
Capston Project 1004

Table of Contents

1. **Introduction**
2. **Problem Statement**
3. **Installing & Importing Libraries**
 - 3.1 **Installing Libraries**
 - 3.2 **Upgrading Libraries**
 - 3.3 **Importing Libraries**
4. **Data Acquisition & Description**
5. **Data Pre-Profiling**
6. **Data Pre-Processing**
7. **Data Post-Profiling**
8. **Exploratory Data Analysis**
9. **Summarization** </br> 9.1 **Conclusion** </br> 9.2 **Actionable Insights** </br>



In []:

1. Introduction

- AccredianTelecom, one of the leading telecom players
 - AccredianTelecom is a Telecom networking service company. which allows users to connect among friends or people.
-

2. Problem Statement

- Which age group is more on AccredianTelecom network
 - To identify certain patterns with respect to how the users are making use of AccredianTelecom depending on State, Age & gender.
 - To understand a user's demographic characteristics based on their mobile usage, geolocation, and mobile device properties
-

3. Installing & Importing Libraries

```
In [1]: !pip install -q datascience
        !pip install -q pandas-profiling
```

```
In [2]: import pandas as pd                                # Importing pandas
        from pandas_profiling import ProfileReport         # Import Pandas Profiling
        #-----
        import numpy as np                                # Importing numpy
        #-----
        import matplotlib.pyplot as plt                   # Importing matplotlib
        import seaborn as sns                             # Importing seaborn
        %matplotlib inline
        #-----
        import scipy as sp

        #pd.set_option('display.max_rows', None)           # Display all rows
        #pd.set_option('display.max_columns', None)        # Display all columns
```

```
C:\Users\Abhishek\AppData\Local\Temp\ipykernel_29836\2561167681.py:2: DeprecationWarning: `import pandas_profiling` is going to be deprecated by April 1st. Please use `import ydata_profiling` instead.
  from pandas_profiling import ProfileReport               # Import Pandas Profiling
as Profiling (To generate Univariate Analysis)
```

4. Data Acquisition & Description

In [3]: `event = pd.read_csv('C:/Users/Abhishek/Downloads/1004 data set/events_data.csv')`

In [4]: `event`

Out[4]:

	event_id	device_id	timestamp	longitude	latitude	city	state
0	2765368	2.973348e+18	2016-05-07 22:52:05	77.225676	28.730140	Delhi	Delhi
1	2955066	4.734221e+18	2016-05-01 20:44:16	88.388361	22.660325	Calcutta	WestBengal
2	605968	-3.264500e+18	2016-05-02 14:23:04	77.256809	28.757906	Delhi	Delhi
3	448114	5.731369e+18	2016-05-03 13:21:16	80.343613	13.153332	Chennai	TamilNadu
4	665740	3.388880e+17	2016-05-06 03:51:05	85.997745	23.842609	Bokaro	Jharkhand
...
3252945	2687452	-1.937028e+18	2016-05-07 23:33:14	73.891597	18.544124	Pune	Maharashtra
3252946	1051580	3.345851e+18	2016-05-03 05:13:30	72.837258	19.018432	Mumbai	Maharashtra
3252947	1316227	-6.406040e+18	2016-05-01 16:03:28	77.235578	28.764065	Delhi	Delhi
3252948	381262	-2.920741e+18	2016-05-05 17:22:36	83.326044	17.765488	Visakhapatnam	AndhraPradesh
3252949	522592	3.212750e+18	2016-05-07 17:34:18	77.308533	9.779918	Kambam	TamilNadu

3252950 rows × 7 columns

In [5]: `event.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3252950 entries, 0 to 3252949
Data columns (total 7 columns):
#   Column      Dtype
---  -----  ---
0   event_id    int64
1   device_id   float64
2   timestamp   object
3   longitude   float64
4   latitude    float64
5   city        object
6   state       object
dtypes: float64(3), int64(1), object(3)
memory usage: 173.7+ MB
```

In [6]: `event.isna().sum()`

```
Out[6]: event_id      0
        device_id    453
        timestamp     0
        longitude     423
        latitude      423
        city          0
        state         377
        dtype: int64
```

Observation In events data

- Events Data has 7 columns and 3252950 rows
- 5 continous and 2 categorical variable
- Device id beeing considered as identifier
- Device Id, Lat, Longituted and state has missing values

```
In [7]: target_states = ['AndhraPradesh', 'Pondicherry', 'Mizoram', 'AndamanandNicobarIslands']

# Filter the data based on the target states
filtered_eventdata = event[event['state'].isin(target_states)]

# Print the filtered data
print(filtered_eventdata)
```

	event_id	device_id	timestamp	longitude	latitude	\
5	1078723	-5.124242e+17	2016-05-02 02:21:20	83.398244	17.768149	
7	280014	-8.879644e+18	2016-05-05 13:06:01	78.155397	16.390327	
12	2334601	-6.018833e+17	2016-05-05 11:17:48	83.380111	17.828583	
32	2064864	-2.764521e+18	2016-05-03 23:58:20	83.315014	17.825280	
48	1341801	4.986891e+18	2016-05-07 15:24:58	83.324339	17.778384	
...
3252915	2486328	-2.943655e+18	2016-05-03 19:09:18	83.371765	17.800655	
3252922	2905298	5.141558e+18	2016-05-04 10:16:36	83.339048	17.751325	
3252930	2264739	-3.616572e+18	2016-05-07 12:45:06	83.945946	18.336945	
3252943	1045746	-1.370786e+18	2016-05-01 00:15:22	91.911920	25.642058	
3252948	381262	-2.920741e+18	2016-05-05 17:22:36	83.326044	17.765488	

	city	state
5	Visakhapatnam	AndhraPradesh
7	Wanparti	AndhraPradesh
12	Visakhapatnam	AndhraPradesh
32	Visakhapatnam	AndhraPradesh
48	Visakhapatnam	AndhraPradesh
...
3252915	Visakhapatnam	AndhraPradesh
3252922	Visakhapatnam	AndhraPradesh
3252930	Srikakulam	AndhraPradesh
3252943	Shillong	Meghalaya
3252948	Visakhapatnam	AndhraPradesh

[329125 rows x 7 columns]

```
In [8]: filtered_eventdata.isna().sum()
```

```
Out[8]: event_id      0
        device_id    69
        timestamp    0
        longitude    63
        latitude     63
        city         0
        state        0
        dtype: int64
```

```
In [9]: filtered_eventdata["longitude"]=filtered_eventdata["longitude"].fillna(83.368896)
        filtered_eventdata["latitude"] =filtered_eventdata["latitude"].fillna(17.798819)
        filtered_eventdata['device_id'].fillna(filtered_eventdata['device_id'].mode()[0], i
```

C:\Users\Abhishek\AppData\Local\Temp\ipykernel_29836\1949865122.py:1: SettingWithCo
pyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
        filtered_eventdata["longitude"]=filtered_eventdata["longitude"].fillna(83.368896)
C:\Users\Abhishek\AppData\Local\Temp\ipykernel_29836\1949865122.py:2: SettingWithCo  
pyWarning:
```

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
        filtered_eventdata["latitude"] =filtered_eventdata["latitude"].fillna(17.798819)
C:\Users\Abhishek\AppData\Local\Temp\ipykernel_29836\1949865122.py:3: SettingWithCo  
pyWarning:
```

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
        filtered_eventdata['device_id'].fillna(filtered_eventdata['device_id'].mode()[0],  
        inplace=True)
```

```
In [10]: filtered_eventdata.isna().sum()
```

```
Out[10]: event_id      0
        device_id     0
        timestamp     0
        longitude     0
        latitude     0
        city         0
        state        0
        dtype: int64
```

```
In [ ]:
```

```
In [11]: mob = pd.read_excel('C:/Users/Abhishek/Downloads/1004 data set/phon.xlsx')
```

```
In [12]: mob.isna().sum()
```

```
Out[12]: Unnamed: 0      0
          device_id      0
          phone_brand     0
          device_model     0
          dtype: int64
```

```
In [13]: mob
```

```
Out[13]:
```

	Unnamed: 0	device_id	phone_brand	device_model
0	0	1877775838486906112	vivo	Y13
1	1	-3766087376657243136	小米	V183
2	2	-6238937574958216192	OPPO	R7s
3	3	8973197758510676992	三星	A368t
4	4	-2015528097870763008	小米	红米Note2
...
87721	87721	-4961458925928573952	华为	荣耀畅玩4X
87722	87722	-8819817317449262080	华为	荣耀6
87723	87723	-3358291377416934912	华为	荣耀畅玩4
87724	87724	3282788959750982144	小米	MI 2
87725	87725	2491639413207285760	酷比	M1

87726 rows × 4 columns

Observation In Mobile data

- Events Data has 4 columns and 87726 rows
- 2 continuous and 2 categorical variable
- Device id beeing considered as identifier
- No Missing values found

```
In [14]: age = pd.read_excel('C:/Users/Abhishek/Downloads/1004 data set/age.xlsx')
```

```
In [15]: age
```

```
Out[15]:
```

	Unnamed: 0	device_id	gender	age	group
0	0	-8076087639492063232	M	35	M32-38
1	1	-2897161552818059776	M	35	M32-38
2	2	-8260683887967679488	M	35	M32-38
3	3	-4938849341048082432	M	30	M29-31
4	4	245133531816851904	M	30	M29-31
...
74640	74640	4682031842235089920	M	30	M29-31
74641	74641	-9178703742877135872	M	30	M29-31
74642	74642	180946546684162304	M	20	M22-
74643	74643	1390702386071992064	M	37	M32-38
74644	74644	89181010588227344	M	25	M23-26

74645 rows × 5 columns

Observation In Age data

- Events Data has 5 columns and 74645 rows
- 3 continous and 2 categorical variable
- Device id beeing considered as identifier
- Noo null values found

meargging the data frame

```
In [16]: EveAge=pd.merge(left=filtered_eventdata,right=age,on="device_id",how="left")
```

```
In [17]: EveAge
```

Out[17]:

	event_id	device_id	timestamp	longitude	latitude	city	state
0	1078723	-5.124242e+17	2016-05-02 02:21:20	83.398244	17.768149	Visakhapatnam	AndhraPradesh
1	280014	-8.879644e+18	2016-05-05 13:06:01	78.155397	16.390327	Wanparti	AndhraPradesh
2	2334601	-6.018833e+17	2016-05-05 11:17:48	83.380111	17.828583	Visakhapatnam	AndhraPradesh
3	2064864	-2.764521e+18	2016-05-03 23:58:20	83.315014	17.825280	Visakhapatnam	AndhraPradesh
4	1341801	4.986891e+18	2016-05-07 15:24:58	83.324339	17.778384	Visakhapatnam	AndhraPradesh
...
329120	2486328	-2.943655e+18	2016-05-03 19:09:18	83.371765	17.800655	Visakhapatnam	AndhraPradesh
329121	2905298	5.141558e+18	2016-05-04 10:16:36	83.339048	17.751325	Visakhapatnam	AndhraPradesh
329122	2264739	-3.616572e+18	2016-05-07 12:45:06	83.945946	18.336945	Srikakulam	AndhraPradesh
329123	1045746	-1.370786e+18	2016-05-01 00:15:22	91.911920	25.642058	Shillong	Meghalaya
329124	381262	-2.920741e+18	2016-05-05 17:22:36	83.326044	17.765488	Visakhapatnam	AndhraPradesh

329125 rows × 11 columns

In [18]: dataall=pd.merge(left=EveAge,right=mob,on="device_id",how="left")

In [19]: dataall

Out[19]:

	event_id	device_id	timestamp	longitude	latitude	city	state
0	1078723	-5.124242e+17	2016-05-02 02:21:20	83.398244	17.768149	Visakhapatnam	AndhraPradesh
1	280014	-8.879644e+18	2016-05-05 13:06:01	78.155397	16.390327	Wanparti	AndhraPradesh
2	2334601	-6.018833e+17	2016-05-05 11:17:48	83.380111	17.828583	Visakhapatnam	AndhraPradesh
3	2064864	-2.764521e+18	2016-05-03 23:58:20	83.315014	17.825280	Visakhapatnam	AndhraPradesh
4	1341801	4.986891e+18	2016-05-07 15:24:58	83.324339	17.778384	Visakhapatnam	AndhraPradesh
...
329120	2486328	-2.943655e+18	2016-05-03 19:09:18	83.371765	17.800655	Visakhapatnam	AndhraPradesh
329121	2905298	5.141558e+18	2016-05-04 10:16:36	83.339048	17.751325	Visakhapatnam	AndhraPradesh
329122	2264739	-3.616572e+18	2016-05-07 12:45:06	83.945946	18.336945	Srikakulam	AndhraPradesh
329123	1045746	-1.370786e+18	2016-05-01 00:15:22	91.911920	25.642058	Shillong	Meghalaya
329124	381262	-2.920741e+18	2016-05-05 17:22:36	83.326044	17.765488	Visakhapatnam	AndhraPradesh

329125 rows × 14 columns

In [20]: dataall.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 329125 entries, 0 to 329124
Data columns (total 14 columns):
#   Column          Non-Null Count  Dtype
---  -
0   event_id        329125 non-null  int64
1   device_id       329125 non-null  float64
2   timestamp       329125 non-null  object
3   longitude       329125 non-null  float64
4   latitude        329125 non-null  float64
5   city            329125 non-null  object
6   state           329125 non-null  object
7   Unnamed: 0_x    240842 non-null  float64
8   gender          240842 non-null  object
9   age             240842 non-null  float64
10  group           240842 non-null  object
11  Unnamed: 0_y    240842 non-null  float64
12  phone_brand     240842 non-null  object
13  device_model    240842 non-null  object
dtypes: float64(6), int64(1), object(7)
memory usage: 37.7+ MB
```

In [21]: `dataall.describe()`

Out[21]:

	event_id	device_id	longitude	latitude	Unnamed: 0_x	age
count	3.291250e+05	3.291250e+05	329125.000000	329125.000000	240842.000000	240842.000000
mean	1.626038e+06	9.478685e+16	82.467417	17.596729	35900.501765	31.152527
std	9.404014e+05	5.341850e+18	2.215380	1.537158	20590.376326	9.825075
min	5.985000e+03	-9.222173e+18	12.567400	10.941103	33.000000	12.000000
25%	8.114810e+05	-4.718484e+18	82.226211	17.744599	18233.000000	24.000000
50%	1.627460e+06	-2.849262e+16	83.339626	17.774964	36599.000000	29.000000
75%	2.440367e+06	4.883859e+18	83.372565	17.807230	51906.000000	35.000000
max	3.252943e+06	9.220807e+18	92.859813	41.871900	74595.000000	88.000000

5. Data Pre-Profiling

- Handling missing values in device id, replacing with mode.
- Handling missing values in Latitude and longitude replacing with mode.

In [22]:

```

null_frame = pd.DataFrame(index=dataall.columns.values)
null_frame['Null Frequency']=dataall.isnull().sum().values
percent=dataall.isnull().sum().values/dataall.shape[0]
null_frame["Missing%"]=np.round(percent,decimals=4)*100

```

In [23]: `null_frame.transpose()`

Out[23]:

	event_id	device_id	timestamp	longitude	latitude	city	state	Unnamed: 0_x	gender
Null Frequency	0.0	0.0	0.0	0.0	0.0	0.0	0.0	88283.00	88283.00
Missing%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	26.82	26.82

In [25]: `dataall.isna().sum()`

```
Out[25]: event_id      0
         device_id    0
         timestamp    0
         longitude     0
         latitude     0
         city         0
         state        0
         Unnamed: 0_x  88283
         gender       88283
         age         88283
         group       88283
         Unnamed: 0_y  88283
         phone_brand  88283
         device_model 88283
         dtype: int64
```

6. Data Pre-Processing

- This section is emphasised on performing data manipulation over unstructured data for further processing and analysis.
- To modify unstructured data to structured data you need to verify and manipulate the integrity of the data by:
 - Handling missing data,
 - Handling redundant data,
 - Handling inconsistent data,
 - Handling outliers,
 - Handling typos

```
In [26]: dataa11["device_id"] = abs(dataa11["device_id"])
```

```
In [27]: dataa11 = dataa11.drop(["Unnamed: 0_x", "Unnamed: 0_y"], axis=1)
```

```
In [28]: dataa11["gender"].fillna(dataa11['gender'].mode()[0], inplace=True)

dataa11["phone_brand"].fillna(dataa11['phone_brand'].mode()[0], inplace=True)
dataa11["device_model"].fillna(dataa11['device_model'].mode()[0], inplace=True)

dataa11["age"].fillna(dataa11['age'].mode()[0], inplace=True)
dataa11["group"].fillna(dataa11['group'].mode()[0], inplace=True)
```

```
In [29]: dataa11.isna().sum()
```

```
Out[29]: event_id      0
         device_id    0
         timestamp    0
         longitude     0
         latitude     0
         city         0
         state        0
         gender       0
         age          0
         group        0
         phone_brand  0
         device_model  0
         dtype: int64
```

```
In [30]: data=dataall.copy()
```

7. Data Post-Profiling

```
In [31]: data.isna().sum()
```

```
Out[31]: event_id      0
         device_id    0
         timestamp    0
         longitude     0
         latitude     0
         city         0
         state        0
         gender       0
         age          0
         group        0
         phone_brand  0
         device_model  0
         dtype: int64
```

```
In [32]: data["phone_brand"].unique()
```

```
Out[32]: array(['小米', '三星', '华为', '波导', 'OPPO', '酷派', '魅族', '天语', '优米', 'vivo',
                'TCL', '维图', '一加', '夏新', 'HTC', '百立丰', '联想', '美图', '大Q', '海信',
                '酷比魔方', '锤子', '中国移动', 'LG', '朵唯', '乐视', '奇酷', '爱派尔', '努比亚',
                '欧博信', '语信', '小杨树', '聆韵', 'ZUK', '康佳', '富可视', '诺基亚', '欧新', '酷比', '黑米',
                '奥克斯', '沃普丰', '欧奇', 'LOGO', '优购', '梦米', '纽曼', '糖葫芦', '西米', '酷珀',
                '海尔', '邦华', '华硕', '乡米', '摩托罗拉', '诺亚信', '米歌'], dtype=object)
```

「蛇姑婆」——「C...」

```
'糖葫芦': 'Sugar',
'亿通': 'Yitong',
'欧新': 'New European',
'米奇': 'Mickey',
'酷比魔方': 'Coolbite',
'蓝魔': 'Blue demon',
'小杨树': 'Small poplar tree',
'贝尔丰': 'Bellferta',
'糯米': 'Sticky rice',
'米歌': 'Rice song',
'E人E本': 'E people e',
'西米': 'Sago',
'大Q': 'Large Q',
'台电': 'Taipower',
'飞利浦': 'Philips',
'唯米': 'Rice',
'大显': 'Greatly',
'长虹': 'Changhong',
'维图': 'Vitamin',
'青橙': 'Orange',
'本为': 'This is',
'虾米': 'Shrimp',
'夏新': 'Xia Xin',
'帷幄': 'Curtain',
'百加': 'Hundred and Maca',
'SUGAR': 'SUGAR',
'欧奇': 'Oichi',
'世纪星': 'Century star',
'智镁': 'Magnesium',
'欧比': 'Obi',
'基伍': 'Foundation',
'飞秒': 'Femondo',
'德赛': 'Virtue',
'易派': 'Easily',
'谷歌': 'Google',
'金星数码': 'Venus Digital',
'广信': 'Widely believed',
'诺亚信': 'Noah',
'MIL': 'THOUSAND',
'白米': 'White rice',
'大可乐': 'Cola',
'宝捷讯': 'Baoxun',
'优语': 'Excellent language',
'首云': 'Shouyun',
'瑞米': 'Ryme',
'瑞高': 'Ruigao',
'沃普丰': 'Walpone',
'摩乐': 'Caravan',
'鲜米': 'Fresh rice',
'凯利通': 'Kellytong',
'唯比': 'Only',
'欧沃': 'Owa',
'丰米': 'Rich rice',
'恒宇丰': 'Hengyufeng',
'奥克斯': 'Oaks',
'西门子': 'Siemens',
'欧乐迪': 'Oletdi',
'PPTV': 'PPTV'}
```

```
In [34]: for index, row in data.iterrows():
In [35]: data["phone_brand"].unique()
Out[35]: if phone_brand in replacement_mapping:
data.at[index, 'phone_brand'] = replacement_mapping[phone_brand]
meizu, cianyu, omniui, vivo, icl, vitamin, one plus,
'Xia Xin', 'HTC', 'Bai Lifeng', 'Lenovo', 'meitu', 'Large Q',
'Hisense', 'Coolbite', 'hammer', 'zhongfu mobile', 'LG', 'Duowei',
'LeEco', 'Cool', 'Aipaer', 'Nubia', 'Hakuhin in Europe',
'Linguistic', 'Small poplar tree', 'Listen to the rhyme', 'ZUK',
'Konka', 'Richness', 'Nokia', 'New European', 'Cooler',
'black rice', 'Oaks', 'Walpone', 'Oichi', 'LOGO',
'Premium purchase', 'Dream rice', 'Newman', 'Sugar', 'Sago',
'Haier', 'Banghua', 'Asus', 'Villain', 'Motorola', 'Noah',
'Rice song'], dtype=object)
```

```
In [36]: data.head()
```

```
Out[36]:
```

	event_id	device_id	timestamp	longitude	latitude	city	state	gender
0	1078723	5.124242e+17	2016-05-02 02:21:20	83.398244	17.768149	Visakhapatnam	AndhraPradesh	M
1	280014	8.879644e+18	2016-05-05 13:06:01	78.155397	16.390327	Wanparti	AndhraPradesh	M
2	2334601	6.018833e+17	2016-05-05 11:17:48	83.380111	17.828583	Visakhapatnam	AndhraPradesh	M
3	2064864	2.764521e+18	2016-05-03 23:58:20	83.315014	17.825280	Visakhapatnam	AndhraPradesh	M
4	1341801	4.986891e+18	2016-05-07 15:24:58	83.324339	17.778384	Visakhapatnam	AndhraPradesh	M

```
In [37]: data.dtypes
```

```
Out[37]: event_id      int64
device_id    float64
timestamp     object
longitude     float64
latitude     float64
city          object
state         object
gender        object
age           float64
group         object
phone_brand   object
device_model  object
dtype: object
```

```
In [38]: data.to_csv("C:\\Users\\Abhishek\\Downloads\\readyforEDA.csv")
```

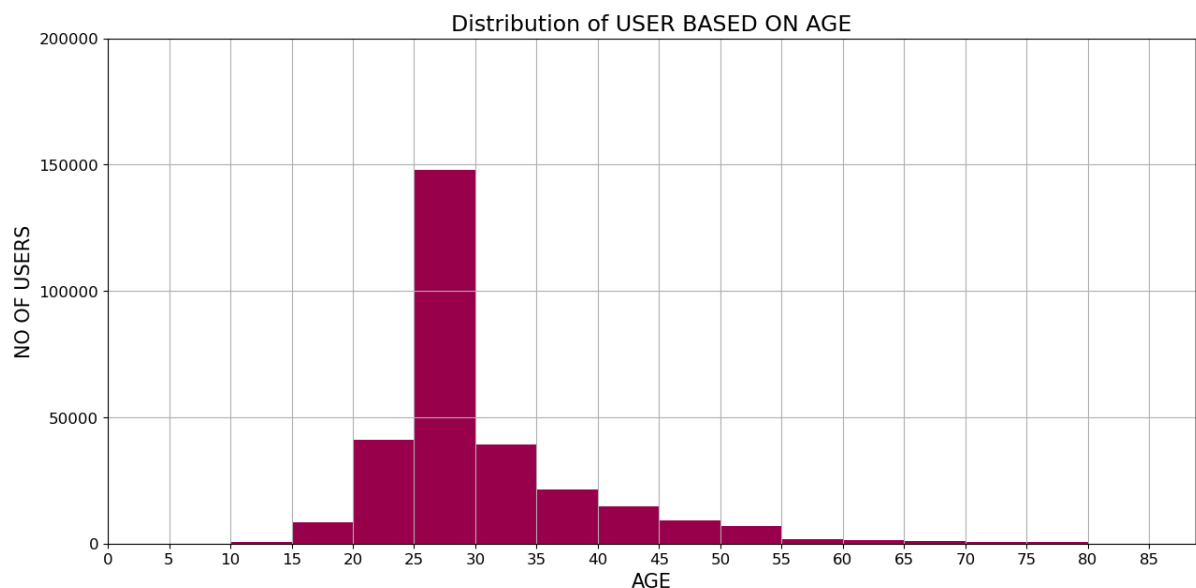
Shortlisting data for top 10 brand only

```
In [39]: targetbrand = ["Xiǎomǐ", "Samsung", "Huawei", "OPPO", "vivo ", "meizu", "Coolpad", "HTC", "  
# Filter the data for top 10 brand  
data_top10brand = data[data['phone_brand'].isin(targetbrand)]
```

8. Exploratory Data Analysis

- This section is emphasised on asking the right questions and perform analysis using the data.
- Note that there is no limit how deep you can go, but make sure not to get distracted from right track.

```
In [63]: figure=plt.figure(figsize=[15,7])  
data_top10brand['age'].plot.hist(bins=np.arange(10, 90, 5),color="#99004C",grid=True)  
  
plt.xlabel("AGE",size=15)  
plt.ylabel("NO OF USERS",size=15)  
plt.title("Distribution of USER BASED ON AGE",size=17)  
  
plt.xticks(ticks=np.arange(0, 90, 5), size=12)  
plt.yticks(ticks=np.arange(0, 250000, 50000), size=12)  
plt.show()
```



- Based on comprehensive data, we have found that the majority of our users fall within the age range of 20 to 35.

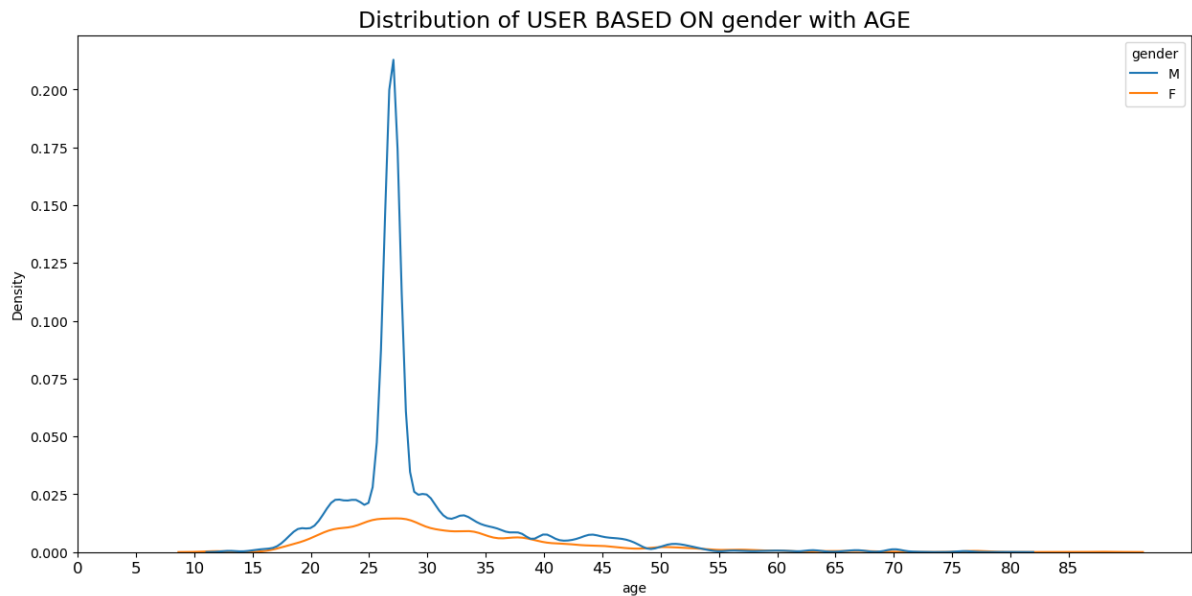

```
In [64]: figure=plt.figure(figsize=[15,7])

sns.kdeplot(data=data_top10brand, x="age", hue="gender")

plt.title("Distribution of USER BASED ON gender with AGE",size=17)

plt.xticks(ticks=np.arange(0, 90, 5), size=12)

plt.show()
```



- Male users are More as compared to female users

1.Distribution of Users(device_id) across States

```
In [65]: figure = plt.figure(figsize=[15, 7])

data.groupby(by=['state'])['device_id'].count().plot.bar()

plt.xticks(size=12, rotation=0)

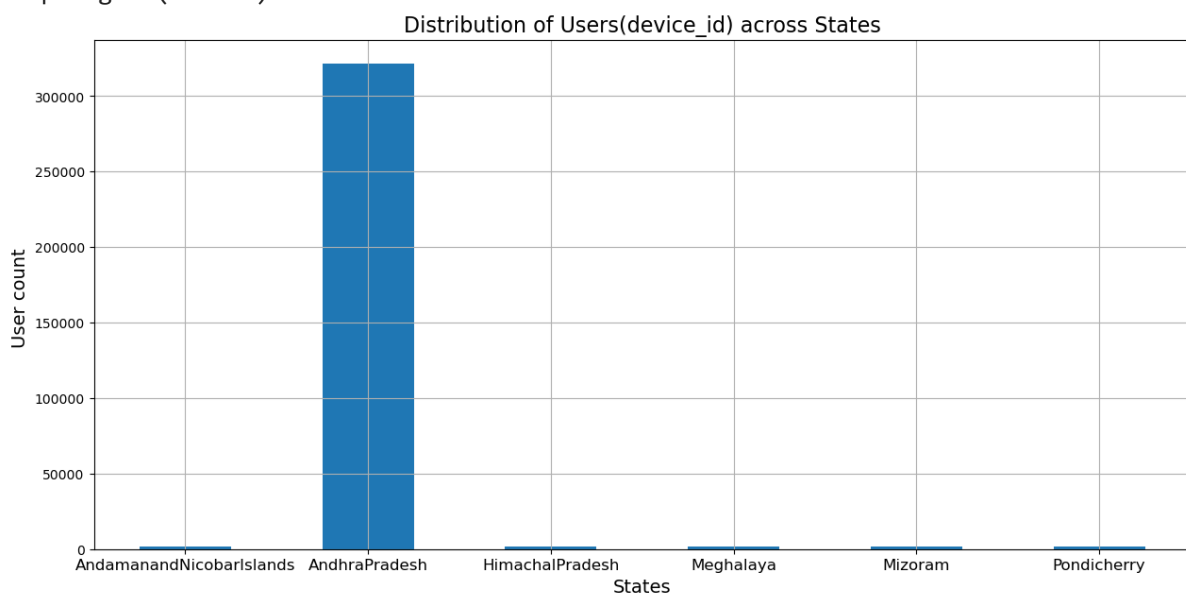
plt.xlabel(xlabel='States', size=14)
plt.ylabel(ylabel='User count ', size=14)

plt.title(label='Distribution of Users(device_id) across States', size=16)
plt.grid(b=True)
plt.show()

plt.show()
```

C:\Users\Abhishek\AppData\Local\Temp\ipykernel_17940\889067946.py:14: MatplotlibDeprecationWarning: The 'b' parameter of grid() has been renamed 'visible' since Matplotlib 3.5; support for the old name will be dropped two minor releases later.

```
plt.grid(b=True)
```



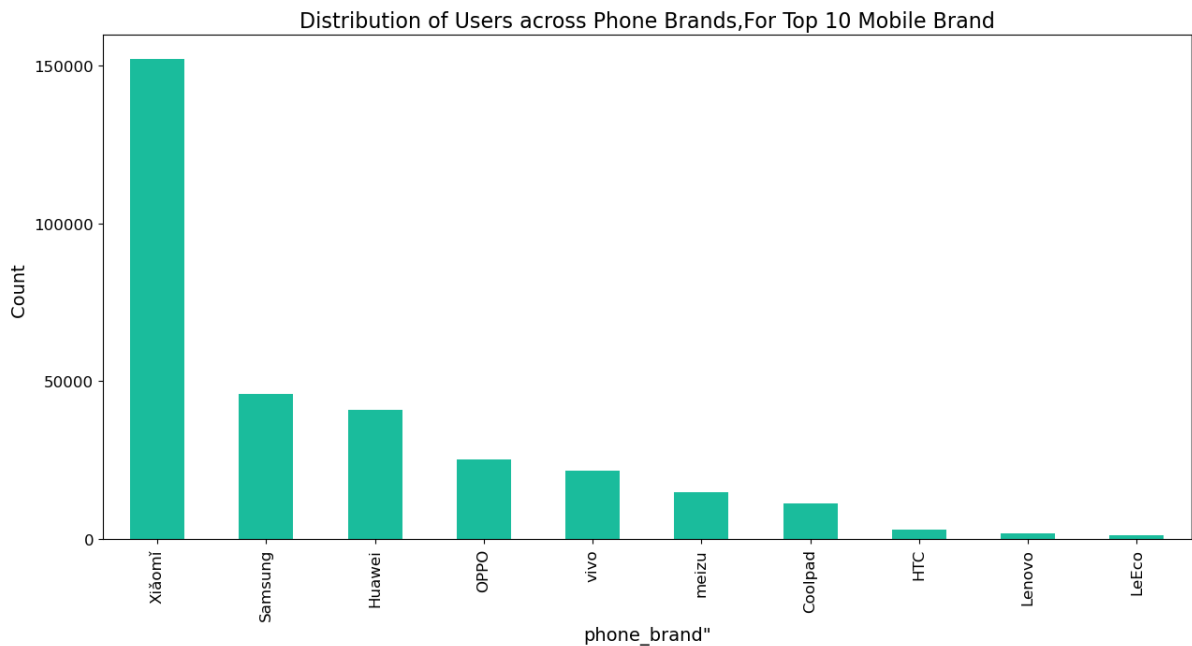
- Andhra Pradesh, a state in India, has a significant number of users.
- This slide will provide an overview of the user statistics in Andhra Pradesh

2. Distribution of Users across Phone Brands(Consider only 10 Most used Phone Brands).

```
In [66]: figure = plt.figure(figsize=[15, 7])
data["phone_brand"].value_counts()[0:10].plot.bar(color='#1ABC9C',

)

plt.xticks(size=12, rotation=90)
plt.yticks(ticks=np.arange(0, 170000, 50000), size=12)
plt.xlabel(xlabel='phone_brand', size=14)
plt.ylabel(ylabel='Count ', size=14, rotation=90
)
plt.title(label='Distribution of Users across Phone Brands,For Top 10 Mobile Brand')
plt.show()
```



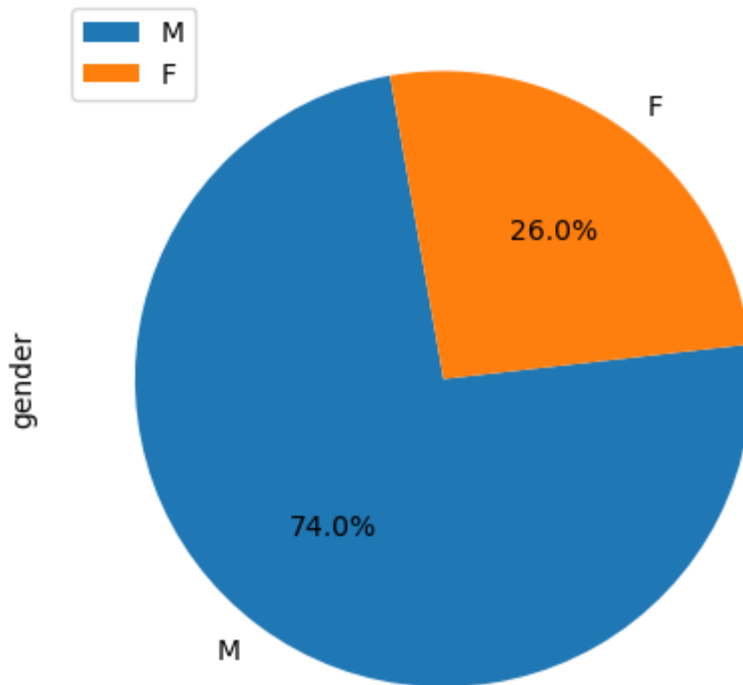
- The global smartphone market has witnessed significant changes in recent years, with consumers embracing a variety of brands and models.
- Highlight Xiaomi, Samsung, and Huawei as the leading brands.
- Xiaomi - Rising in Popularity

3 Distribution of Users across Gender.

```
In [67]: figure = plt.figure(figsize=[5, 5])
data['gender'].value_counts().plot.pie(autopct='%3.1f%%', startangle=100, legend=True
#% value autopct='%3.1f%%' shows the % value in pie chart

plt.title(label='Distribution of Users across Gender', size=16,color='black')#Label
plt.show() # Dispaly the output by rendering visual on the screen
```

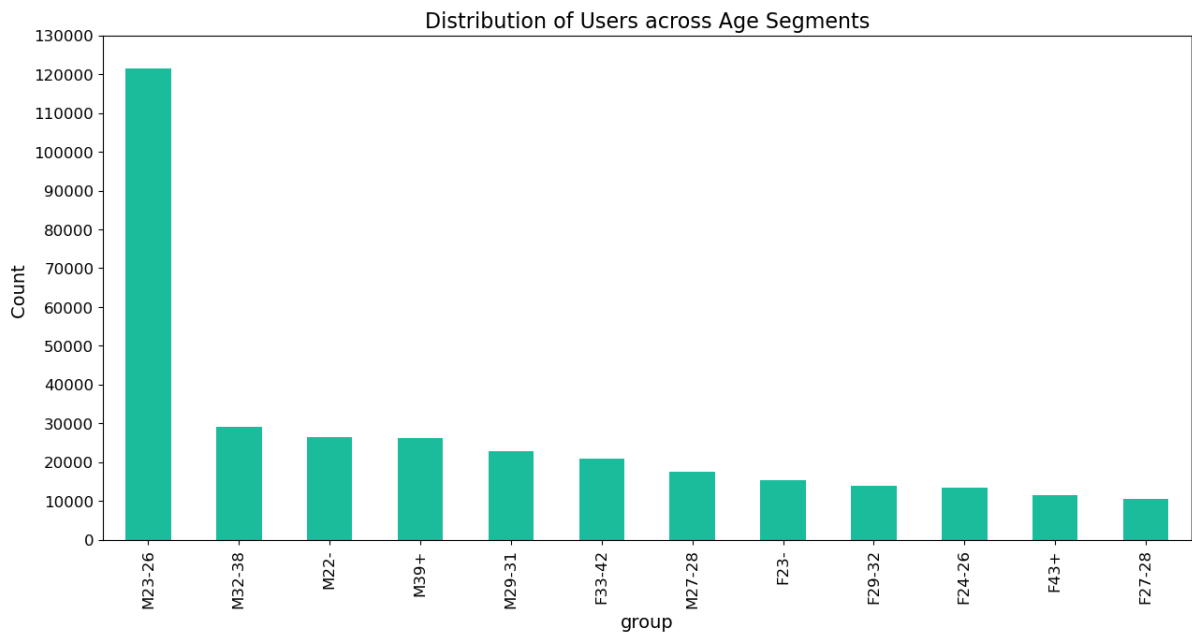
Distribution of Users across Gender



- The user base consists of 74% male users and 26% female users.
- This data indicates a slight majority of male users

4. Distribution of Users across Age Segments.

```
In [68]: figure = plt.figure(figsize=[15, 7])
data["group"].value_counts().plot.bar(color='#1ABC9C')
plt.xticks(size=12, rotation=90)
plt.yticks(ticks=np.arange(0, 140000, 10000), size=12)
plt.xlabel(xlabel='group', size=14)
plt.ylabel(ylabel='Count ', size=14, rotation=90)
plt.title(label='Distribution of Users across Age Segments', size=16)
plt.show()
```



- Understanding the user demographics is crucial for effective targeting and tailoring of products and services.
- The age group M23-36 exhibits the highest number of male users.
- The age group F33-42 exhibits the highest number of Female users

In []:

5. Distribution of Phone Brands(Consider only 10 Most used Phone Brands) for each Age Segment, State, Gender.

Distribution of Top 10 Phone Brands for each Age Segment

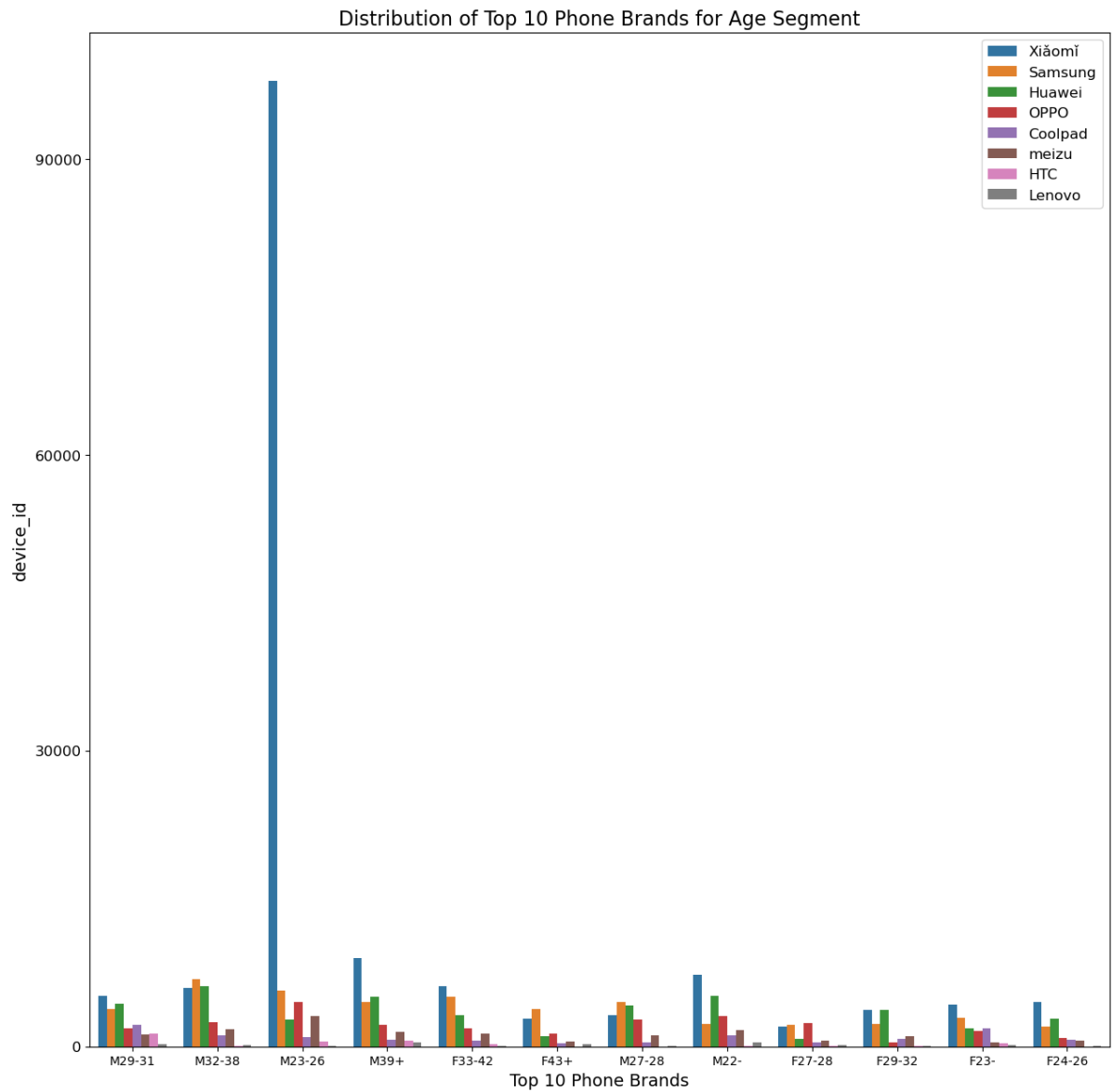
```
In [69]: figure = plt.figure(figsize=[15,15])
sns.countplot(data=data_top10brand, x="group", hue="phone_brand")

plt.yticks(ticks=np.arange(0, 100000, 30000), size=12)
plt.xlabel(xlabel='Top 10 Phone Brands', size=14)

plt.ylabel(ylabel='device_id', size=14)

plt.title(label='Distribution of Top 10 Phone Brands for Age Segment', size=16)
plt.legend(fontsize=12)

plt.show()
```



- Xiaomi Brand is Rising in Popularity FOR the age group
M29-31,M32-26,M39+,F33-42,M22,F29-32,F23-,F24-25
- Samsung brand is Rising in Popularity FOR the age group M32-38,F43+,m27-28,f27-28.

```
In [70]: data_top10brand.groupby(by=["group", 'phone_brand'])['device_id'].count()
```

```
Out[70]: group  phone_brand
F23-   Coolpad      1835
       HTC         299
       Huawei     1825
       Lenovo      162
       OPPO      1535
       ...
M39+   Lenovo       396
       OPPO      2224
       Samsung    4481
       Xiaomi     8959
       meizu      1524
Name: device_id, Length: 96, dtype: int64
```

```
In [71]: pd.set_option('display.max_rows', None) # Display all rows
pd.set_option('display.max_columns', None) # Display all columns
```

Distribution of Top 10 Phone Brands for each State

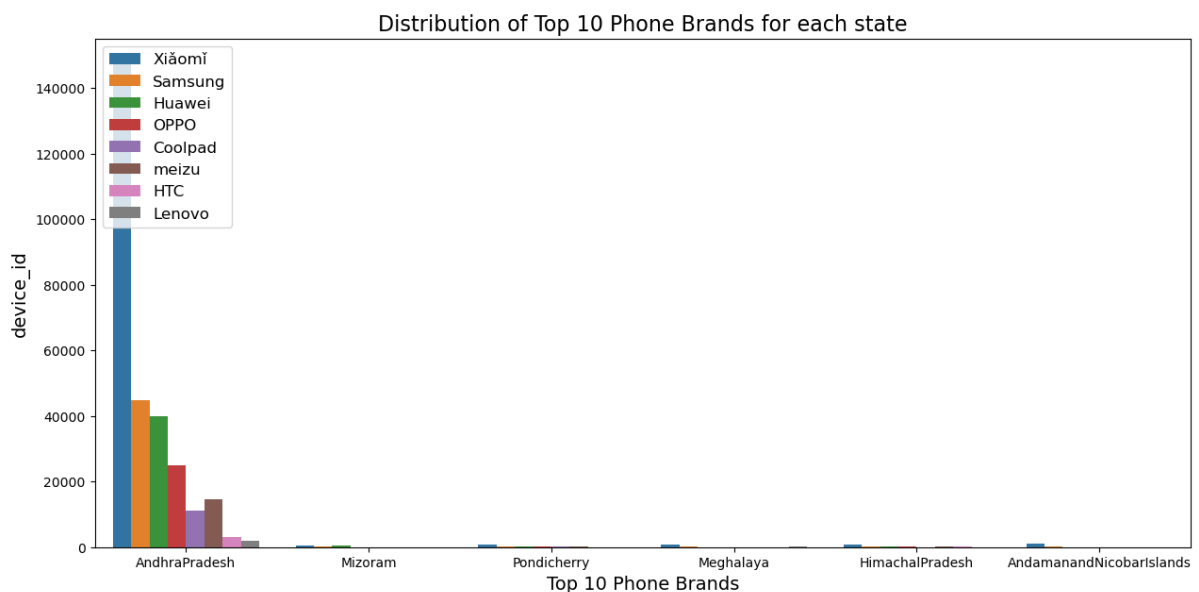
```
In [72]: figure = plt.figure(figsize=[15, 7])
sns.countplot(data=data_top10brand, x="state", hue="phone_brand")
plt.yticks()

plt.xlabel(xlabel='Top 10 Phone Brands', size=14)

plt.ylabel(ylabel='device_id', size=14)

plt.title(label='Distribution of Top 10 Phone Brands for each state', size=16)
plt.legend(fontsize=12)

plt.show()
```



- Xiaomi is Rising in Popularity in each State

Distribution of Top 10 Phone Brands for each Gender

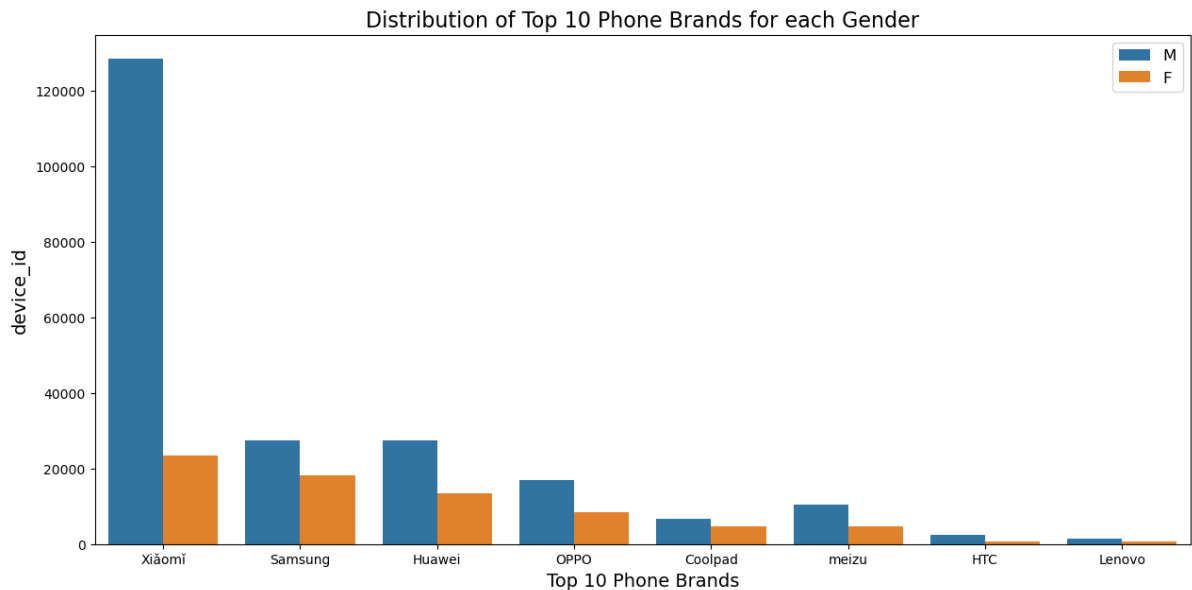
```
In [73]: figure = plt.figure(figsize=[15, 7])
sns.countplot(data=data_top10brand, x="phone_brand", hue="gender")
plt.yticks()

plt.xlabel(xlabel='Top 10 Phone Brands', size=14)

plt.ylabel(ylabel='device_id', size=14)

plt.title(label='Distribution of Top 10 Phone Brands for each Gender', size=16)
plt.legend(fontsize=12)

plt.show()
```



- Male users are IN MEJORITY for variety of brands.
- Xiaomi - Rising in Popularity.

```
In [74]: data_top10brand.groupby(by=['phone_brand', "gender"]['device_id'].count()
```

```
Out[74]: phone_brand  gender
Coolpad          F          4688
              M          6593
HTC              F           641
              M          2545
Huawei            F         13372
              M         27530
Lenovo           F           670
              M          1351
OPPO             F          8498
              M         16885
Samsung          F         18307
              M         27568
Xiǎomǐ           F         23484
              M        128489
meizu            F          4577
              M         10387
Name: device_id, dtype: int64
```

```
In [ ]:
```

6. Distribution of Gender for each State, Age Segment, and Phone Brand(Consider only 10 Most Used Phone Brands).

Distribution of Gender for each State For Top 10 Most Used Phone Brands

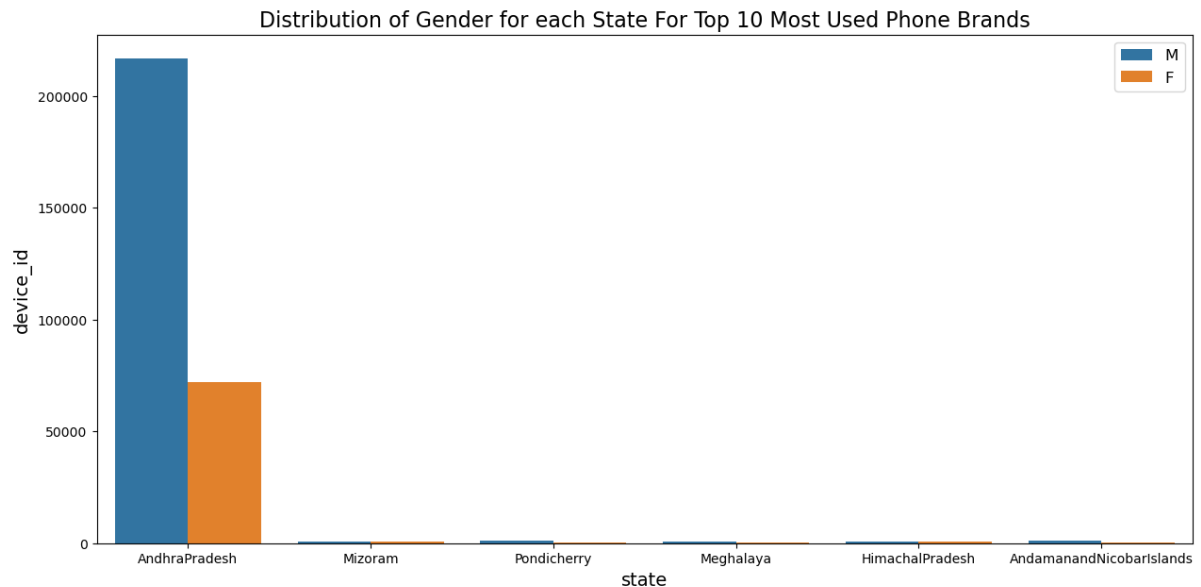

```
In [75]: figure = plt.figure(figsize=[15, 7])
sns.countplot(data=data_top10brand, x="state", hue="gender")
plt.yticks()

plt.xlabel(xlabel='state', size=14)

plt.ylabel(ylabel='device_id', size=14)

plt.title(label='Distribution of Gender for each State For Top 10 Most Used Phone B
plt.legend(fontsize=12)

plt.show()
```



Distribution of Gender for each Age Group for Top 10 Most Used Phone Brands

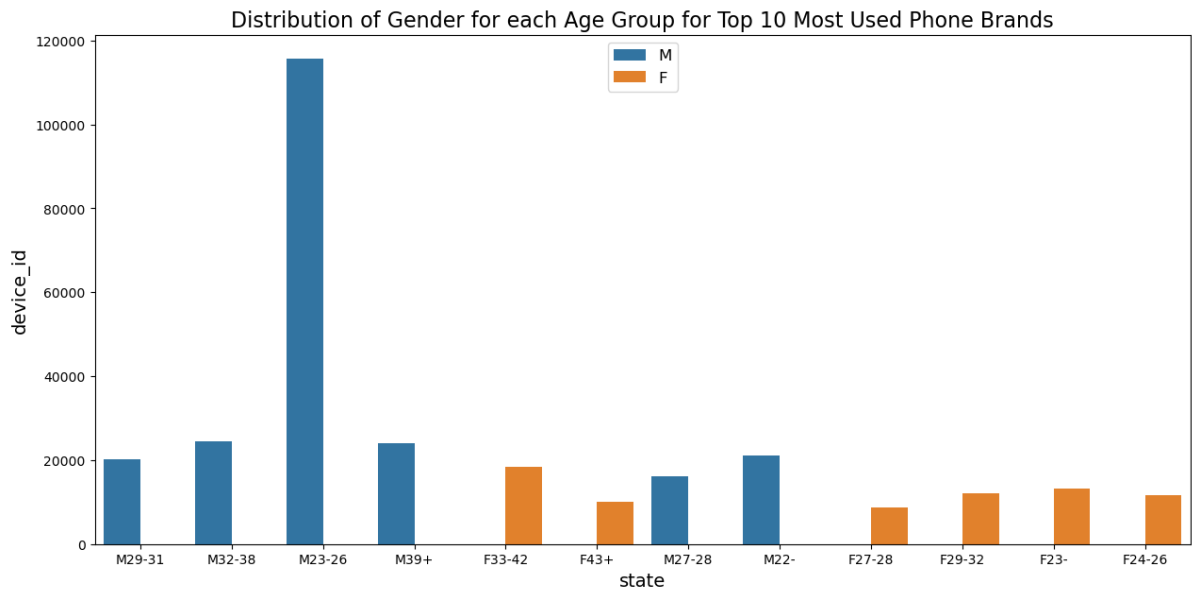
```
In [76]: figure = plt.figure(figsize=[15, 7])
sns.countplot(data=data_top10brand, x="group", hue="gender")
plt.yticks()

plt.xlabel(xlabel='state', size=14)

plt.ylabel(ylabel='device_id', size=14)

plt.title(label='Distribution of Gender for each Age Group for Top 10 Most Used Pho
plt.legend(fontsize=12)

plt.show()
```



Distribution of Gender for Top 10 Most Used Phone Brands

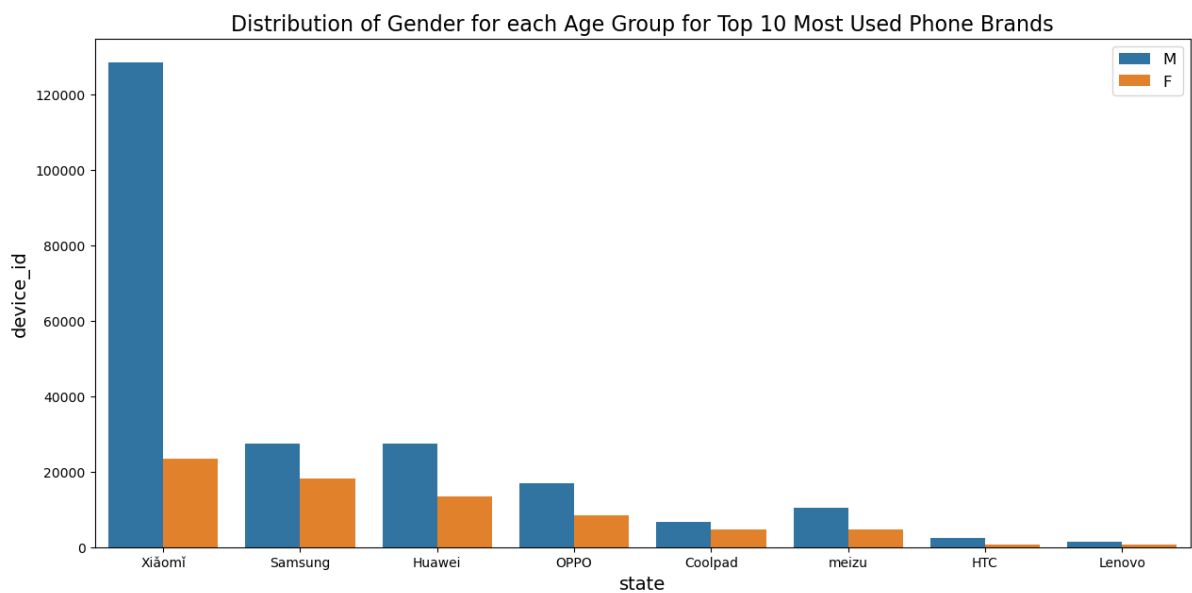
```
In [77]: figure = plt.figure(figsize=[15, 7])
sns.countplot(data=data_top10brand, x="phone_brand", hue="gender")
plt.yticks()

plt.xlabel(xlabel='state', size=14)

plt.ylabel(ylabel='device_id', size=14)

plt.title(label='Distribution of Gender for each Age Group for Top 10 Most Used Pho
plt.legend(fontsize=12)

plt.show()
```



7 Distribution of Age Segments for each State, Gender, and Phone Brand(Consider only 10 Most Used Phone Brands).

Distribution of Age Segments for each State

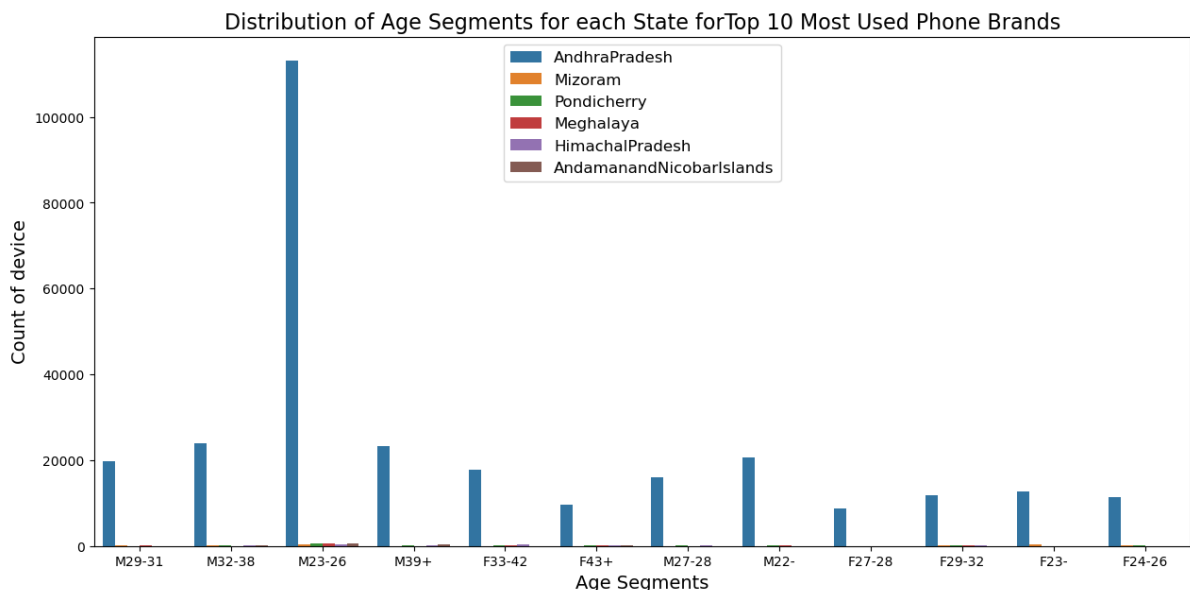
```
In [78]: figure = plt.figure(figsize=[15, 7])
sns.countplot(data=data_top10brand, x="group", hue="state")
plt.yticks()

plt.xlabel(xlabel='Age Segments', size=14)

plt.ylabel(ylabel='Count of device', size=14)

plt.title(label='Distribution of Age Segments for each State forTop 10 Most Used Ph
plt.legend(fontsize=12)

plt.show()
```



Distribution of Age Segments for each gender

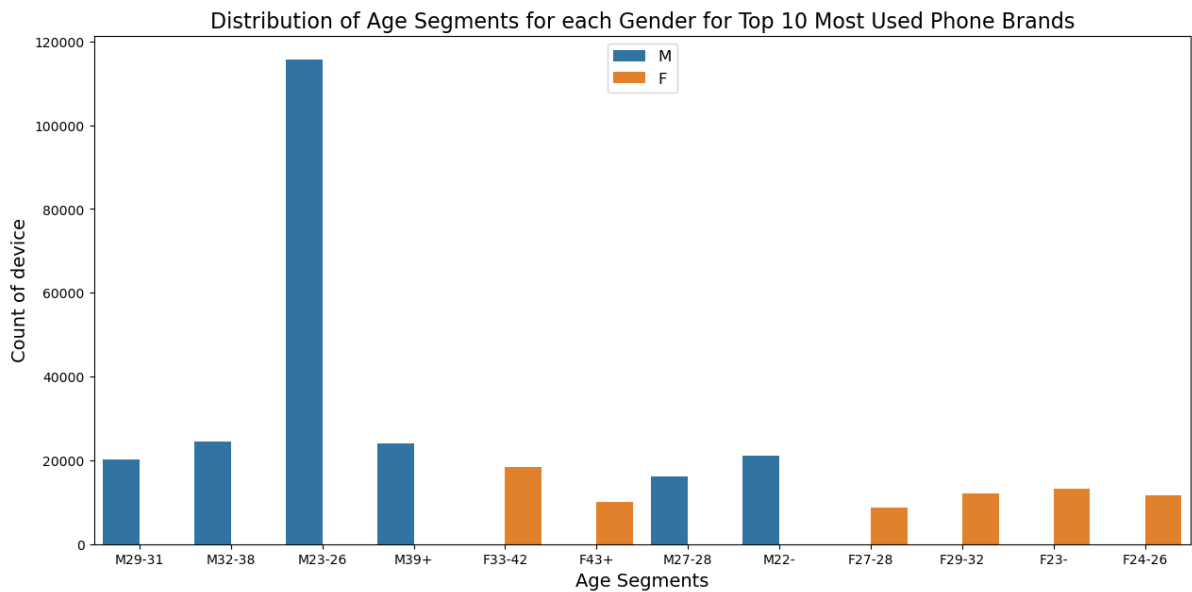
```
In [79]: figure = plt.figure(figsize=[15, 7])
sns.countplot(data=data_top10brand, x="group", hue="gender")
plt.yticks()

plt.xlabel(xlabel='Age Segments', size=14)

plt.ylabel(ylabel='Count of device', size=14)

plt.title(label='Distribution of Age Segments for each Gender for Top 10 Most Used
plt.legend(fontsize=12)

plt.show()
```



Distribution of Age Segments for Brand

```
In [80]: figure = plt.figure(figsize=[15, 7])
sns.countplot(data=data_top10brand, x="group", hue="phone_brand")

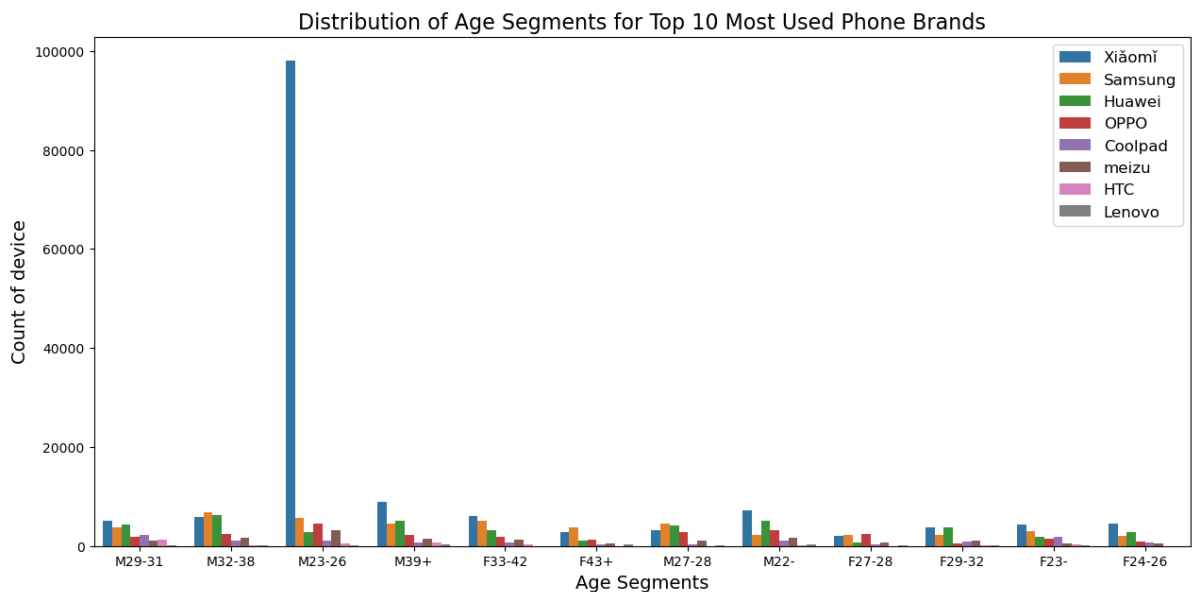
plt.yticks()

plt.xlabel(xlabel='Age Segments', size=14)

plt.ylabel(ylabel='Count of device', size=14)

plt.title(label='Distribution of Age Segments for Top 10 Most Used Phone Brands', s
plt.legend(fontsize=12)

plt.show()
```



8. Hourly distribution of Phone Calls.

converting to date time format

```
In [81]: data['timestamp'] = pd.to_datetime(data['timestamp'])
```

```
In [82]: data['hour'] = data['timestamp'].dt.hour
```

```
In [ ]:
```

```
In [83]: figure = plt.figure(figsize=[15, 5])

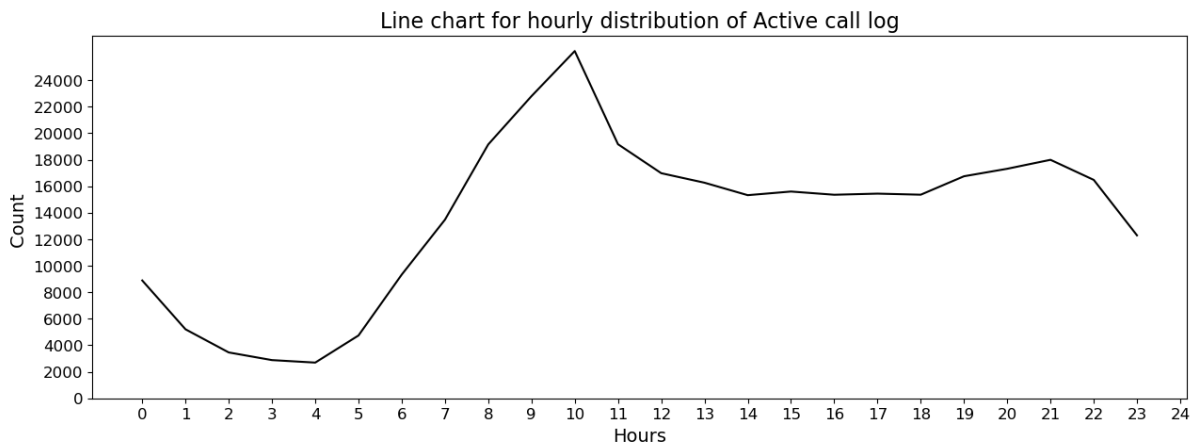
data.groupby(by = 'hour')['device_id'].count().plot.line(color='black')

plt.xticks(np.arange(0,25,1),size=12)
plt.yticks(np.arange(0,26000,2000),size=12)

plt.xlabel(xlabel='Hours', size=14)
plt.ylabel(ylabel='Count ', size=14,rotation=90)

plt.title(label='Line chart for hourly distribution of Active call log', size=16)

plt.show()
```



- users are highly active during the designated call hours, specifically from 8.00am to 9:00 pm

```
In [84]: data['day_of_week'] = data['timestamp'].dt.day_of_week + 1
```

```
In [85]: figure = plt.figure(figsize=[15, 5])

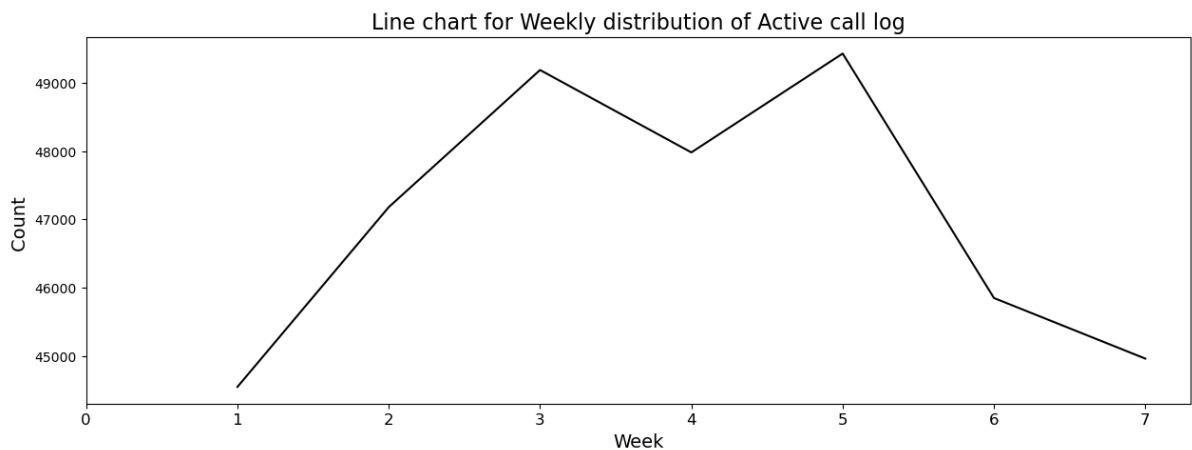
data.groupby(by = 'day_of_week')['device_id'].count().plot.line(color='black')

plt.xticks(np.arange(0,8,1),size=12)

plt.xlabel(xlabel='Week', size=14)
plt.ylabel(ylabel='Count ', size=14,rotation=90)

plt.title(label='Line chart for Weekly distribution of Active call log', size=16)

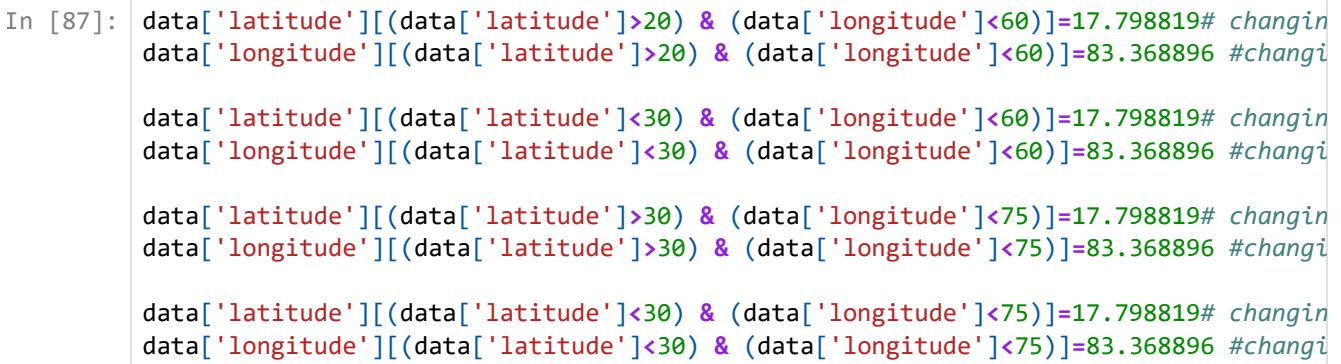
plt.show()
```



- Users tend to be less active on calls on Sundays and Saturdays
- Possible reasons for decreased activity (e.g., rest days, family time)

Handling Improper Lat & Long

```
In [86]: plt.scatter(x=data['longitude'], y=data['latitude'])
plt.show()
```



```
C:\Users\Abhishek\AppData\Local\Temp\ipykernel_17940\1978007308.py:1: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/
user_guide/indexing.html#returning-a-view-versus-a-copy
    data['latitude'][(data['latitude']>20) & (data['longitude']<60)]=17.798819# chang
ing lat latitude']>20 and longitude']<60
C:\Users\Abhishek\AppData\Local\Temp\ipykernel_17940\1978007308.py:2: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/
user_guide/indexing.html#returning-a-view-versus-a-copy
    data['longitude'][(data['latitude']>20) & (data['longitude']<60)]=83.368896 #chan
ging lon latitude']>20 and longitude']<60
C:\Users\Abhishek\AppData\Local\Temp\ipykernel_17940\1978007308.py:4: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/
user_guide/indexing.html#returning-a-view-versus-a-copy
    data['latitude'][(data['latitude']<30) & (data['longitude']<60)]=17.798819# chang
ing lat latitude']<30) & (data['longitude']<60)
C:\Users\Abhishek\AppData\Local\Temp\ipykernel_17940\1978007308.py:5: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/
user_guide/indexing.html#returning-a-view-versus-a-copy
    data['longitude'][(data['latitude']<30) & (data['longitude']<60)]=83.368896 #chan
ging lon latitude']<30) & (data['longitude']<60)
C:\Users\Abhishek\AppData\Local\Temp\ipykernel_17940\1978007308.py:7: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/
user_guide/indexing.html#returning-a-view-versus-a-copy
    data['latitude'][(data['latitude']>30) & (data['longitude']<75)]=17.798819# chang
ing lat latitude']>30) & (data['longitude']<75)
C:\Users\Abhishek\AppData\Local\Temp\ipykernel_17940\1978007308.py:8: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/
user_guide/indexing.html#returning-a-view-versus-a-copy
    data['longitude'][(data['latitude']>30) & (data['longitude']<75)]=83.368896 #chan
ging lon latitude']>30) & (data['longitude']<75)
C:\Users\Abhishek\AppData\Local\Temp\ipykernel_17940\1978007308.py:10: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

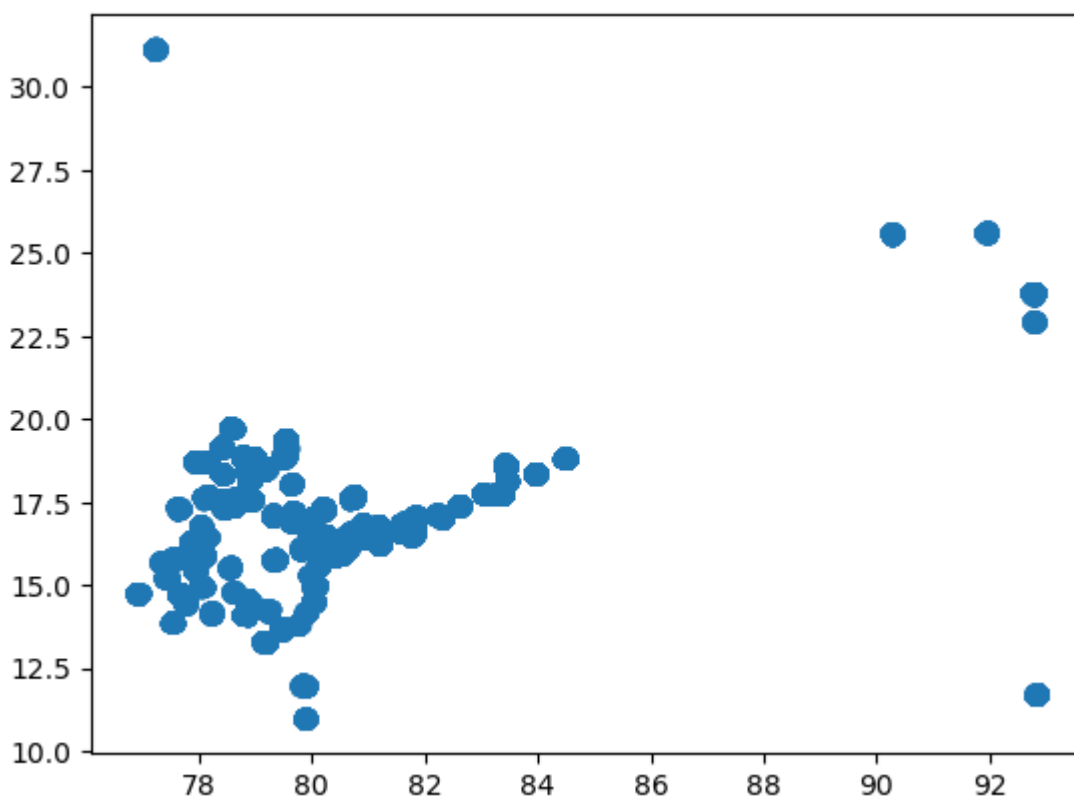
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/
user_guide/indexing.html#returning-a-view-versus-a-copy
    data['latitude'][(data['latitude']<30) & (data['longitude']<75)]=17.798819# chang
ing lat latitude']<30) & (data['longitude']<75)
C:\Users\Abhishek\AppData\Local\Temp\ipykernel_17940\1978007308.py:11: SettingWithC
opyWarning:
```


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
data['longitude'][(data['latitude']<30) & (data['longitude']<75)]=83.368896 #changing lon latitude']<30) & (data['longitude']<75)]
```

```
In [88]: plt.scatter(x=data['longitude'], y=data['latitude'])  
plt.show()
```



```
In [ ]:
```

Plot the Users on the Map.

```
In [89]: pip install folium
```

Requirement already satisfied: folium in c:\users\abhishek\anaconda3\lib\site-packages (0.14.0)
 Requirement already satisfied: numpy in c:\users\abhishek\anaconda3\lib\site-packages (from folium) (1.23.5)
 Requirement already satisfied: branca>=0.6.0 in c:\users\abhishek\anaconda3\lib\site-packages (from folium) (0.6.0)
 Requirement already satisfied: jinja2>=2.9 in c:\users\abhishek\anaconda3\lib\site-packages (from folium) (2.11.3)
 Requirement already satisfied: requests in c:\users\abhishek\anaconda3\lib\site-packages (from folium) (2.28.1)
 Requirement already satisfied: MarkupSafe>=0.23 in c:\users\abhishek\anaconda3\lib\site-packages (from jinja2>=2.9->folium) (2.0.1)
 Requirement already satisfied: certifi>=2017.4.17 in c:\users\abhishek\anaconda3\lib\site-packages (from requests->folium) (2022.9.14)
 Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\abhishek\anaconda3\lib\site-packages (from requests->folium) (2.0.4)
 Requirement already satisfied: idna<4,>=2.5 in c:\users\abhishek\anaconda3\lib\site-packages (from requests->folium) (3.3)
 Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\abhishek\anaconda3\lib\site-packages (from requests->folium) (1.26.11)
 Note: you may need to restart the kernel to use updated packages.

```
In [90]: import folium
from folium.plugins import MarkerCluster

latitude_list = data["latitude"]
longitude_list = data["longitude"]

map = folium.Map(location=[latitude_list[0], longitude_list[0]], zoom_start=12)

marker_cluster = MarkerCluster().add_to(map)
for lat, lon in zip(latitude_list, longitude_list):
    folium.Marker(location=[lat, lon], icon=None).add_to(marker_cluster)
map
```



9. Summarization---

-

In []:

9.1 Conclusion

-

In []:

In []:

In []:

In []:

In []:

In []:

In []:

In []:

In []: