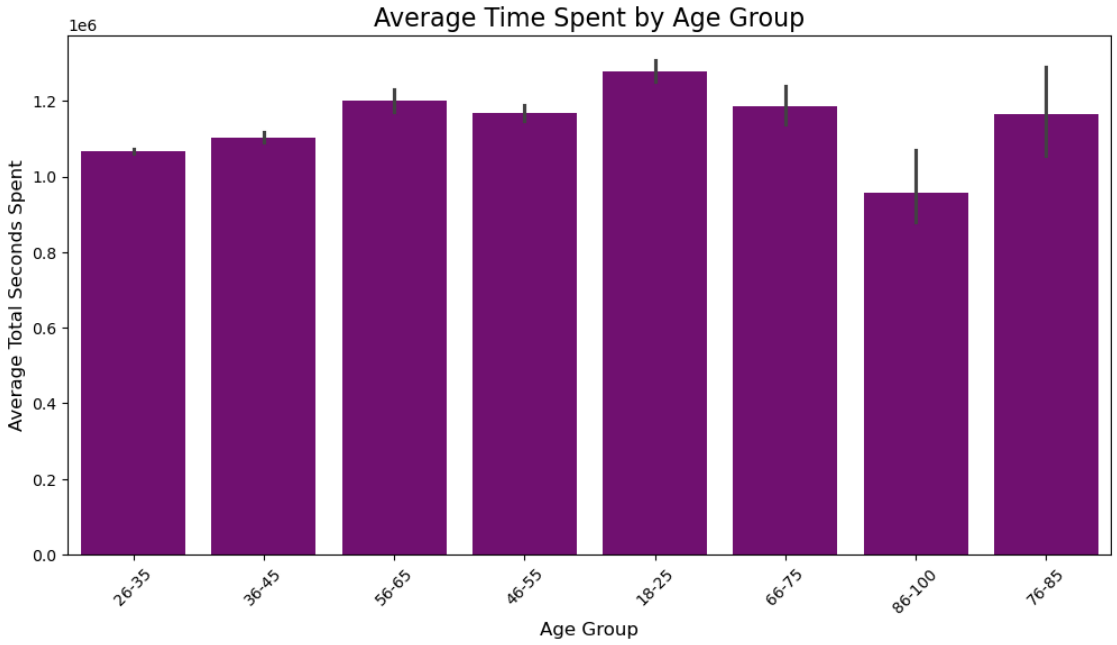
```
213447                2014                 11                   9
213448                2014                 11                   9
213449                2014                 11                   9
213450                2014                 11                   9

        year_first_booking  age_group  num_actions  num_devices  total_secs
0                     2013      26-35         38.0          1.0    874009.0
1                     2013      36-45         38.0          1.0    874009.0
2                     2010      56-65         38.0          1.0    874009.0
3                     2012      36-45         38.0          1.0    874009.0
4                     2010      36-45         38.0          1.0    874009.0
…                        …          …            …            …          …
213446                2013      26-35        110.0          2.0   5142543.0
213447                2013      26-35        238.0          3.0   2880071.0
213448                2013      26-35         18.0          1.0    344129.0
213449                2013      26-35         75.0          1.0    342756.0
213450                2013      26-35         41.0          1.0   2760357.0

[213451 rows x 25 columns]
```

[64]:
```
plt.figure(figsize=(8,10))
ax=df_sessions['device_type'].value_counts().plot(kind='bar',␣
 ↪color='lightgreen')
plt.title('Device Type Distribution')
plt.xlabel('Device Type')
plt.ylabel('Count')
# ax.bar_label(ax.containers[0])
plt.tight_layout()
plt.show()
```



Most users use **Mac Desktop**, **Windows Desktop**, and **iPhones**. Fewer users prefer **Android phones** and **iPads**, while very few use rare devices like **Blackberry** or **Windows Phone**.ipynb_checkpoints

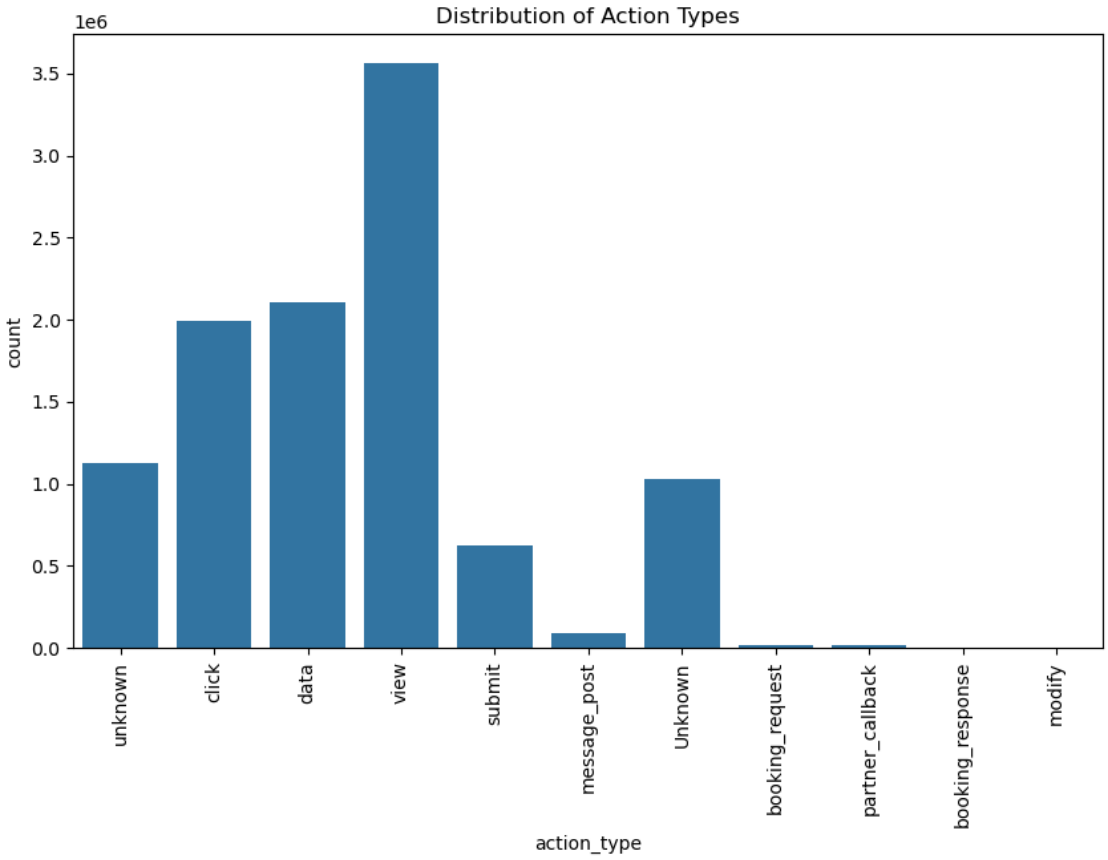
[65]:
```
plt.figure(figsize=(12, 6))
ax=sns.barplot(x='age_group', y='total_secs', data=df_train,  estimator='mean',␣
 ↪color='purple')
plt.title("Average Time Spent by Age Group", fontsize=16)
plt.xlabel("Age Group", fontsize=12)
plt.ylabel("Average Total Seconds Spent", fontsize=12)
plt.xticks(rotation=45)
plt.show()
```



shows the **average total time spent** by different age groups.

- The **18-25** age group spends the most time on average.
- Other age groups, like **56-65** and **26-35**, also have high average time spent.
- Older age groups, especially **86-100**, spend less time on average compared to younger ones.

[66]:
```
plt.figure(figsize=(10, 6))
sns.countplot(data=df_sessions, x='action_type')
plt.title('Distribution of Action Types')
plt.xticks(rotation=90)
plt.show()
```

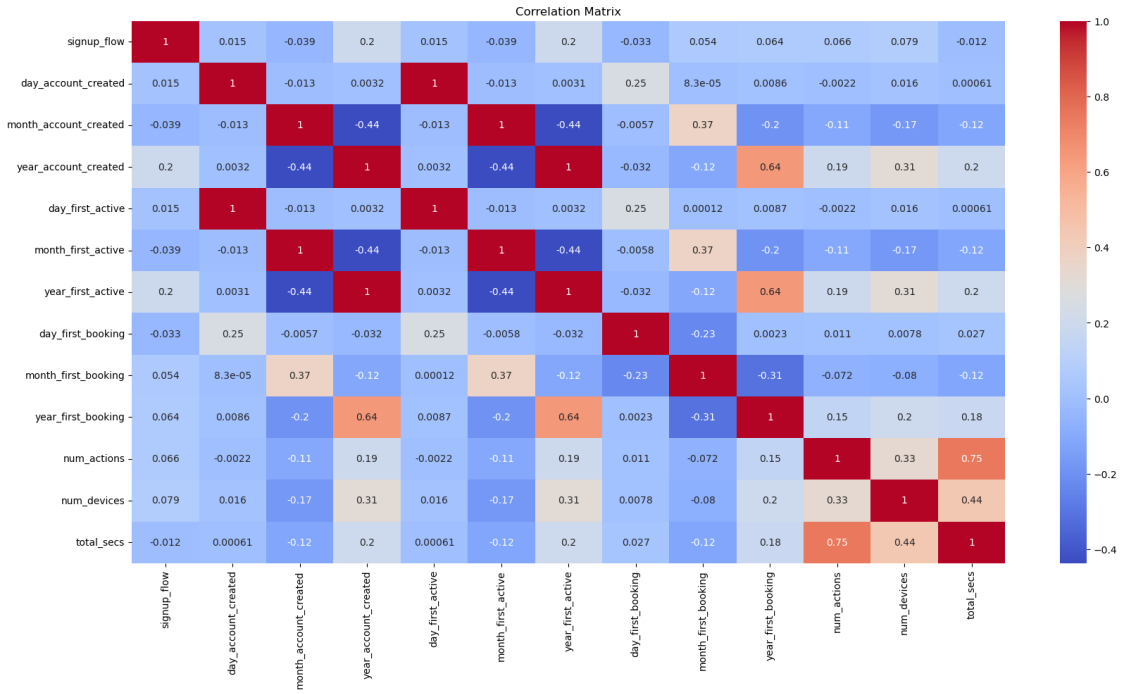


distribution of action types:

- **View** is the most common action type, with the highest count.
- **Click** and **Data** actions are also frequent, but less than "View".
- Other actions like **Submit** and **Unknown** occur less often.
- Rare actions include **Message Post**, **Booking Request**, and **Modify**.

3-MultiVariate Analysis

[67]:
```
plt.figure(figsize=(20, 10))

corr_matrix = df_train.select_dtypes(include='number').corr()#correlation␣
 ↪heatmap to show relatioships
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



correlation matrix shows the relationship between different variables:

1. **Strong Relationships** (Useful for prediction):
   - **`num_actions` & `total_secs`**: Correlation = **0.75**
     -> More time on the platform = More actions. Both are important for predictions.
   - **`year_account_created` & `year_first_booking`**: Correlation = **0.64**
     -> Users often book in the same year they create their account.
2. **Weak Relationships** (Not very useful):
   - **`signup_flow`**: Correlation with other variables is between **-0.03 and 0.08**
     -> No strong impact, can be ignored.
3. **Moderate Negative Relationship**:
   - **`month_account_created` & `year_account_created`**: Correlation = **-0.44**
     -> Inverse relationship, may have some predictive value.
4. **Weak Action-Booking Link**:
   - **`num_actions` & `year_first_booking`**: Correlation = **0.15**
     -> Number of actions does not strongly indicate the booking year.

7.0.1 Final Tip:

Focus on **`num_actions`**, **`total_secs`**, **`year_account_created`**, and **`year_first_booking`**. Ignore **`signup_flow`**.

==========sample_submission Data==========

[68]:
```
df_sample_submission_NDF
```

[68]:
```
             id country
0      5uwns89zht     NDF
1      jtl0dijy2j     NDF
2      xx0ulgorjt     NDF
3      6c6puo6ix0     NDF
4      czqhjk3yfe     NDF
…           …     …
62091  cv0na2lf5a     NDF
62092  zp8xfonng8     NDF
62093  fa6260ziny     NDF
62094  87k0fy4ugm     NDF
62095  9uqfg8txu3     NDF

[62096 rows x 2 columns]
```

[69]:
```
print(f'shape of data is {df_sample_submission_NDF.shape[0]} ,␣
↪{df_sample_submission_NDF.shape[1]}')
```

```
shape of data is 62096 , 2
```

[70]:
```
df_sample_submission_NDF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62096 entries, 0 to 62095
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   id       62096 non-null  object
 1   country  62096 non-null  object
dtypes: object(2)
memory usage: 970.4+ KB
```

[71]:
```
for col in df_sample_submission_NDF.columns:
    print(f' {col}:  \n number of unique value for each column␣
↪{df_sample_submission_NDF[col].nunique()} , \n unique values is␣
↪{df_sample_submission_NDF[col].unique()}')
    print('='*100)
```

```
 id:
 number of unique value for each column 62096 ,
 unique values is ['5uwns89zht' 'jtl0dijy2j' 'xx0ulgorjt' … 'fa6260ziny'
'87k0fy4ugm'
 '9uqfg8txu3']
====================================================================================
====================
 country:
 number of unique value for each column 1 ,
 unique values is ['NDF']
====================================================================================
```

==========

==========**countries Data**==========

[72]:
```
df_countries
```

[72]:

	country_destination	lat_destination	lng_destination	distance_km	\
0	AU	-26.853388	133.275160	15297.7440	
1	CA	62.393303	-96.818146	2828.1333	
2	DE	51.165707	10.452764	7879.5680	
3	ES	39.896027	-2.487694	7730.7240	
4	FR	46.232193	2.209667	7682.9450	
5	GB	54.633220	-3.432277	6883.6590	
6	IT	41.873990	12.564167	8636.6310	
7	NL	52.133057	5.295250	7524.3203	
8	PT	39.553444	-7.839319	7355.2534	
9	US	36.966427	-95.844030	0.0000	

	destination_km2	destination_language	language_levenshtein_distance
0	7741220.0	eng	0.00
1	9984670.0	eng	0.00
2	357022.0	deu	72.61
3	505370.0	spa	92.25
4	643801.0	fra	92.06
5	243610.0	eng	0.00
6	301340.0	ita	89.40
7	41543.0	nld	63.22
8	92090.0	por	95.45
9	9826675.0	eng	0.00

[73]:
```
print(f'shape of data is {df_countries.shape[0]} , {df_countries.shape[1]}')
```

```
shape of data is 10 , 7
```

[74]:
```
df_countries.isnull().sum()
```

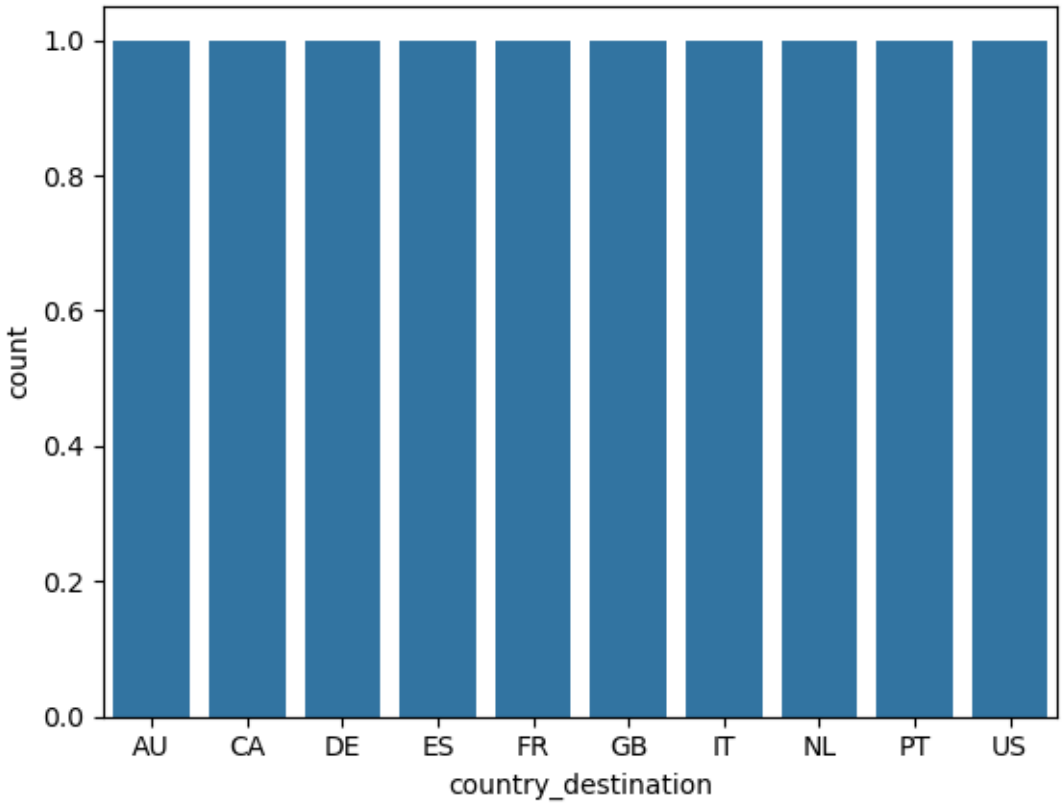
[74]:
```
country_destination             0
lat_destination                 0
lng_destination                 0
distance_km                     0
destination_km2                 0
destination_language            0
language_levenshtein_distance   0
dtype: int64
```

[75]:
```
for col in df_countries.columns:
```

```
    print(f' {col}:  \n number of unique value for each column␣
 ↪{df_countries[col].nunique()} , \n unique values is {df_countries[col].
 ↪unique()}')
    print('='*100)
```

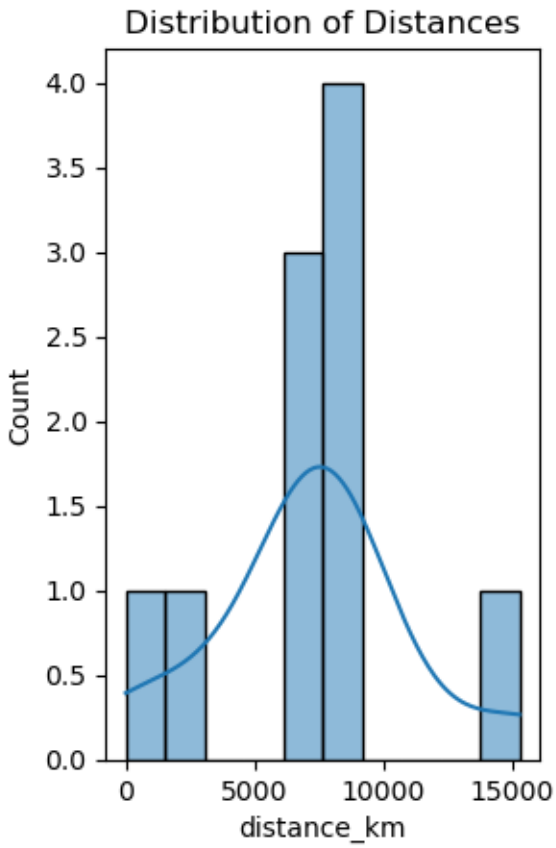
```
 country_destination:
 number of unique value for each column 10 ,
 unique values is ['AU' 'CA' 'DE' 'ES' 'FR' 'GB' 'IT' 'NL' 'PT' 'US']
====================================================================================
====================
 lat_destination:
 number of unique value for each column 10 ,
 unique values is [-26.853388  62.393303  51.165707  39.896027  46.232193
54.63322
  41.87399   52.133057  39.553444  36.966427]
====================================================================================
====================
 lng_destination:
 number of unique value for each column 10 ,
 unique values is [133.27516   -96.818146   10.452764   -2.4876945   2.209667
-3.4322774
  12.564167    5.29525    -7.839319  -95.84403  ]
====================================================================================
====================
 distance_km:
 number of unique value for each column 10 ,
 unique values is [15297.744   2828.1333  7879.568   7730.724   7682.945
6883.659
  8636.631   7524.3203  7355.2534     0.    ]
====================================================================================
====================
 destination_km2:
 number of unique value for each column 10 ,
 unique values is [7741220. 9984670.  357022.  505370.  643801.  243610.
301340.   41543.
   92090. 9826675.]
====================================================================================
====================
 destination_language :
 number of unique value for each column 7 ,
 unique values is ['eng' 'deu' 'spa' 'fra' 'ita' 'nld' 'por']
====================================================================================
====================
 language_levenshtein_distance:
 number of unique value for each column 7 ,
 unique values is [ 0.   72.61 92.25 92.06 89.4  63.22 95.45]
====================================================================================
====================
```

[76]:
```
ax=sns.countplot(x='country_destination', data=df_countries)
```



[77]:
```
plt.subplot(1, 2, 1)
sns.histplot(df_countries['distance_km'], bins=10, kde=True)
plt.title('Distribution of Distances')
```

[77]:
```
Text(0.5, 1.0, 'Distribution of Distances')
```



The graph shows most distances are between 5,000 and 10,000 km. Short (0–5,000 km) and long distances (above 10,000 km) are less common. Medium distances are the most frequent.

==========**age_gender Data**==========

[78]:
```
df_age_gender
```

[78]:

	age_bucket	country_destination	gender	population_in_thousands	year
0	100+	AU	male	1.0	2015.0
1	95-99	AU	male	9.0	2015.0
2	90-94	AU	male	47.0	2015.0
3	85-89	AU	male	118.0	2015.0
4	80-84	AU	male	199.0	2015.0
..	…	…	…	…	…
415	95-99	US	male	115.0	2015.0
416	90-94	US	male	541.0	2015.0
417	15-19	US	female	10570.0	2015.0
418	85-89	US	male	1441.0	2015.0
419	80-84	US	male	2442.0	2015.0

```
[420 rows x 5 columns]
```

[79]:
```
df_age_gender.isnull().sum()
```

[79]:
```
age_bucket                0
country_destination       0
gender                    0
population_in_thousands   0
year                      0
dtype: int64
```

[80]:
```
df_age_gender.info()
```

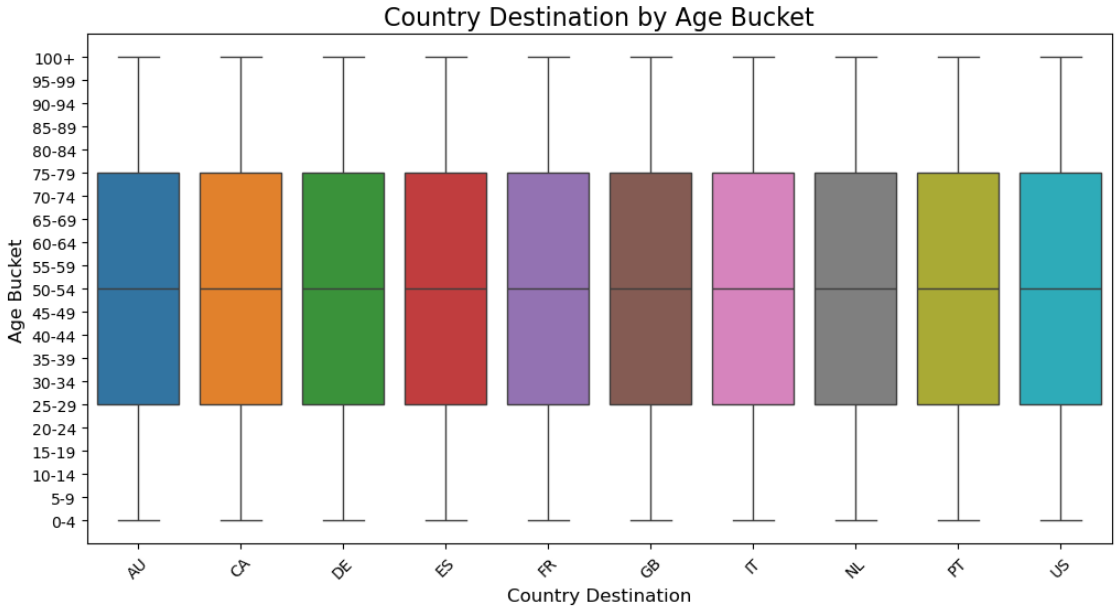
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 420 entries, 0 to 419
Data columns (total 5 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   age_bucket               420 non-null    object
 1   country_destination      420 non-null    object
 2   gender                   420 non-null    object
 3   population_in_thousands  420 non-null    float64
 4   year                     420 non-null    float64
dtypes: float64(2), object(3)
memory usage: 16.5+ KB
```

[81]:
```
columns = ['age_bucket', 'country_destination', 'gender', 'year']
for col in columns:
    print(f'{col}:')
    print(f'Number of unique values: {df_age_gender[col].nunique()}')
    print(f'Unique values: {df_age_gender[col].unique()}')
    print('=' * 100)
```

```
age_bucket:
Number of unique values: 21
Unique values: ['100+' '95-99' '90-94' '85-89' '80-84' '75-79' '70-74' '65-69'
'60-64'
 '55-59' '50-54' '45-49' '40-44' '35-39' '30-34' '25-29' '20-24' '15-19'
 '10-14' '5-9' '0-4']
====================================================================================
====================
country_destination:
Number of unique values: 10
Unique values: ['AU' 'CA' 'DE' 'ES' 'FR' 'GB' 'IT' 'NL' 'PT' 'US']
====================================================================================
====================
gender:
Number of unique values: 2
```

```
Unique values: ['male' 'female']
================================================================================
====================
year:
Number of unique values: 1
Unique values: [2015.]
================================================================================
====================
```

[82]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(data=df_age_gender, x='country_destination', y='age_bucket',␣
↪hue='country_destination')
plt.title('Country Destination by Age Bucket', fontsize=16)
plt.xlabel('Country Destination', fontsize=12)
plt.ylabel('Age Bucket', fontsize=12)
plt.xticks(rotation=45)
plt.legend([], frameon=False)
plt.show()
```



shows the relationship between **country destinations** and **age groups** of users.

- The **median age** is similar across all countries, mostly in the **50-54 age range**.
- The **age range** (spread) is also consistent for all destinations.
- No major outliers or unusual patterns are visible.

==========**df_test**

Data==========

I will apply the same processes I used for the training data to the test data, with some additional steps for better handling.

[83]:
```
df_test.head()
```

[83]:
```
           id date_account_created  timestamp_first_active  \
0  5uwns89zht           2014-07-01          20140701000006
1  jtl0dijy2j           2014-07-01          20140701000051
2  xx0ulgorjt           2014-07-01          20140701000148
3  6c6puo6ix0           2014-07-01          20140701000215
4  czqhjk3yfe           2014-07-01          20140701000305

   date_first_booking     gender   age signup_method  signup_flow language  \
0                 NaN     FEMALE  35.0      facebook            0       en
1                 NaN  -unknown-   NaN         basic            0       en
2                 NaN  -unknown-   NaN         basic            0       en
3                 NaN  -unknown-   NaN         basic            0       en
4                 NaN  -unknown-   NaN         basic            0       en

  affiliate_channel affiliate_provider first_affiliate_tracked signup_app  \
0            direct             direct                untracked      Moweb
1            direct             direct                untracked      Moweb
2            direct             direct                   linked        Web
3            direct             direct                   linked        Web
4            direct             direct                untracked        Web

  first_device_type  first_browser
0            iPhone  Mobile Safari
1            iPhone  Mobile Safari
2   Windows Desktop         Chrome
3   Windows Desktop             IE
4       Mac Desktop         Safari
```

[84]:
```
print(f'shape of data is {df_test.shape[0]} , {df_test.shape[1]}')
```

```
shape of data is 62096 , 15
```

[85]:
```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62096 entries, 0 to 62095
Data columns (total 15 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       62096 non-null  object
 1   date_account_created     62096 non-null  object
 2   timestamp_first_active   62096 non-null  int64
```

```
 3   date_first_booking       0 non-null      float64
 4   gender                   62096 non-null  object
 5   age                      33220 non-null  float64
 6   signup_method            62096 non-null  object
 7   signup_flow              62096 non-null  int64
 8   language                 62096 non-null  object
 9   affiliate_channel        62096 non-null  object
 10  affiliate_provider       62096 non-null  object
 11  first_affiliate_tracked  62076 non-null  object
 12  signup_app               62096 non-null  object
 13  first_device_type        62096 non-null  object
 14  first_browser            62096 non-null  object
dtypes: float64(2), int64(2), object(11)
memory usage: 7.1+ MB
```

[86]:
```
df_test.isnull().sum()
```

[86]:
```
id                             0
date_account_created           0
timestamp_first_active         0
date_first_booking         62096
gender                         0
age                        28876
signup_method                  0
signup_flow                    0
language                       0
affiliate_channel              0
affiliate_provider             0
first_affiliate_tracked       20
signup_app                     0
first_device_type              0
first_browser                  0
dtype: int64
```

[87]:
```
df_test.duplicated().sum()
```

[87]:
```
0
```

[88]:
```
for col in df_test.columns:
    print(f' {col}:  \n number of unique value for each column {df_test[col].
 ↪nunique()} , \n unique values is {df_test[col].unique()}')
    print('='*100)
```

```
 id:
 number of unique value for each column 62096 ,
 unique values is ['5uwns89zht' 'jtl0dijy2j' 'xx0ulgorjt' … 'fa6260ziny'
'87k0fy4ugm'
 '9uqfg8txu3']
```

```
================================================================================
====================
 date_account_created:
 number of unique value for each column 92 ,
 unique values is ['2014-07-01' '2014-07-02' '2014-07-03' '2014-07-04'
'2014-07-05'
 '2014-07-06' '2014-07-07' '2014-07-08' '2014-07-09' '2014-07-10'
 '2014-07-11' '2014-07-12' '2014-07-13' '2014-07-14' '2014-07-15'
 '2014-07-16' '2014-07-17' '2014-07-18' '2014-07-19' '2014-07-20'
 '2014-07-21' '2014-07-22' '2014-07-23' '2014-07-24' '2014-07-25'
 '2014-07-26' '2014-07-27' '2014-07-28' '2014-07-29' '2014-07-30'
 '2014-07-31' '2014-08-01' '2014-08-02' '2014-08-03' '2014-08-04'
 '2014-08-05' '2014-08-06' '2014-08-07' '2014-08-08' '2014-08-09'
 '2014-08-10' '2014-08-11' '2014-08-12' '2014-08-13' '2014-08-14'
 '2014-08-15' '2014-08-16' '2014-08-17' '2014-08-18' '2014-08-19'
 '2014-08-20' '2014-08-21' '2014-08-22' '2014-08-23' '2014-08-24'
 '2014-08-25' '2014-08-26' '2014-08-27' '2014-08-28' '2014-08-29'
 '2014-08-30' '2014-08-31' '2014-09-01' '2014-09-02' '2014-09-03'
 '2014-09-04' '2014-09-05' '2014-09-06' '2014-09-07' '2014-09-08'
 '2014-09-09' '2014-09-10' '2014-09-11' '2014-09-12' '2014-09-13'
 '2014-09-14' '2014-09-15' '2014-09-16' '2014-09-17' '2014-09-18'
 '2014-09-19' '2014-09-20' '2014-09-21' '2014-09-22' '2014-09-23'
 '2014-09-24' '2014-09-25' '2014-09-26' '2014-09-27' '2014-09-28'
 '2014-09-29' '2014-09-30']
================================================================================
====================
 timestamp_first_active:
 number of unique value for each column 62096 ,
 unique values is [20140701000006 20140701000051 20140701000148 …
20140930235408
 20140930235430 20140930235901]
================================================================================
====================
 date_first_booking:
 number of unique value for each column 0 ,
 unique values is [nan]
================================================================================
====================
 gender:
 number of unique value for each column 4 ,
 unique values is ['FEMALE' '-unknown-' 'MALE' 'OTHER']
================================================================================
====================
 age:
 number of unique value for each column 124 ,
 unique values is [3.500e+01       nan 2.800e+01 4.800e+01 3.000e+01 2.400e+01
5.600e+01
 3.300e+01 3.100e+01 5.300e+01 3.400e+01 2.500e+01 2.700e+01 3.200e+01
```

```
 5.900e+01 5.700e+01 2.600e+01 3.600e+01 2.200e+01 5.100e+01 3.800e+01
 3.900e+01 4.700e+01 4.200e+01 5.200e+01 1.050e+02 2.900e+01 5.400e+01
 1.900e+01 2.300e+01 4.100e+01 7.400e+01 4.900e+01 2.100e+01 4.400e+01
 5.000e+01 6.100e+01 4.500e+01 4.600e+01 3.700e+01 4.000e+01 1.800e+01
 2.000e+01 4.300e+01 8.000e+01 6.700e+01 5.500e+01 7.200e+01 6.300e+01
 5.800e+01 6.800e+01 8.300e+01 6.000e+01 7.000e+01 7.500e+01 9.500e+01
 6.500e+01 6.200e+01 1.060e+02 6.600e+01 1.700e+01 6.400e+01 7.100e+01
 7.600e+01 7.300e+01 8.900e+01 1.040e+02 1.090e+02 7.900e+01 1.600e+01
 6.900e+01 1.100e+02 9.300e+01 7.700e+01 1.000e+02 1.030e+02 9.900e+01
 8.100e+01 8.500e+01 9.700e+01 1.010e+02 1.937e+03 1.500e+01 1.934e+03
 9.400e+01 1.922e+03 1.931e+03 1.944e+03 7.800e+01 1.954e+03 8.600e+01
 1.940e+03 1.947e+03 1.020e+02 9.600e+01 1.927e+03 1.945e+03 2.000e+03
 1.928e+03 8.800e+01 2.001e+03 1.938e+03 8.400e+01 1.930e+03 1.968e+03
 8.200e+01 1.939e+03 1.951e+03 1.920e+03 9.000e+01 1.923e+03 1.926e+03
 9.100e+01 1.070e+02 9.200e+01 1.925e+03 9.800e+01 1.933e+03 1.935e+03
 1.000e+00 1.948e+03 2.002e+03 1.941e+03 1.924e+03 1.080e+02]
==========================================================================
====================
 signup_method:
 number of unique value for each column 4 ,
 unique values is ['facebook' 'basic' 'google' 'weibo']
==========================================================================
====================
 signup_flow:
 number of unique value for each column 7 ,
 unique values is [ 0 25  8 23 12 14 21]
==========================================================================
====================
 language:
 number of unique value for each column 24 ,
 unique values is ['en' 'de' 'zh' 'fr' 'ko' 'sv' 'no' 'it' 'es' 'nl' 'ja' 'ru'
'pt' 'tr'
 'cs' 'el' 'hu' 'pl' 'da' 'fi' 'th' 'ca' '-unknown-' 'id']
==========================================================================
====================
 affiliate_channel:
 number of unique value for each column 7 ,
 unique values is ['direct' 'sem-brand' 'sem-non-brand' 'seo' 'remarketing'
'other'
 'content']
==========================================================================
====================
 affiliate_provider:
 number of unique value for each column 17 ,
 unique values is ['direct' 'google' 'bing' 'facebook' 'other' 'craigslist'
'padmapper'
 'email-marketing' 'yahoo' 'baidu' 'naver' 'gsp' 'facebook-open-graph'
 'meetup' 'vast' 'daum' 'yandex']
```

```
==========================================================================
====================
 first_affiliate_tracked:
 number of unique value for each column 7 ,
 unique values is ['untracked' 'linked' 'omg' 'product' 'marketing' 'tracked-
other' nan
 'local ops']
==========================================================================
====================
 signup_app:
 number of unique value for each column 4 ,
 unique values is ['Moweb' 'Web' 'iOS' 'Android']
==========================================================================
====================
 first_device_type:
 number of unique value for each column 9 ,
 unique values is ['iPhone' 'Windows Desktop' 'Mac Desktop' 'iPad' 'Android
Tablet'
 'Android Phone' 'Desktop (Other)' 'Other/Unknown' 'SmartPhone (Other)']
==========================================================================
====================
 first_browser:
 number of unique value for each column 31 ,
 unique values is ['Mobile Safari' 'Chrome' 'IE' 'Safari' '-unknown-' 'Firefox'
 'Chrome Mobile' 'Android Browser' 'IE Mobile' 'BlackBerry Browser'
 'Opera' 'Silk' 'Mobile Firefox' 'AOL Explorer' 'SeaMonkey' 'Opera Mobile'
 'wOSBrowser' 'Chromium' 'Apple Mail' 'Maxthon' 'IBrowse' 'Sogou Explorer'
 'Iron' 'Yandex.Browser' 'SiteKiosk' 'Pale Moon' 'Nintendo Browser'
 'Opera Mini' 'CometBird' 'IceWeasel' 'UC Browser']
==========================================================================
====================
```

8 Fill 'date_first_booking' in df_test from df_train

[89]:
```
df_test['date_first_booking'].value_counts
```

[89]:
```
<bound method IndexOpsMixin.value_counts of 0       NaN
1       NaN
2       NaN
3       NaN
4       NaN
         ..
62091   NaN
62092   NaN
62093   NaN
62094   NaN
62095   NaN
```

```
Name: date_first_booking, Length: 62096, dtype: float64>
```

To fill the missing values in the `date_first_booking` column in `df_test` with random dates from the `date_first_booking` column in `df_train`, follow these steps:

1. **Find the date range**: Get the minimum and maximum dates from `df_train['date_first_booking']`.
2. **Generate random dates**: Create random dates within this range.
3. **Fill missing values**: Use these random dates to fill the NaN values in `df_test['date_first_booking']`.

This process ensures that missing dates in `df_test` are filled with random dates between the earliest and latest dates from `df_train`.

[90]:
```
df_test['date_first_booking'] = pd.to_datetime(df_test['date_first_booking'],␣
 ↪errors='coerce')
```

[91]:
```
oldest_date_1 = df_train['date_first_booking'].min()
latest_date_2 = df_train['date_first_booking'].max()

print(f'date_first_booking: {oldest_date_1} {latest_date_2}')
```

```
date_first_booking: 2010-01-02 00:00:00 2015-06-29 00:00:00
```

[92]:
```
start_date = oldest_date_1
end_date = latest_date_2

date_range = pd.date_range(start=start_date, end=end_date, freq='D')

missing_dates_count = df_test['date_first_booking'].isna().sum()

random_dates = np.random.choice(date_range, size=missing_dates_count)

df_test.loc[df_test['date_first_booking'].isna(), 'date_first_booking'] =␣
 ↪random_dates

print("Done ")
```

```
Done
```

[93]:
```
df_test['date_first_booking']
```

[93]:
```
0        2012-05-09
1        2011-10-29
2        2011-10-10
3        2011-05-22
4        2011-07-02
              ...
62091    2014-06-05
62092    2014-03-18
```

```
62093   2010-07-04
62094   2010-03-05
62095   2013-10-19
Name: date_first_booking, Length: 62096, dtype: datetime64[ns]
```

9 Transformation and Extracting

[94]:
```
df_test['date_account_created'] = pd.
 ↪to_datetime(df_test['date_account_created'])
```

[95]:
```
df_test['timestamp_first_active'] = df_test['timestamp_first_active'].
 ↪astype(str)
df_test['timestamp_first_active'] = pd.
 ↪to_datetime(df_test['timestamp_first_active'], format='%Y%m%d%H%M%S')
print(df_test['timestamp_first_active'].head())
```

```
0   2014-07-01 00:00:06
1   2014-07-01 00:00:51
2   2014-07-01 00:01:48
3   2014-07-01 00:02:15
4   2014-07-01 00:03:05
Name: timestamp_first_active, dtype: datetime64[ns]
```

[96]:
```
df_test['day_account_created'] = df_test['date_account_created'].dt.day
df_test['month_account_created'] = df_test['date_account_created'].dt.month
df_test['year_account_created'] = df_test['date_account_created'].dt.year

df_test['day_first_active'] = df_test['timestamp_first_active'].dt.day
df_test['month_first_active'] = df_test['timestamp_first_active'].dt.month
df_test['year_first_active'] = df_test['timestamp_first_active'].dt.year

df_test['day_first_booking'] = df_test['date_first_booking'].dt.day
df_test['month_first_booking'] = df_test['date_first_booking'].dt.month
df_test['year_first_booking'] = df_test['date_first_booking'].dt.year
```

10 Deleting irrelevant Features

[97]:
```
df_test.drop(columns=['date_account_created', 'timestamp_first_active',␣
 ↪'date_first_booking'], inplace=True)
```

[98]:
```
# df_test = df_test.drop(columns=columns_to_drop)
# df_test.drop(columns=['id'], inplace=True)
```

[99]:
```
df_train.drop(columns=['date_first_booking'], inplace=True)
```

11 check for null values

[100]:
```
df_test.isnull().sum()
```

[100]:
```
id                          0
gender                      0
age                     28876
signup_method               0
signup_flow                 0
language                    0
affiliate_channel           0
affiliate_provider          0
first_affiliate_tracked    20
signup_app                  0
first_device_type           0
first_browser               0
day_account_created         0
month_account_created       0
year_account_created        0
day_first_active            0
month_first_active          0
year_first_active           0
day_first_booking           0
month_first_booking         0
year_first_booking          0
dtype: int64
```

[101]:
```
df_test['age'].value_counts
```

[101]:
```
<bound method IndexOpsMixin.value_counts of 0        35.0
1         NaN
2         NaN
3         NaN
4         NaN
         ...
62091    31.0
62092     NaN
62093     NaN
62094     NaN
62095    49.0
Name: age, Length: 62096, dtype: float64>
```

12 Handle Missing Value

[102]:
```
# df_test['age'] = df_test['age'].fillna(df_test['age'].median())
df_test['first_affiliate_tracked'] = df_test['first_affiliate_tracked'].
 ↪fillna(df_test['first_affiliate_tracked'].mode()[0])
```

[103]:
```
df_test['gender'] = df_test['gender'].replace('-unknown-', 'Unknown')
df_test['language'] = df_test['language'].replace('-unknown-', 'Unknown')
```

[104]:
```
mode_signup_method = df_test['signup_method'].mode()[0]

# Replace 'weibo' with the mode in the 'signup_method' column bec is not␣
 ↪present in the df_train
df_test['signup_method'] = df_test['signup_method'].replace('weibo',␣
 ↪mode_signup_method)
```

[105]:
```
df_test.isnull().any().any()
```

[105]:
```
True
```

[106]:
```
df_test['age'].value_counts()
```

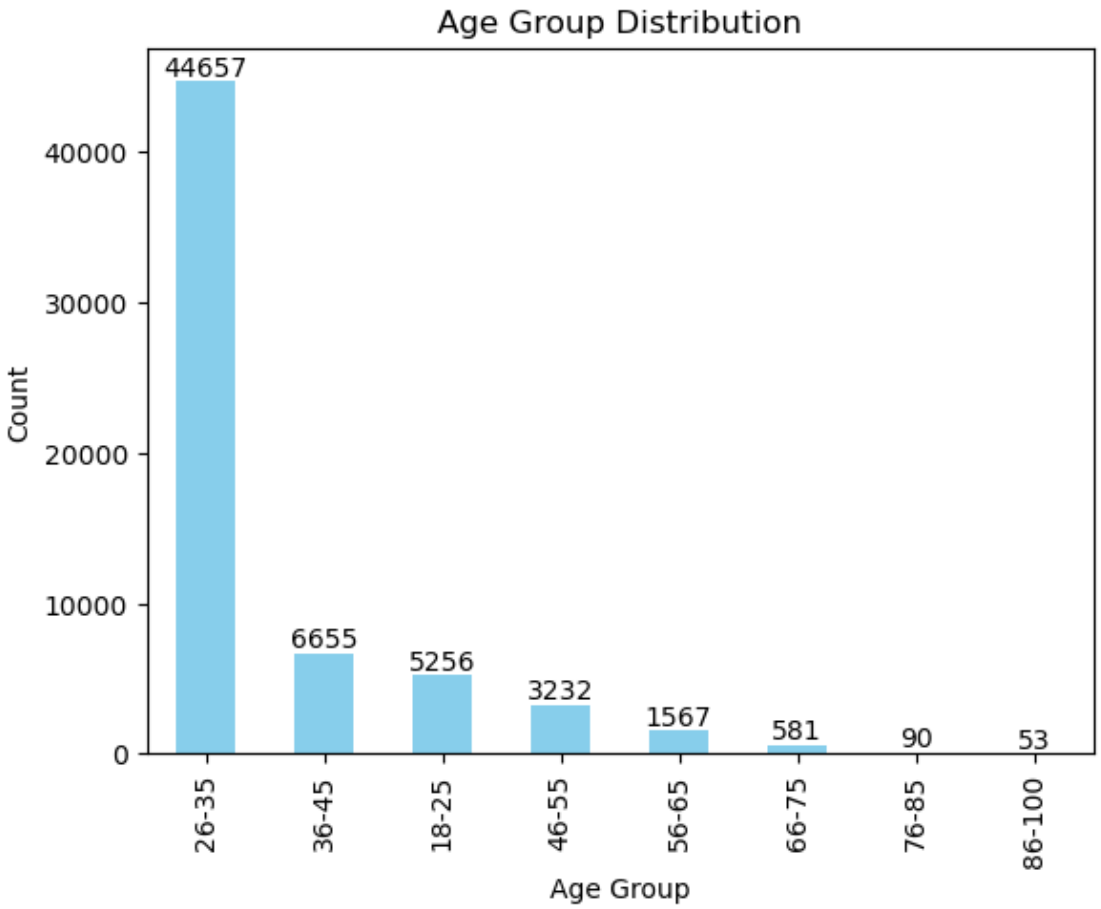
[106]:
```
age
28.0      1748
27.0      1724
26.0      1701
25.0      1690
29.0      1684
          …
1945.0       1
2000.0       1
1938.0       1
1939.0       1
108.0        1
Name: count, Length: 124, dtype: int64
```

[107]:
```
min_age = 18
max_age = 100


df_test['age'] = df_test['age'].apply(lambda x: x if min_age <= x <= max_age␣
 ↪else df_test['age'].median())

bins = [18, 25, 35, 45, 55, 65, 75, 85, 100]
labels = ['18-25', '26-35', '36-45', '46-55', '56-65', '66-75', '76-85',␣
 ↪'86-100']
```

```
df_test['age_group'] = pd.cut(df_test['age'], bins=bins, labels=labels,␣
 ↪right=False)

print(df_test['age_group'].value_counts())
```

```
age_group
26-35    44657
36-45     6655
18-25     5256
46-55     3232
56-65     1567
66-75      581
76-85       90
86-100      53
Name: count, dtype: int64
```

[108]:
```
ax=df_test['age_group'].value_counts().plot(kind='bar', color='skyblue')
plt.title('Age Group Distribution')
plt.xlabel('Age Group')
plt.ylabel('Count')
ax.bar_label(ax.containers[0])

plt.show()
```



[109]:
```
df_test.drop(columns=['age'], inplace=True)
```

[110]:
```
df_test.isna().sum()
```

[110]:
```
id                          0
gender                      0
signup_method               0
signup_flow                 0
language                    0
affiliate_channel           0
affiliate_provider          0
first_affiliate_tracked     0
signup_app                  0
first_device_type           0
first_browser               0
day_account_created         0
month_account_created       0
year_account_created        0
```

```
day_first_active           0
month_first_active         0
year_first_active          0
day_first_booking          0
month_first_booking        0
year_first_booking         0
age_group                  5
dtype: int64
```

[111]:
```
most_frequent_age_group = df_test['age_group'].mode()[0]
df_test['age_group'].fillna(most_frequent_age_group,inplace=True)
```

```
C:\Users\lenovo\AppData\Local\Temp\ipykernel_696\1041072351.py:2: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.
```

```
df_test['age_group'].fillna(most_frequent_age_group,inplace=True)
```

13 Merging Cleaned Testing Data with Sessions Data With Same Steps

[112]:
```
sessions_summary = df_sessions.groupby('id').agg({
    'action': 'count',
    'device_type': 'nunique',
    'secs_elapsed': 'sum'
}).reset_index()

sessions_summary.rename(columns={
    'action': 'num_actions',
    'device_type': 'num_devices',
    'secs_elapsed': 'total_secs'
}, inplace=True)

df_test = df_test.merge(sessions_summary, on='id', how='left')
```

[113]:
```
df_test.isnull().sum()
```

[113]:
```
id                          0
gender                      0
signup_method               0
signup_flow                 0
language                    0
affiliate_channel           0
affiliate_provider          0
first_affiliate_tracked     0
signup_app                  0
first_device_type           0
first_browser               0
day_account_created         0
month_account_created       0
year_account_created        0
day_first_active            0
month_first_active          0
year_first_active           0
day_first_booking           0
month_first_booking         0
year_first_booking          0
age_group                   0
num_actions               428
num_devices               428
total_secs                428
dtype: int64
```

[114]:
```
df_test['num_actions'] = df_test['num_actions'].fillna(df_test['num_actions'].
 ↪median())
df_test['num_devices'] = df_test['num_devices'].fillna(df_test['num_devices'].
 ↪median())
df_test['total_secs'] = df_test['total_secs'].fillna(df_test['total_secs'].
 ↪median())
```

[115]:
```
df_train.to_csv('F:
 ↪\\Andalosia\\airbnb-recruiting-new-user-bookings\\data_clean\\df_train_1.
 ↪csv', index=False)
df_test.to_csv('F:
 ↪\\Andalosia\\airbnb-recruiting-new-user-bookings\\data_clean\\df_test_1.
 ↪csv', index=False)
```

[116]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 213451 entries, 0 to 213450
Data columns (total 24 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   gender                   213451 non-null  object
```

```
 1   signup_method            213451 non-null  object
 2   signup_flow              213451 non-null  int64
 3   language                 213451 non-null  object
 4   affiliate_channel        213451 non-null  object
 5   affiliate_provider       213451 non-null  object
 6   first_affiliate_tracked  213451 non-null  object
 7   signup_app               213451 non-null  object
 8   first_device_type        213451 non-null  object
 9   first_browser            213451 non-null  object
 10  country_destination      213451 non-null  object
 11  day_account_created      213451 non-null  int32
 12  month_account_created    213451 non-null  int32
 13  year_account_created     213451 non-null  int32
 14  day_first_active         213451 non-null  int32
 15  month_first_active       213451 non-null  int32
 16  year_first_active        213451 non-null  int32
 17  day_first_booking        213451 non-null  int32
 18  month_first_booking      213451 non-null  int32
 19  year_first_booking       213451 non-null  int32
 20  age_group                213451 non-null  object
 21  num_actions              213451 non-null  float64
 22  num_devices              213451 non-null  float64
 23  total_secs               213451 non-null  float64
dtypes: float64(3), int32(9), int64(1), object(11)
memory usage: 31.8+ MB
```

[117]:
```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62096 entries, 0 to 62095
Data columns (total 24 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       62096 non-null  object
 1   gender                   62096 non-null  object
 2   signup_method            62096 non-null  object
 3   signup_flow              62096 non-null  int64
 4   language                 62096 non-null  object
 5   affiliate_channel        62096 non-null  object
 6   affiliate_provider       62096 non-null  object
 7   first_affiliate_tracked  62096 non-null  object
 8   signup_app               62096 non-null  object
 9   first_device_type        62096 non-null  object
 10  first_browser            62096 non-null  object
 11  day_account_created      62096 non-null  int32
 12  month_account_created    62096 non-null  int32
 13  year_account_created     62096 non-null  int32
 14  day_first_active         62096 non-null  int32
```

```
 15  month_first_active       62096 non-null  int32
 16  year_first_active        62096 non-null  int32
 17  day_first_booking        62096 non-null  int32
 18  month_first_booking      62096 non-null  int32
 19  year_first_booking       62096 non-null  int32
 20  age_group                62096 non-null  category
 21  num_actions              62096 non-null  float64
 22  num_devices              62096 non-null  float64
 23  total_secs               62096 non-null  float64
dtypes: category(1), float64(3), int32(9), int64(1), object(10)
memory usage: 8.8+ MB
```

[ ]: