Ensemble Learning | Machine Learning In Python.

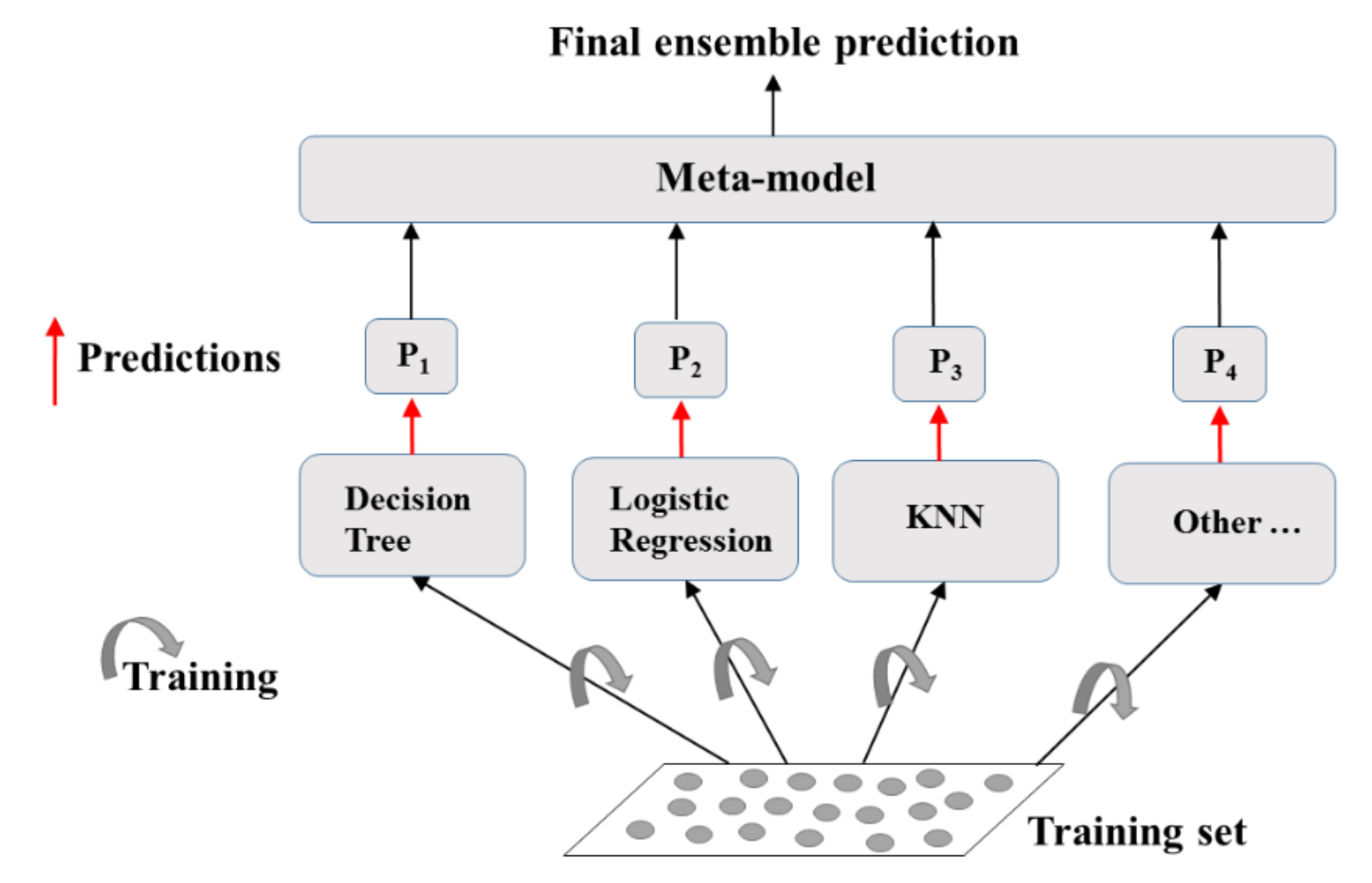
What is Ensemble Learning

- Ensemble Learning uses **multiple machine learning model or mupltiple set of model for the same algorthim which try to make a better prediction.**
- Lets suppose we having a data (particular Datasets) were ensemble model works by traning that dataset on diffrent models and having their each prediction individuly and their each prediction results are combine by using stastical method then finally final prediction is made.
- Morever, we can say that the we having the datasets on which the mutiple algorithms are train on same datasets then finally prediction are made based on the outcome of these individual machine learning algorithms.
- Lets Take an example:-
  - Let suppose that we having the cricket team.
  - In cricket team if see there are many players.Few player for specialization batting and few for wicketkeeping and few for fielding ,few for batting middle order,few for spiner and few for fast baller.
  - Cricket team or any team are made above manner.
  - In the same way,for data every algorithm will be best for the particular set of the data.In this way it work there is the team of the algorithm those are the specialized in the some ways or some field.
  - Then we combine the result of all this algorithm together then it makes the final prediction of data which is at it's best.

In [1]:

```
from IPython.display import Image
Image(filename='C:/Users/Microsoft/Desktop/pandas/1.png')
```

Out[1]:



Why The Ensemble learning is Required ?

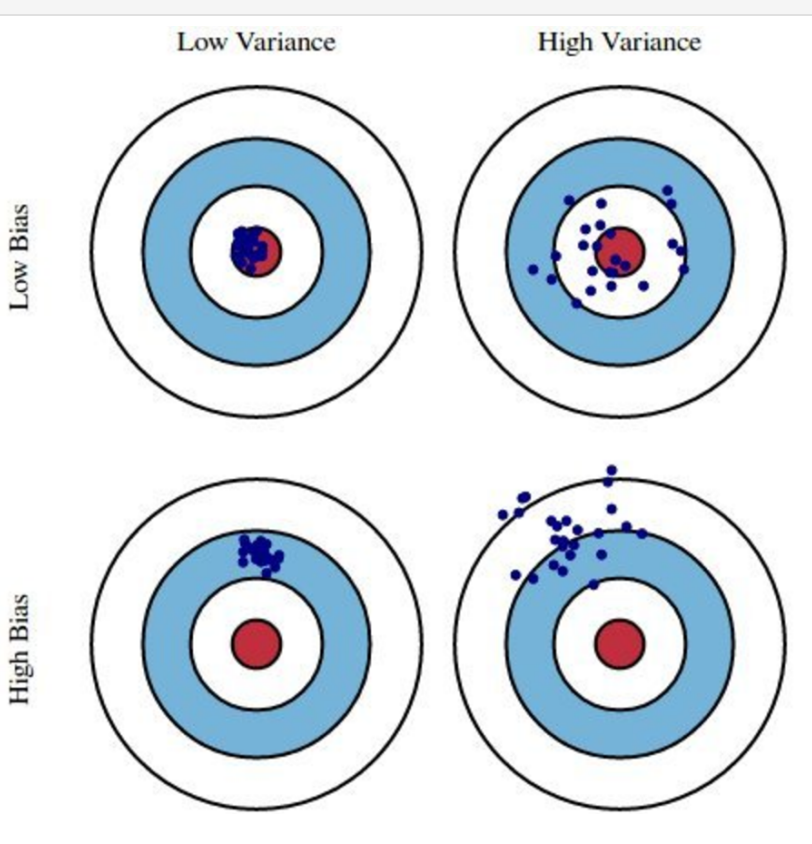
- Once we start with the Ensemble Learning attept to achieve the high accuracy by using single algorithm might endup with the overfitting.
- Every modle has its streight and the weakness.
- Ensemble can be benificial by combining individual model to help hide weakness of individual model.
- lets consider one model might have some weakness but these can cover by using others model those are strong in that weakness.
- If we use the alogorithm in Ensemble Learning with the several kind of model that time it will cover the weakness of individual model and try to give them best to predict the accuarcy of the model.
- What happened if apply the algorithm in our machine learning it create some type of the error.The emersed from any machine models can be broken down into the three components and those components are :-

Bias + Variance + Irreducible error

In [5]:

```
from IPython.display import Image
Image(filename='C:/Users/Microsoft/Desktop/pandas/e..png',width=400,height=500)
```

Out[5]:



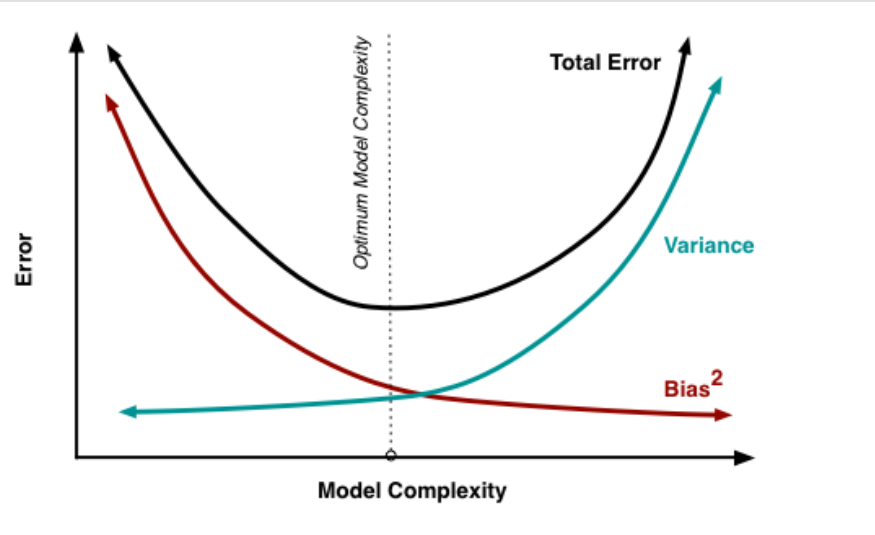
- When we have low bias and Low variance that time all obesrvations or data points are scattered very close and staying themselves around the actual data points.
- When the low bias and high variance then all predicted obesrvations will scattered themselve around the actual data points and but with highly variance inbetween them or we can say that high scattered form.
- When the high bias and low variance then obesrvation will at this time the all predicted obesrvations scattered very closely with low variance but all predicted observations stay away from the atual data. .It will make the heap of the observation(Concentrated manner).
- When the high bias and high variance then obesrvation will at this time the all predicted obesrvations scattered with high variance but stay themselves at the one place which is far away from the actual result or data.

Variance and Bias Trade Off.

In [6]:

```
from IPython.display import Image
Image(filename='C:/Users/Microsoft/Desktop/pandas/unnamed (1).png',width=400,height=500)
```

Out[6]:



- From the diagram if we observed,After the cartain point if start to decreasing the bias then the variance will start increasing and vice versa.
- So,we say here the bias and variance is kind of receprocol to each other.
- So,that means if decreasing the model complexity then biased will be high and if we increase model complexity then bias will be low but variance will be high.
- So,there has to be a sweet spot there we will choose the particular baise and variance so that our model perform best.
- The total error :about the error -It wil start decreasing downwards and then again start incresing the upwards.
- We want such point at that point the model complexity settle down itself with the lowest error and at this point the variance and bias not so high or low that means they must be to stable at which we will get the better and desirable output those who gives us the better performace with low coplexity and low error.
  - variance and bias shold be stable at which there error will be low.
  - Variance is inversly proportional to the bias.
- Bias :-It is useful to represent that how much on an avarage are predicted values are differ from the actual value.

- **High Bias & low Variance :-**
  - It means that we having the **underperforming model** which gives the which on missing essential trends.If we found the high bias and with low variance that time and model complexity will be low .
- **bias is low and variance is high**
  - That means we have a **overfitted model.**
  - But if you have high variance then we need to decrease the model complexity to get the better result.So,this is how fundamental of machine learning work and this is how we can always increase performance of our machine algorithm.

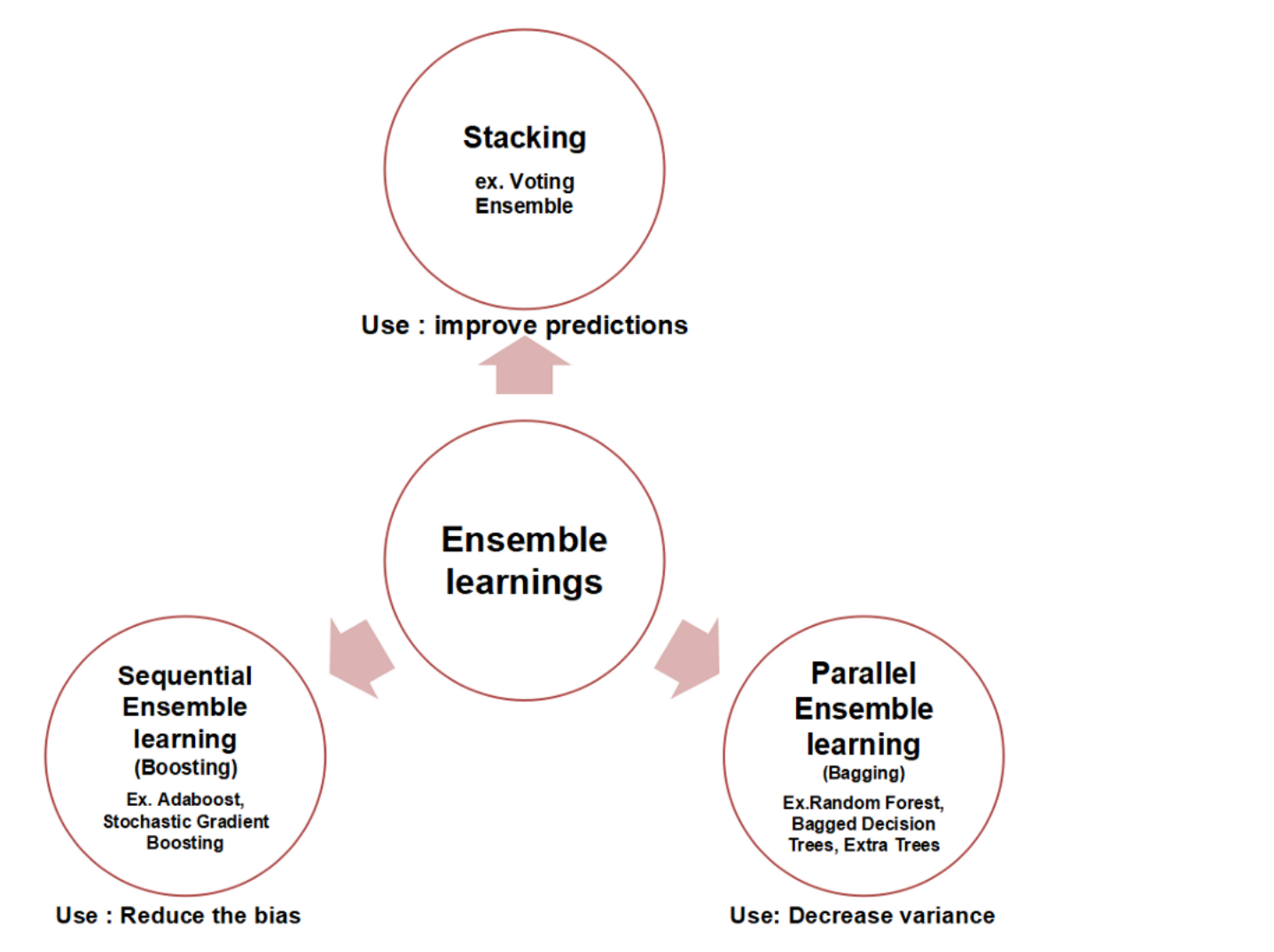
Types Of Ensemble Learning.

- **Basic ensemble learning.**
  - Max voting
  - Avaraging
  - Weighted avearge
- **Advanced ensemble technique.**
  - Stakking
  - Blending
  - Bagging
  - Boosting

In [11]:

```
from IPython.display import Image
Image(filename='C:/Users/Microsoft/Desktop/pandas/1_P0ns6A56MtpGFMQ2g47IYA.png',height=800,width=800)
```

Out[11]:



- **Algorithm based on the bagging.**
  - Bagging meta-estimator
  - Random forest
- **Boosting Algorithms**
  - ADABOOST (adaptive boosting)
  - GBM (gradient boosting)
  - XGB (XG boosting)
  - Light GBM
  - Cat GBM

One by One Explaination About the Ensemble learning.

1.Max-Voting.

- The max voting method is generaly used for classification problem.In this technique,multiple model are used to make prediction for each data point.
- Predictions by each model are considered as Vote and the prediction which will get the from the **majority** models are used as final predictions.
- In this Max voting,if the 3 models are giving output as 1 that is more than one time and the other model gives the 0 only for one time it means that our final prediction will be the 1 not 0, because we had got the max voting for the 1 not for 0.
- Especialy,max voting used in the classification.

2.Averaging.

- Similar to the max voting technique,multiple predictions are made for give data.
- In avaraging,We will take avarage for each prediction here,
- So,the most specificaly averaging is used for the regression problem,were we need to provide the continuous value.
- Since, In the Max Voting technique we can not provide the continuous value but we can provide probablity .
- Avraging can use to get the probablity of being some level.

3.Weighted Average.

- This is an extension of the averaging method.All model are assigned diffrent weights defining the importance of each model for prediction.
- Let's consider that we having the `4 models` and we had assigned the important by using percentage `1st model having the 50%` then `2nd model having 20%` and then `3rd model having the 20%` then `last one having the 10%.`
- So we will do the take result of the 1st model and multiply by 0.5 to the that model then similarly do for the 2nd model and multiply with 0.2 and 3rd model outcome multiply with 0.2 and 4th model outcome will multiply with 0.1.
- Then we make the final prediction this is how weighted avarage work.

Bagging.

- It is also known as Bootstrap algorithm or Bootstarp aggregation or Parellel ensemble learning technique.
- Bagging is a sampling technique in which we create the subset of observation from the origional datasets,with replacement.
- In this technique the sample rows and columns are getting randomly seperated from main datasets for the first decision tree model and for the next time or next decision tree model it again resample the data and provide the sample datasets to the that decision tree model.
- For each decision tree algorithm the bagging algorithm trying to pick some sample rows and columsn from the main data.
- This is called as row and column sampling with replacement.

With Replacement In Bagging.

- It means that there is rows and columns of main dataset will get repeat in each sample for every decision tree model.
- **Randomforest is the perfect example of the bagging.**
- So,the randomforest creates the mutiple subsets like decision tree or we can say multiple decision tree and then randomforest makes the prediction for each decision tree and then finally randomforest gives output on following basis :-
- If Randomforest **classifier** then it will takes **max voting.**
- If it will be **regression** then it will take **avaraging the each of this subset of the trees this is how bagging algorithm work.**
- Bagging is method in that we are running the mutiple decision tree parellely to get the average prediction.
- Bagging Explanation :-https://en.wikipedia.org/wiki/Bootstrap_aggregating

Bagging impact at the training.

In the bagging we use the decision tree and It has two properties that is

- 1.Low bised
- 2.High Variance

In the bagging the we have the multiple decision tree in parellel and each decision tree is creating the low biased and high variance but at the we are going to aggreate the outcome at a place by using the maxvoting for classification and averaging for regression.

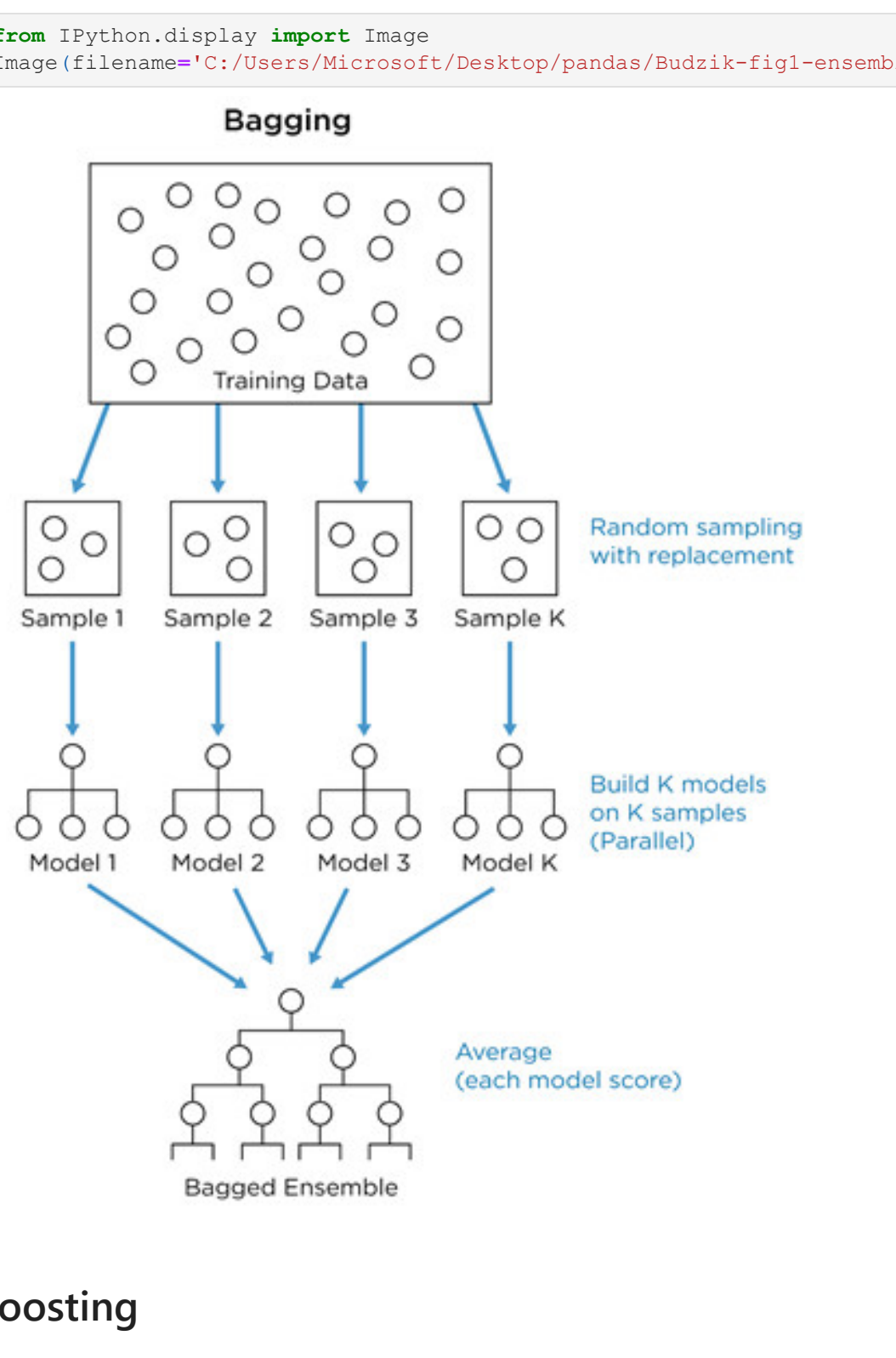
By doing this we are able to achieve the low variance at the end.

This is how we will get the low bias and low varaince at the end ...that's what we want.

In [18]:

```
from IPython.display import Image
Image(filename='C:/Users/Microsoft/Desktop/pandas/Budzik-fig1-ensemble-learning.jpg',height=400,width=500)
```

Out[18]:



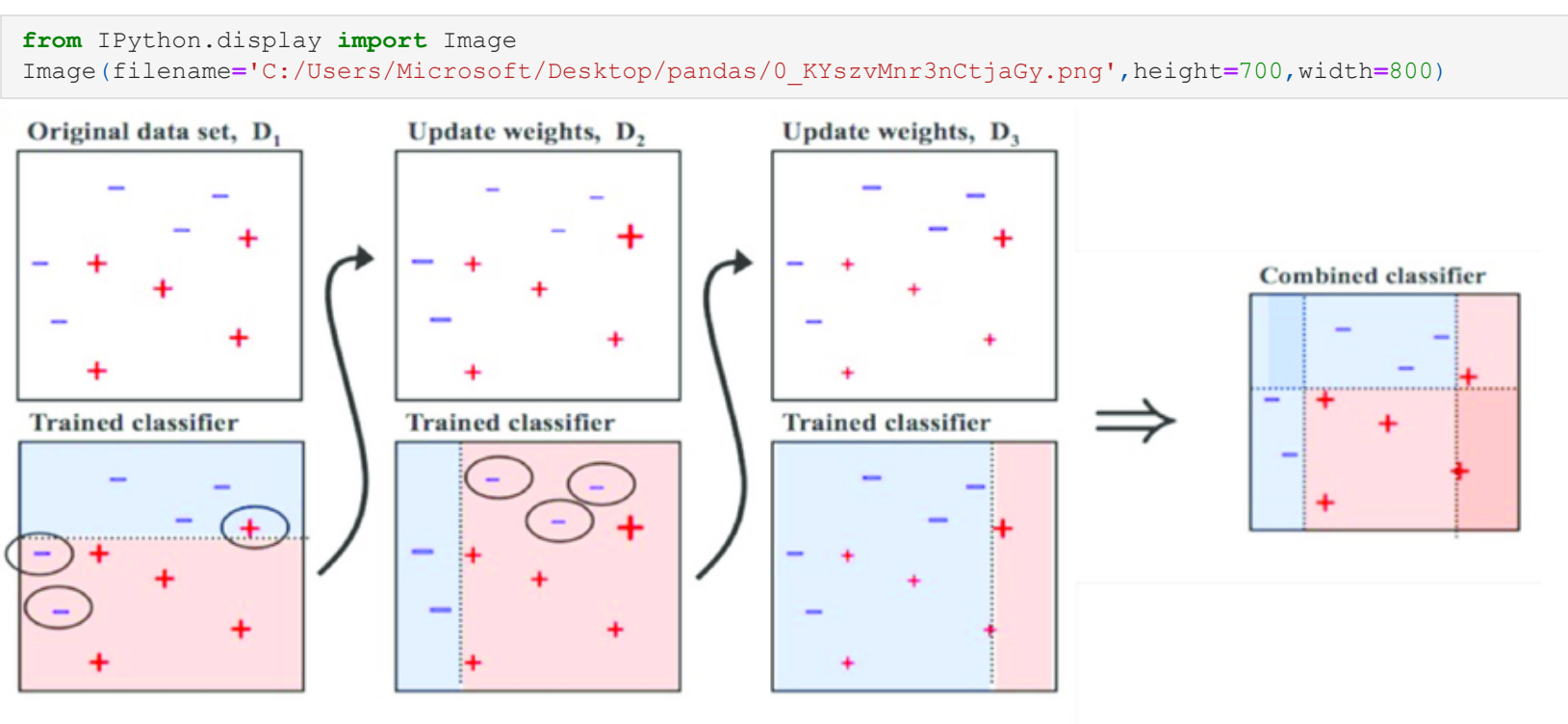
Boosting

- Boosting is squential process, where the each subsequent model attempts to correct the error of the previous model.
- The succeding model are the dependent on the previous model.
- In that ensemble learning we will train the model first time on the avalible data then it will check the wheather it is nicely classified or not if not the there will some error or misclassification.So for that if will train the previous model on the basis of misclassification to improve the model accuracy step by step until and unless we model will get the optimal result or data got purely classified.
- By re-adjusiting or increasing the weight of the weak learner and decraesing the weight of strong learner since in this case our algorithm will get the more preference to the weak learners it happening until the purely and nicely classification will occured.
- However it is making the weak learner to strong learner by adopting the sequential pattern to get the rid of misclassification or errors.
- Boosting explanation:- https://en.wikipedia.org/wiki/Boosting_(machine_learning)

In [22]:

```
from IPython.display import Image
Image(filename='C:/Users/Microsoft/Desktop/pandas/0_KYszvMnr3nCtjaGy.png',height=700,width=800)
```

Out[22]:



Bagging.:-

- Bagging stands for bootstrap aggregation. One way to **reduce the variance** of an estimate is to average together multiple estimates. For example, we can train M different trees on different subsets of the data (chosen randomly with replacement) and compute the ensemble:
- Bagging uses bootstrap sampling to obtain the data subsets for training the base learners. For aggregating the outputs of base learners, bagging uses voting for classification and averaging for regression.

Boosting.:-

- Boosting refers to a family of algorithms that are able to convert weak learners to strong learners. The main principle of boosting is to fit a sequence of weak learners− models that are only slightly better than random guessing.
- The predictions are then combined through a weighted majority vote (classification) or a weighted sum (regression) to produce the final prediction. The principal difference between boosting and the committee methods, such as bagging, is that base learners are trained in sequence on a weighted version of the data.

Stacking.:-

- Stacking is an ensemble learning technique that combines multiple classification or regression models via a meta-classifier or a meta-regressor. The base level models are trained based on a complete training set, then the meta-model is trained on the outputs of the base level model as features.
- The base level often consists of different learning algorithms and therefore stacking ensembles are often heterogeneous.

blog:- https://blog.statsbot.co/ensemble-learning-d1dcd548e936

Algorithm Implement In sklearn

- Bagging
  - A. Random Forest
- Boosting
  - A. XGBoost
  - B. Adaboost
  - C. Gradient Boosting.
  - D.Catboost

- **Random Forest :-** It is another machine learning algorithm that follows the bagging technique.
- **XGBoost(Supreme Gradient Boost) :-** It's is an advance implemenations of the gradient boosting algorithm .It is highly effective ML algorithm although it takes little more time to train and test
- **Adaptive Boosting or AdaBoost :-** It is the one of the sipmplest boosting algorithm.usualy sequence model are created that time it will gone use.
- **Gradient Boosting or GBM :-** is the another machine learning algorithm that works for both classification and Regression.

It is combine boostng algotithm to make weak learner model to strong learner.

Ensemble Learning.

- Ensemble learning is supervised machine learning algorithm it is used to get the better accuracy by minimizing the variance during the period of training the data so that we will rid from overfitting.
- Enemble learning train the same data on the number of algorithm to the prediction and combine the all that result and on the basis of ,maxining votting for classification and avrage vote for the regression type we will decide final prediction.
- The suggesting the ensemble that means it is collecting the all possible the result which comes from the diffrent diffrent algorithm and select the that result on the majority or avarage basis.This is known as ensemble learning.
- **In the ensemble learning we having the many more techiniques that are :-**
  - 1.Sequential Ensemble Learning i.e.Boosting techinique e.g.Adaboost classification.
  - 2.Parellel Ensemble learning techinique i.e. Bagging techinique e.g.RandomForestClassification and Regression.
  - 3.Staking ensemble learning technique.i.e.max Voting techinique.
- Ensemble learning is usefull to maximum accuacy without getting overtting of our model wiith low training time.
- Because it uses the more than one algorithm to predict the better outcome or prediction.Every data is not always good for the evey algorithm to get good accuarcy to do that we need to use some bunch of algorithm to work on it so that we will highest accuarcy.Every model havinhg own advantage and disadvantage to overcome any weakness of the algorithm we use more than one together we will get the better accuarcy.

Bagging.

- Bagging is one of the machine learning techique which is classification or regression.It having pattern of parellel training with the number of decison trees or we can it use the number of decision tresses to the get the possible outcome by combining the all result.
- After combining the all result it will use the aggregation on in such way so that we will get the result as max vote for classification and avrage vote for the regression type of model.
- It can be use for classifiction to the categorical clases or It can use for the regression for the continoue data.
- Bagging is helps to minimize the variance so that buden of the data is seperated to each model seperated in random fashion with replacement.

Boosting.

- Boosting is nothing the Sequential type of ensemble learning technique in that it improving the performance of the classifiction step by step so we will get better accuarcy.
- In that ensemble we will always get the accuarcy better than previous model.
- In the boosting ensemble learning it will first train the data on the initial weight the predict the classification accuarcy if we get the low accuarcy then we update the weight of lower performing attribute i.e. Increses and descres the weight of higher performing attributes. and again train the model on the basis of updated attribute by doing so we we will make weak learning attribue to strong attribute.
- Finally if we chack it the accuarcy stuck somewhere that we can it is an optimal accuarcy that is found by doing the booosting ensemble learning technique.

Algorithm For Machine Learning.

- RandomForest.
- XGBoost.
- Adaboost.
- GradientBoosting.

Let's Start With Machine Learning.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
data =  pd.read_csv('Air_Traffic_Passenger_Statistics.csv')
```

In [3]:
```
data.head()
```

Out[3]:

	Activity Period	Operating Airline	Operating Airline IATA Code	Published Airline	Published Airline IATA Code	GEO Summary	GEO Region	Activity Type Code	Price Category Code	Terminal	Boarding Area	Passenger Count	Adjusted Activity Type Code
0	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Deplaned	Low Fare	Terminal 1	B	27271	Deplaned
1	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Enplaned	Low Fare	Terminal 1	B	29131	Enplaned
2	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Thru / Transit	Low Fare	Terminal 1	B	5415	Thru / Transit * 2
3	200507	Air Canada	AC	Air Canada	AC	International	Canada	Deplaned	Other	Terminal 1	B	35156	Deplaned
4	200507	Air Canada	AC	Air Canada	AC	International	Canada	Enplaned	Other	Terminal 1	B	34090	Enplaned

In [4]:
```
data.tail()
```

Out[4]:

	Activity Period	Operating Airline	Operating Airline IATA Code	Published Airline	Published Airline IATA Code	GEO Summary	GEO Region	Activity Type Code	Price Category Code	Terminal	Boarding Area	Passenger Count	Adj Ac
15002	201603	Virgin America	VX	Virgin America	VX	Domestic	US	Enplaned	Low Fare	Terminal 2	D	194636	Enp
15003	201603	Virgin America	VX	Virgin America	VX	International	Mexico	Deplaned	Low Fare	International	A	4189	Dep
15004	201603	Virgin America	VX	Virgin America	VX	International	Mexico	Enplaned	Low Fare	Terminal 2	D	4693	Enp
15005	201603	Virgin Atlantic	VS	Virgin Atlantic	VS	International	Europe	Deplaned	Other	International	A	12313	Dep
15006	201603	Virgin Atlantic	VS	Virgin Atlantic	VS	International	Europe	Enplaned	Other	International	A	10898	Enp

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15007 entries, 0 to 15006
Data columns (total 16 columns):
Activity Period                15007 non-null int64
Operating Airline              15007 non-null object
Operating Airline IATA Code    14953 non-null object
Published Airline              15007 non-null object
Published Airline IATA Code    14953 non-null object
GEO Summary                    15007 non-null object
GEO Region                     15007 non-null object
Activity Type Code             15007 non-null object
Price Category Code            15007 non-null object
Terminal                       15007 non-null object
Boarding Area                  15007 non-null object
Passenger Count                15007 non-null int64
Adjusted Activity Type Code    15007 non-null object
Adjusted Passenger Count       15007 non-null int64
Year                           15007 non-null int64
Month                          15007 non-null object
dtypes: int64(4), object(12)
memory usage: 1.1+ MB
```

In [6]:
```
data.describe()
```

Out[6]:

	Activity Period	Passenger Count	Adjusted Passenger Count	Year
count	15007.000000	15007.000000	15007.000000	15007.000000
mean	201045.073366	29240.521090	29331.917105	2010.385220
std	313.336196	58319.509284	58284.182219	3.137589
min	200507.000000	1.000000	1.000000	2005.000000
25%	200803.000000	5373.500000	5495.500000	2008.000000
50%	201011.000000	9210.000000	9354.000000	2010.000000
75%	201308.000000	21158.500000	21182.000000	2013.000000
max	201603.000000	659837.000000	659837.000000	2016.000000

In [7]:
```
data.isnull().sum()
```

Out[7]:
```
Activity Period                 0
Operating Airline               0
Operating Airline IATA Code    54
Published Airline               0
Published Airline IATA Code    54
GEO Summary                     0
GEO Region                      0
Activity Type Code              0
Price Category Code             0
Terminal                        0
Boarding Area                   0
Passenger Count                 0
Adjusted Activity Type Code     0
Adjusted Passenger Count        0
Year                            0
Month                           0
dtype: int64
```

In [8]:
```
data.nunique()
```

Out[8]:
```
Activity Period                  129
Operating Airline                 77
Operating Airline IATA Code       73
Published Airline                 68
Published Airline IATA Code       64
GEO Summary                        2
GEO Region                         9
Activity Type Code                 3
Price Category Code                2
Terminal                           5
Boarding Area                      8
Passenger Count                11699
Adjusted Activity Type Code        3
Adjusted Passenger Count       11702
Year                              12
Month                             12
dtype: int64
```

In [9]:
```
data.shape
```

Out[9]:
```
(15007, 16)
```

In [10]:
```
dataframe = pd.DataFrame({'Dtypes':data.dtypes,'Unique':data.nunique(),'Duplicated':data.duplicated().sum(),'Nu
```

In [11]:
```
dataframe
```

Out[11]:

	Dtypes	Unique	Duplicated	Null Vallues
Activity Period	int64	129	0	0
Operating Airline	object	77	0	0
Operating Airline IATA Code	object	73	0	54
Published Airline	object	68	0	0
Published Airline IATA Code	object	64	0	54
GEO Summary	object	2	0	0
GEO Region	object	9	0	0
Activity Type Code	object	3	0	0
Price Category Code	object	2	0	0
Terminal	object	5	0	0
Boarding Area	object	8	0	0
Passenger Count	int64	11699	0	0
Adjusted Activity Type Code	object	3	0	0
Adjusted Passenger Count	int64	11702	0	0
Year	int64	12	0	0
Month	object	12	0	0

In [12]:
```
data['Operating Airline IATA Code'].value_counts().sort_values(ascending=False).head()
```

Out[12]:
```
UA    3046
OO     963
AS     751
DL     386
AC     366
Name: Operating Airline IATA Code, dtype: int64
```

In [13]:
```
data['Operating Airline IATA Code'].isnull().sum()
```

Out[13]:
```
54
```

In [14]:
```
data.dropna(subset=['Operating Airline IATA Code'],inplace=True)
```

In [15]:
```
data.isnull().sum()
```

Out[15]:
```
Activity Period                0
Operating Airline              0
Operating Airline IATA Code    0
Published Airline              0
Published Airline IATA Code    0
GEO Summary                    0
GEO Region                     0
Activity Type Code             0
Price Category Code            0
Terminal                       0
Boarding Area                  0
Passenger Count                0
Adjusted Activity Type Code    0
Adjusted Passenger Count       0
Year                           0
Month                          0
dtype: int64
```

In [16]:
```
data.shape
```

Out[16]:
```
(14953, 16)
```

In [17]:
```
data.head()
```

Out[17]:

	Activity Period	Operating Airline	Operating Airline IATA Code	Published Airline	Published Airline IATA Code	GEO Summary	GEO Region	Activity Type Code	Price Category Code	Terminal	Boarding Area	Passenger Count	Adjusted Activity Type Code
0	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Deplaned	Low Fare	Terminal 1	B	27271	Deplaned
1	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Enplaned	Low Fare	Terminal 1	B	29131	Enplaned
2	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Thru / Transit	Low Fare	Terminal 1	B	5415	Thru / Transit * 2
3	200507	Air Canada	AC	Air Canada	AC	International	Canada	Deplaned	Other	Terminal 1	B	35156	Deplaned
4	200507	Air Canada	AC	Air Canada	AC	International	Canada	Enplaned	Other	Terminal 1	B	34090	Enplaned

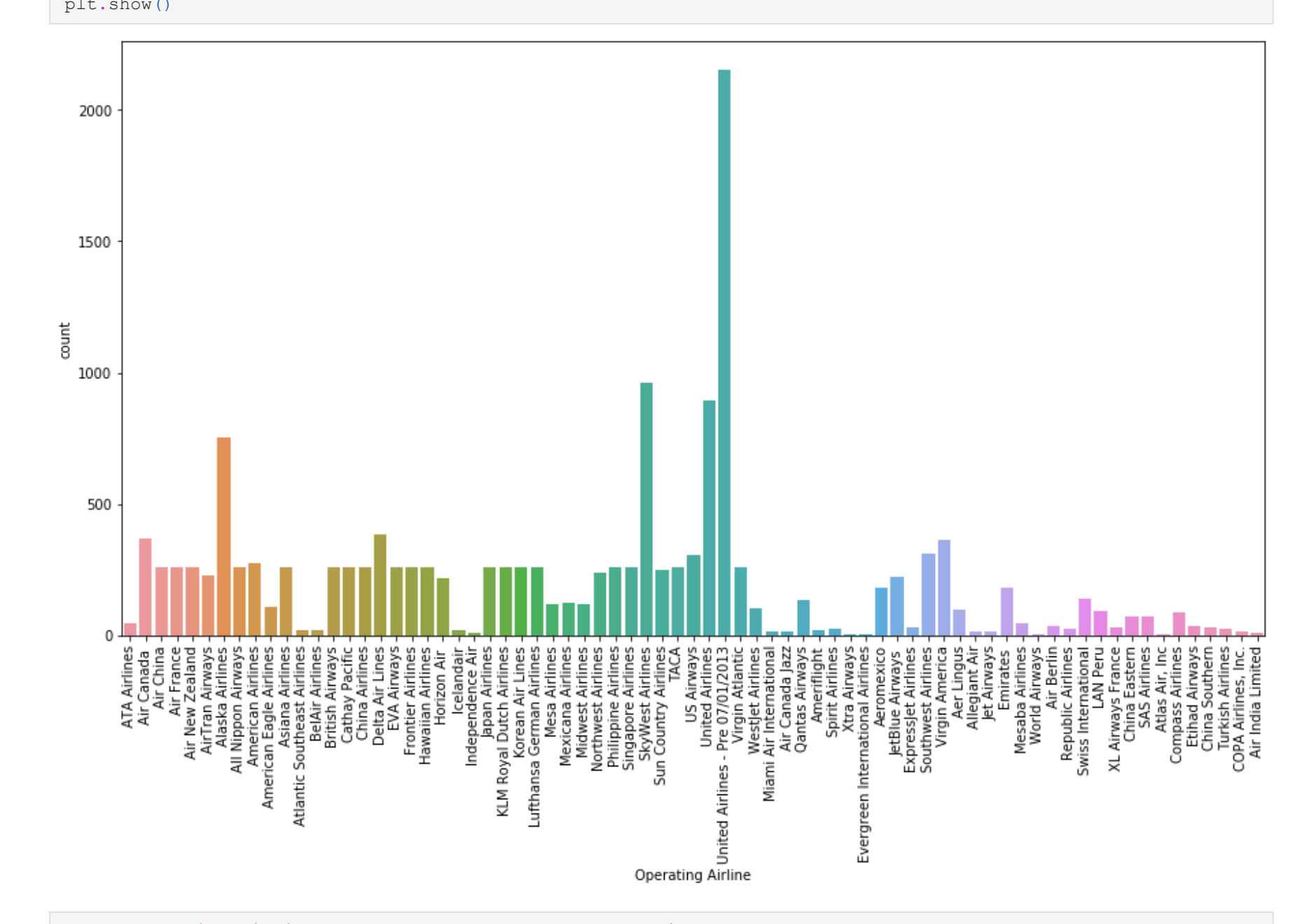
EDA.

In [18]:
```
data['Operating Airline'].value_counts().sort_values(ascending=False).head()
```

Out[18]:
```
United Airlines - Pre 07/01/2013    2154
SkyWest Airlines                     963
United Airlines                      892
Alaska Airlines                      751
Delta Air Lines                      386
Name: Operating Airline, dtype: int64
```

Counts Barplot for Operating Airline,Operating Airline IATA Code,Published Airline,GEO region Count etc..

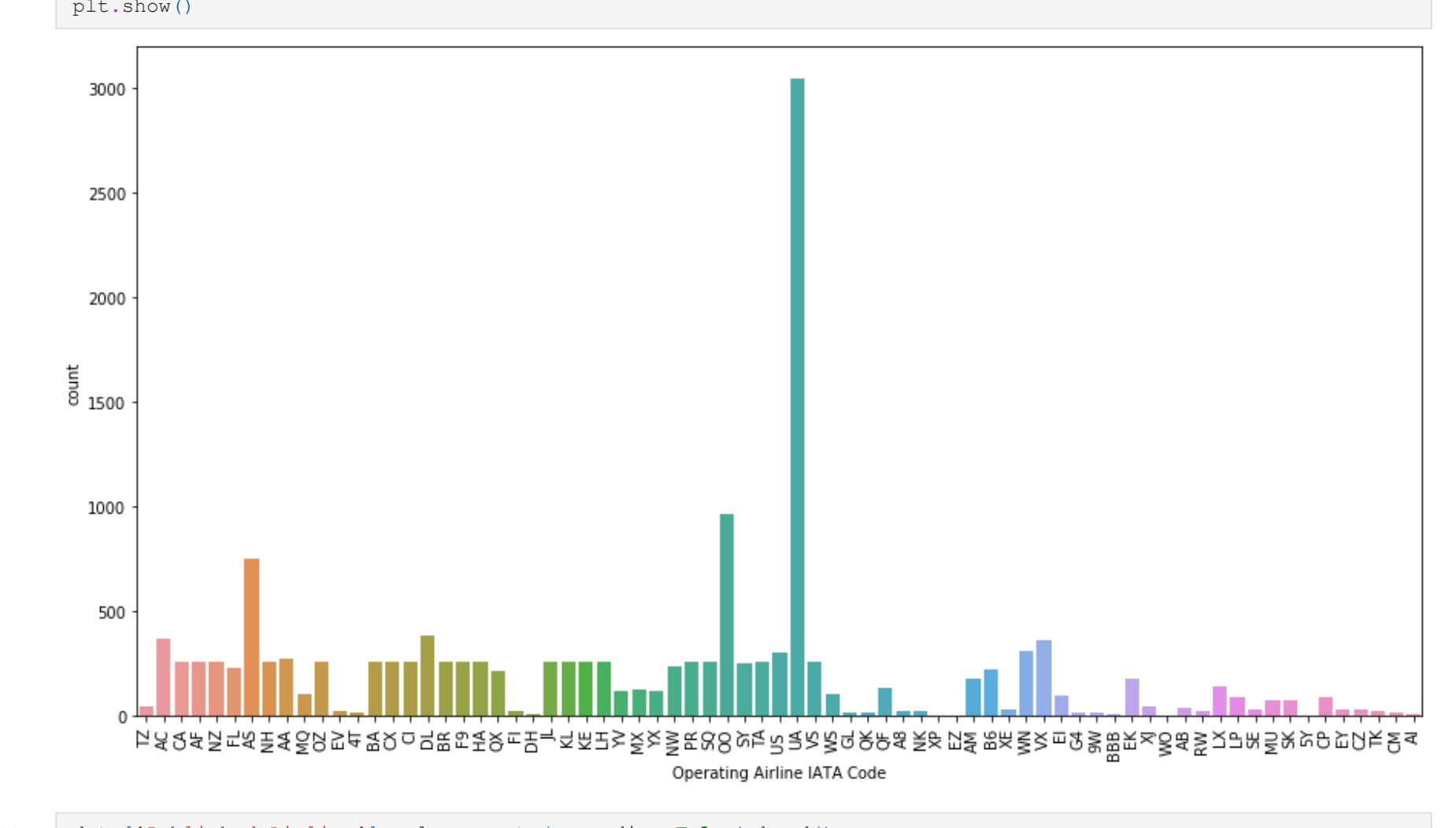
In [19]:
```
plt.figure(figsize=(15,8))
sns.countplot(data['Operating Airline'])
plt.xticks(rotation=90)
plt.show()
```



In [20]:
```
data['Operating Airline IATA Code'].value_counts(ascending=False).head()
```

Out[20]:
```
UA    3046
OO     963
AS     751
DL     386
AC     366
Name: Operating Airline IATA Code, dtype: int64
```

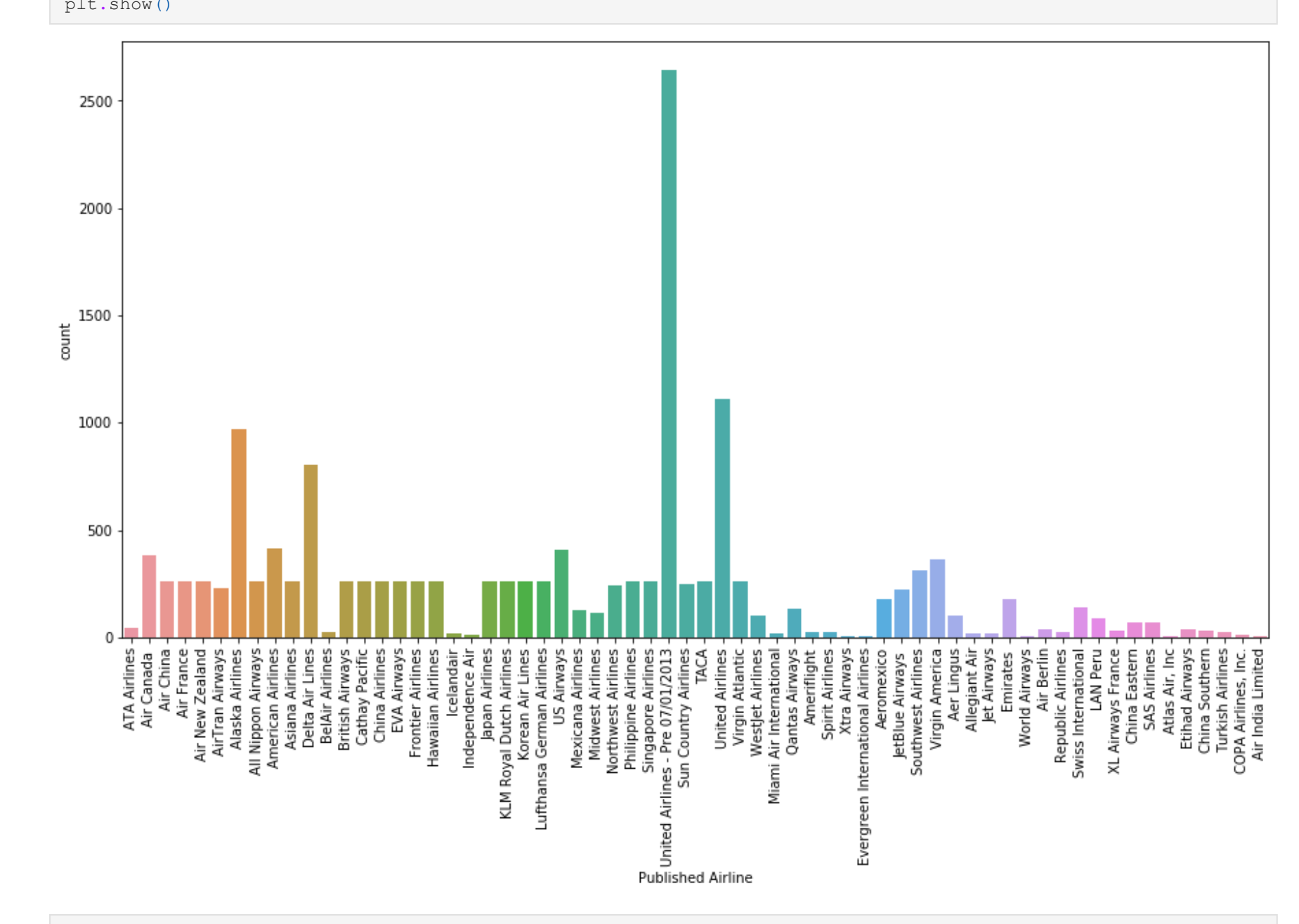
In [21]:
```
plt.figure(figsize=(15,8))
sns.countplot(data['Operating Airline IATA Code'])
plt.xticks(rotation=90)
plt.show()
```



In [22]:
```
data['Published Airline'].value_counts(ascending=False).head()
```

Out[22]:
```
United Airlines - Pre 07/01/2013    2645
United Airlines                     1107
Alaska Airlines                      969
Delta Air Lines                      803
American Airlines                    416
Name: Published Airline, dtype: int64
```

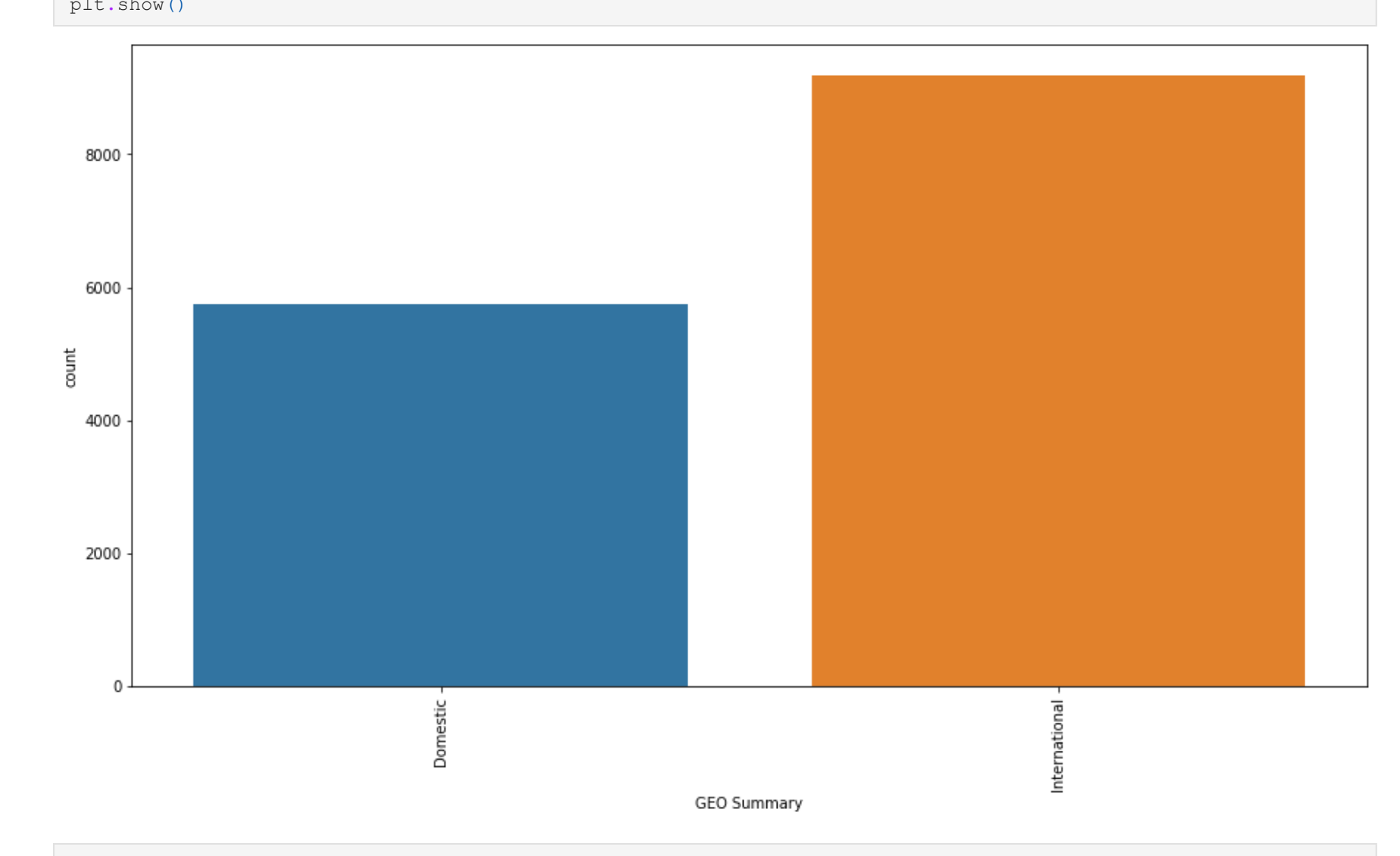
In [23]:
```
plt.figure(figsize=(15,8))
sns.countplot(data['Published Airline'])
plt.xticks(rotation=90)
plt.show()
```



In [24]:
```
data['GEO Summary'].value_counts(ascending=False).head()
```

Out[24]:
```
International    9196
Domestic         5757
Name: GEO Summary, dtype: int64
```

In [25]:
```
plt.figure(figsize=(15,8))
sns.countplot(data['GEO Summary'])
plt.xticks(rotation=90)
plt.show()
```



In [26]:
```
data['GEO Region'].value_counts().sort_values(ascending=False).head()
```

Out[26]:
```
US        5757
Asia      3272
Europe    2078
Canada    1418
Mexico    1115
Name: GEO Region, dtype: int64
```

In [27]:
```
data['Activity Type Code'].value_counts().sort_values(ascending=False)
```

Out[27]:
```
Deplaned          7043
Enplaned          6991
Thru / Transit     919
Name: Activity Type Code, dtype: int64
```

In [28]:
```
data['Price Category Code'].value_counts().sort_values(ascending=False)
```

Out[28]:
```
Other       13069
Low Fare     1884
Name: Price Category Code, dtype: int64
```

In [29]:
```
data['Terminal'].value_counts().sort_values(ascending=False)
```

Out[29]:
```
International    9144
Terminal 1       3241
Terminal 3       2218
Terminal 2        324
Other              26
Name: Terminal, dtype: int64
```

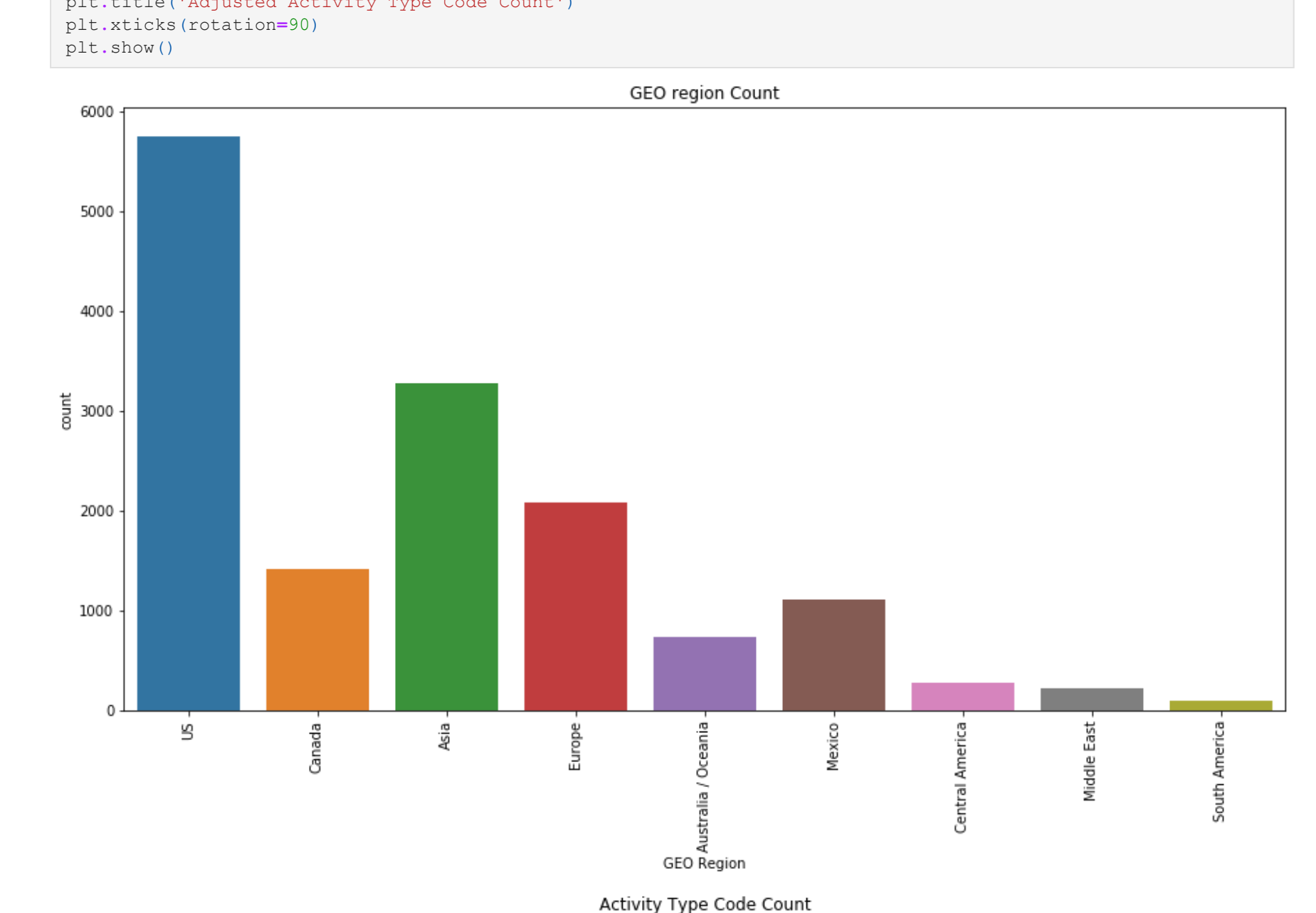
In [30]:
```
data['Boarding Area'].value_counts().sort_values(ascending=False)
```

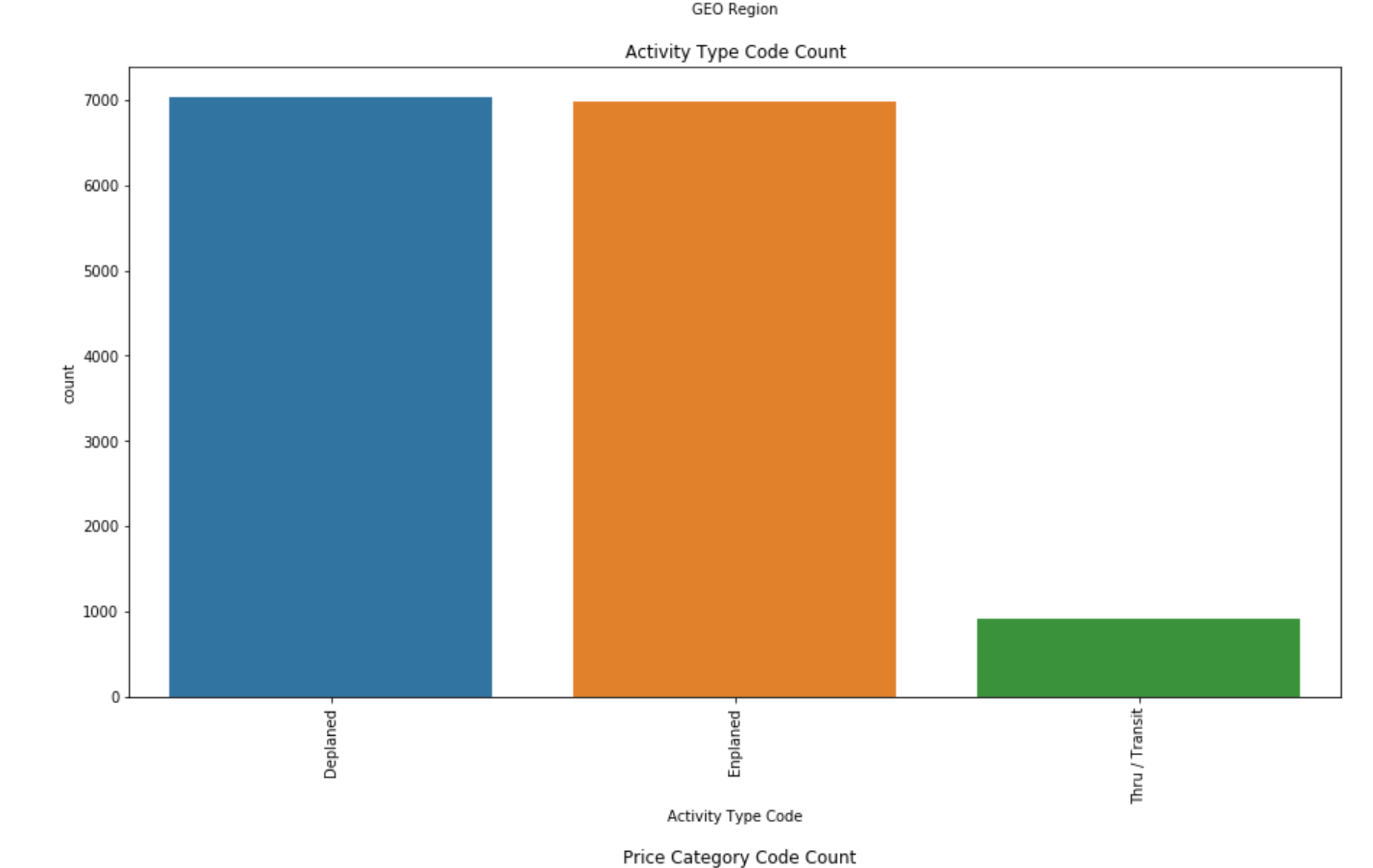
Out[30]:
```
A        5177
G        3987
B        1993
F        1377
C        1228
E         841
D         324
Other      26
Name: Boarding Area, dtype: int64
```

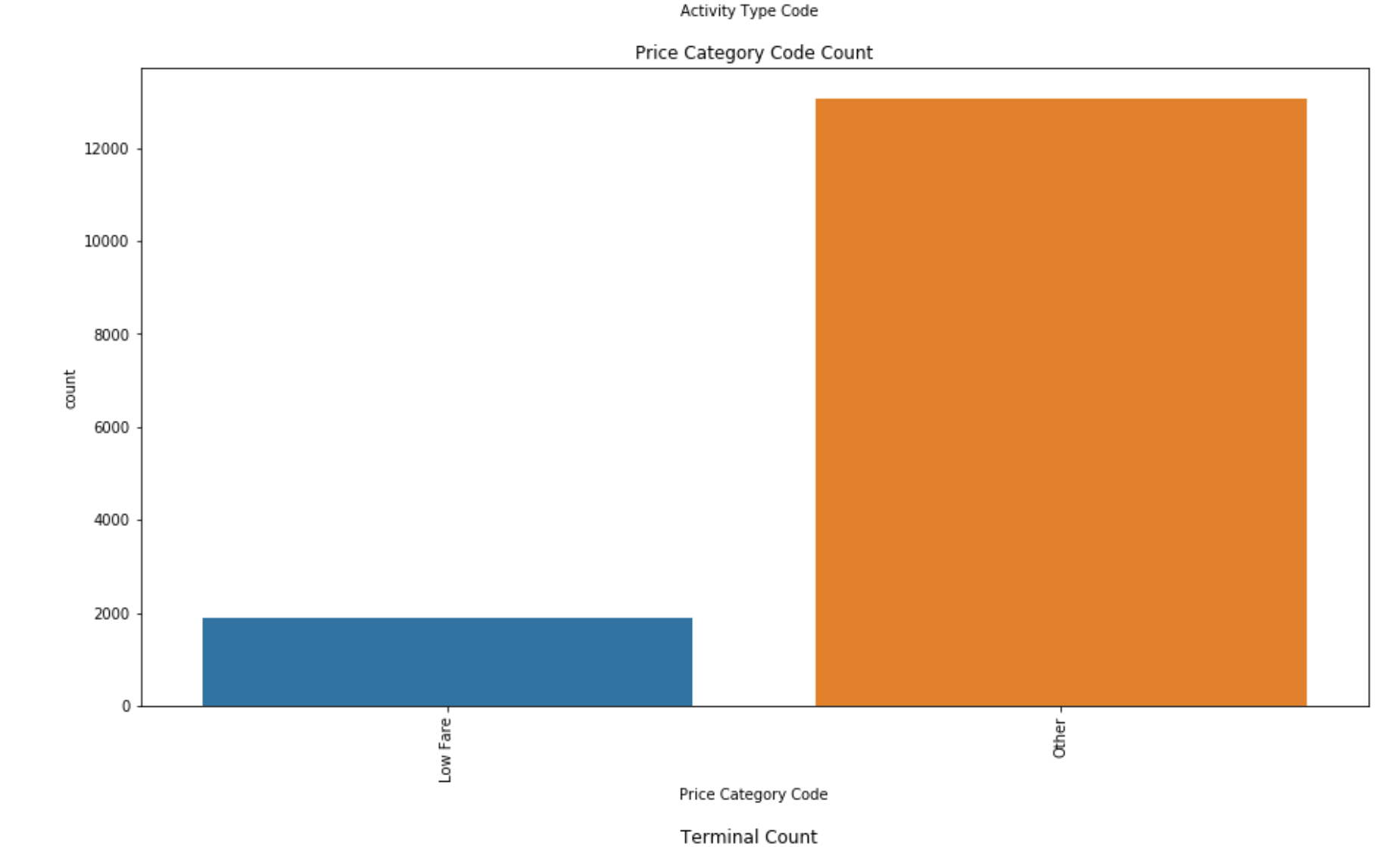
In [31]:
```
data['Activity Type Code'].value_counts().sort_values(ascending=False)
```

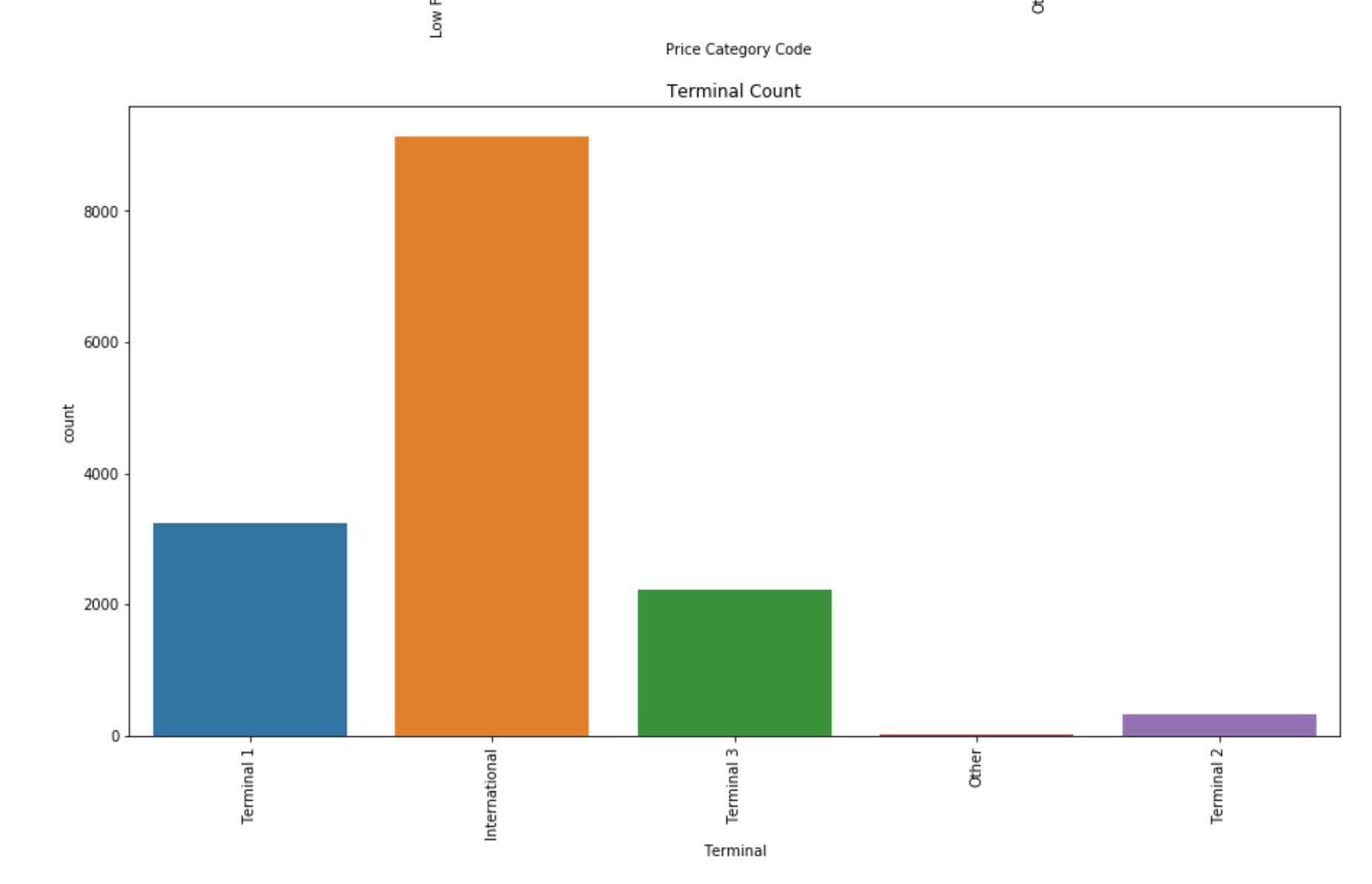
Out[31]:
```
Deplaned          7043
Enplaned          6991
Thru / Transit     919
Name: Activity Type Code, dtype: int64
```

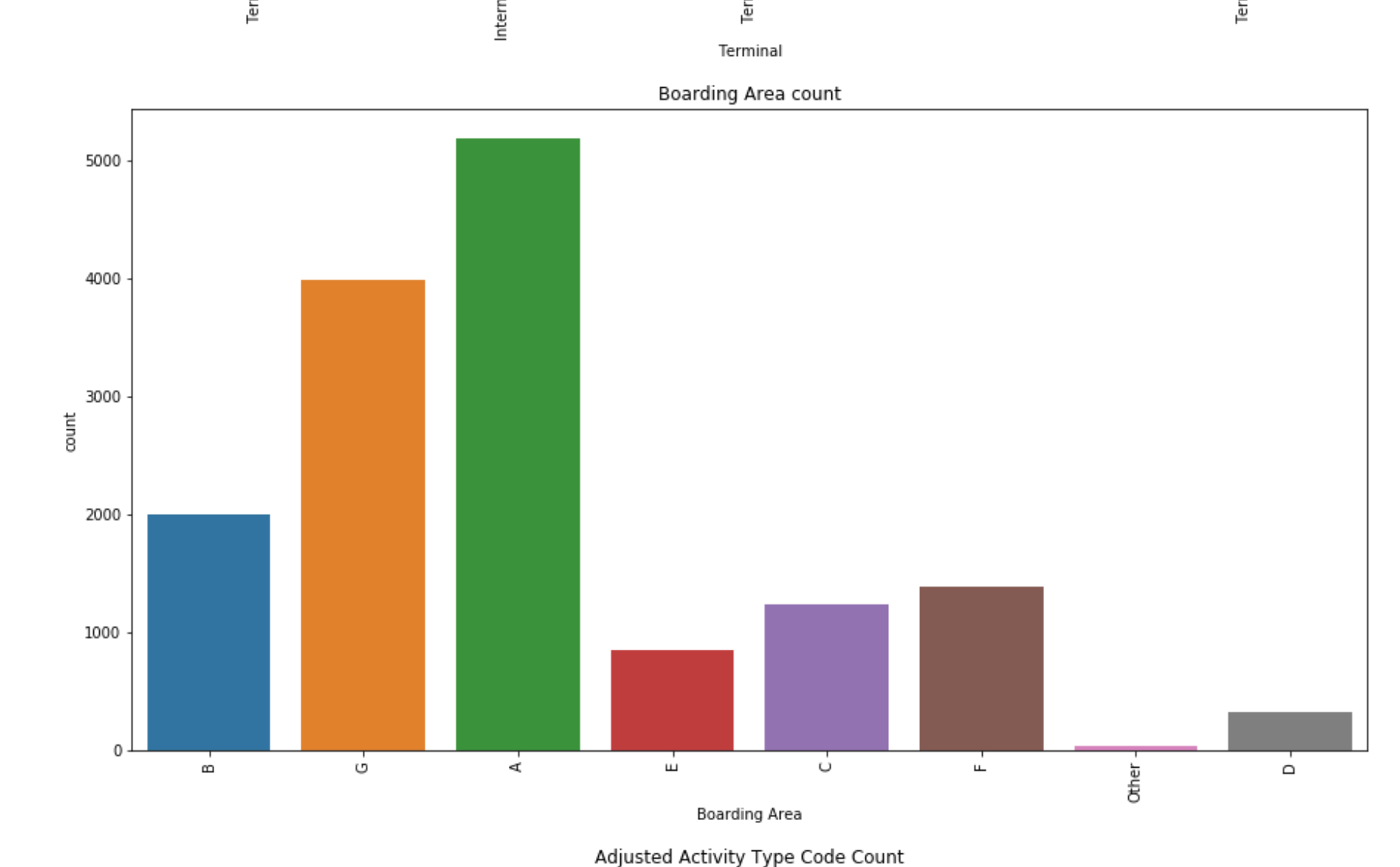
In [32]:
```
data['GEO Region'].value_counts().sort_values(ascending=False).head()
plt.figure(figsize=(15,8))
sns.countplot(data['GEO Region'])
plt.title('GEO region Count')
plt.xticks(rotation=90)
plt.show()

plt.figure(figsize=(15,8))
sns.countplot(data['Activity Type Code'])
plt.title('Activity Type Code Count')
plt.xticks(rotation=90)
plt.show()

plt.figure(figsize=(15,8))
sns.countplot(data['Price Category Code'])
plt.title('Price Category Code Count')
plt.xticks(rotation=90)
plt.show()

plt.figure(figsize=(15,8))
sns.countplot(data['Terminal'])
plt.title('Terminal Count')
plt.xticks(rotation=90)
plt.show()

plt.figure(figsize=(15,8))
sns.countplot(data['Boarding Area'])
plt.title('Boarding Area count')
plt.xticks(rotation=90)
plt.show()

plt.figure(figsize=(15,8))
sns.countplot(data['Adjusted Activity Type Code'])
plt.title('Adjusted Activity Type Code Count')
plt.xticks(rotation=90)
plt.show()
```

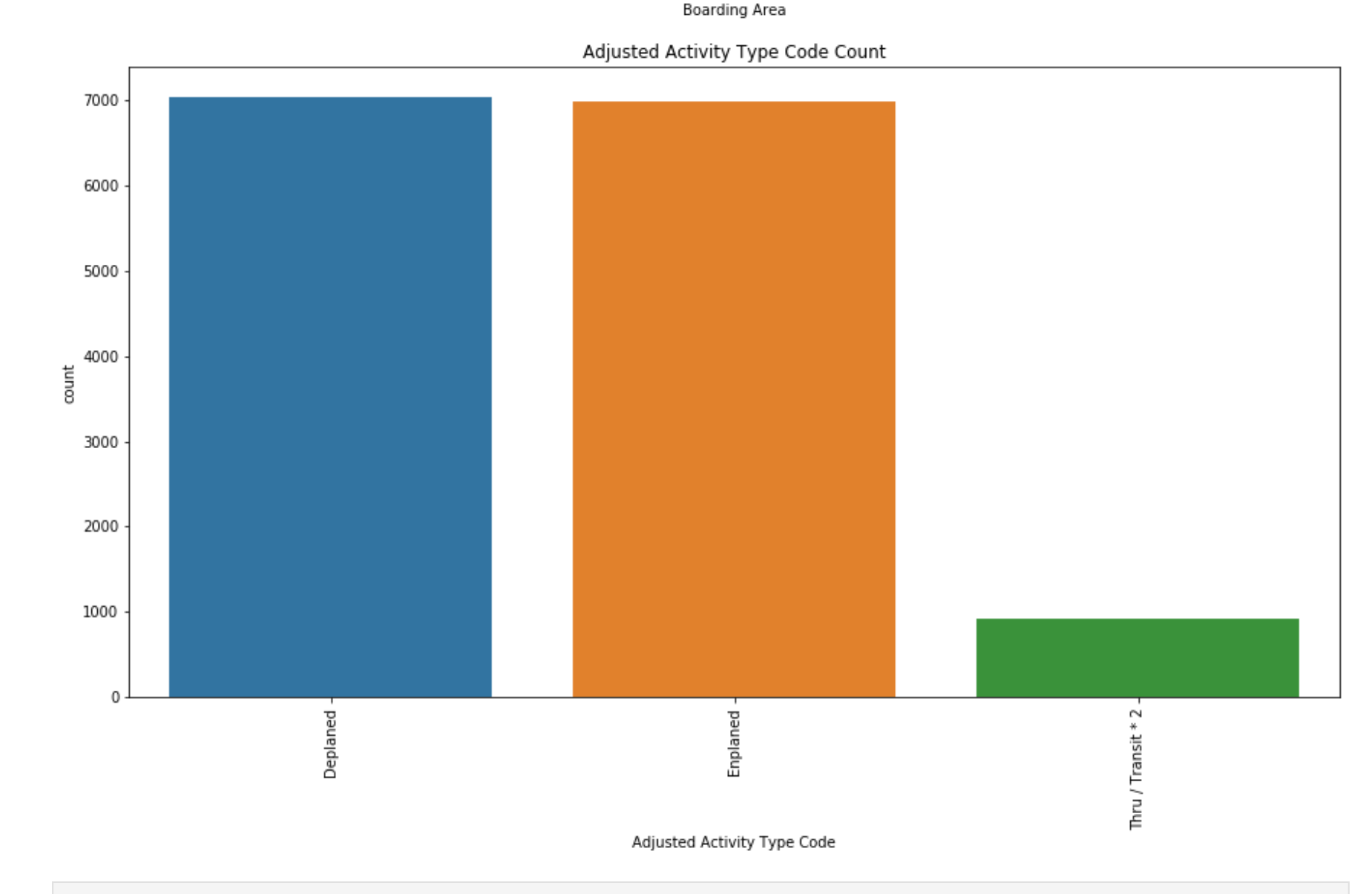












In [33]:
```
data.set_index(['Year','Month'],inplace=True)
```

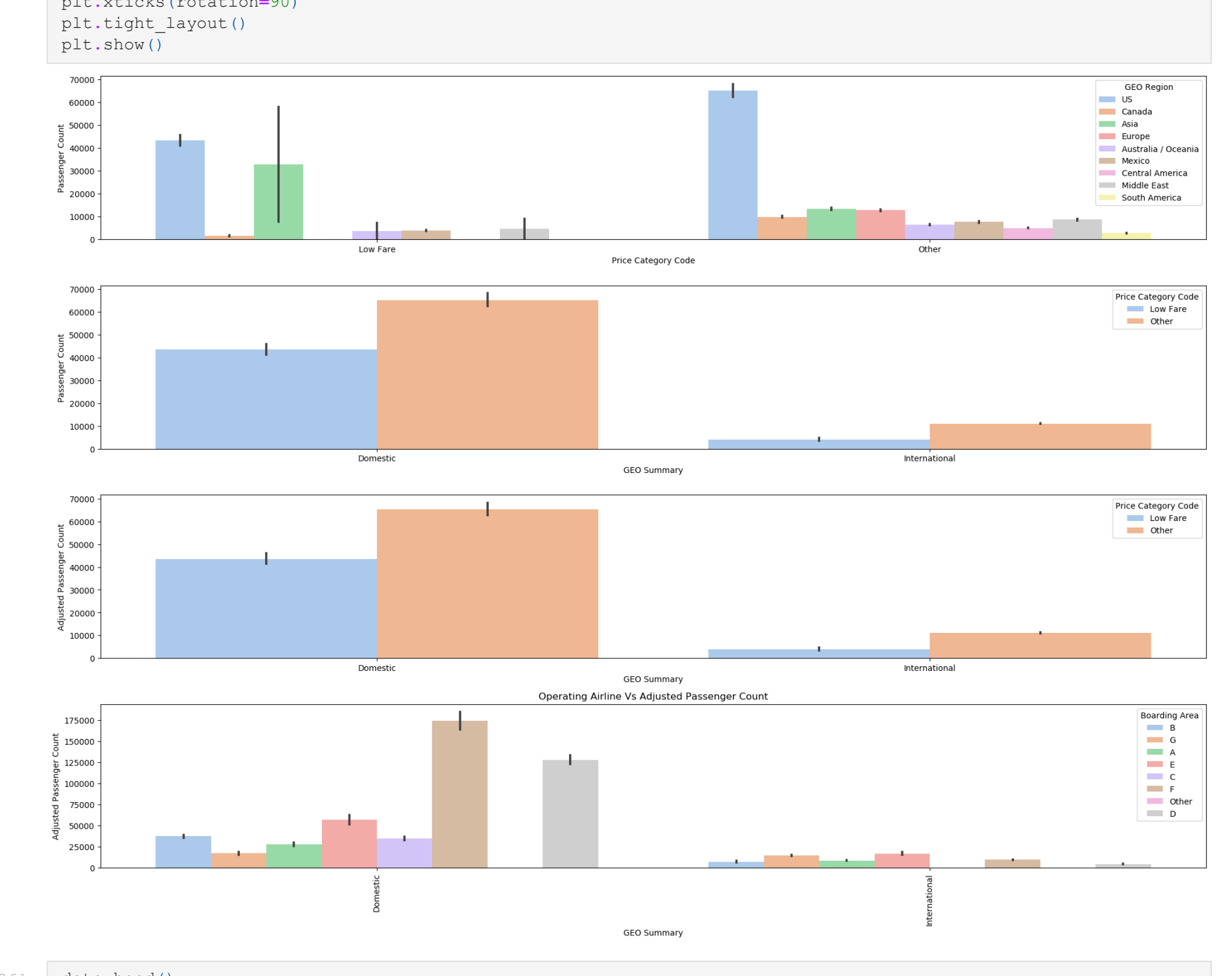
In [34]:
```
data.head()
```

Out[34]:

Year	Month	Activity Period	Operating Airline	Operating Airline IATA Code	Published Airline	Published Airline IATA Code	GEO Summary	GEO Region	Activity Type Code	Price Category Code	Terminal	Boarding Area	Passenger Count
	July	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Deplaned	Low Fare	Terminal 1	B	27271
	July	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Enplaned	Low Fare	Terminal 1	B	29131
2005	July	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Thru / Transit	Low Fare	Terminal 1	B	5415
	July	200507	Air Canada	AC	Air Canada	AC	International	Canada	Deplaned	Other	Terminal 1	B	35156
	July	200507	Air Canada	AC	Air Canada	AC	International	Canada	Enplaned	Other	Terminal 1	B	34090

Relational Barplot.

In [35]:
```
fig = plt.figure(figsize=(20, 15),dpi=100)
ax1 = fig.add_subplot(411)
ax2 = fig.add_subplot(412)
ax3 = fig.add_subplot(413)
ax4 = fig.add_subplot(414)
sns.barplot(data['Price Category Code'],data['Passenger Count'],hue='GEO Region',data=data,ax =ax1,palette ='pa
plt.title('Price Category Code vs Passenger Count')
sns.barplot(data['GEO Summary'],data['Passenger Count'],hue='Price Category Code',data=data,ax=ax2,palette='pas
plt.title('GEO Summary Vs Passenger Count')
plt.xticks(rotation=90)
sns.barplot(data['GEO Summary'],data['Adjusted Passenger Count'],hue='Price Category Code',data=data,ax=ax3,pal
plt.title('Operating Airline Vs Adjusted Passenger Count')
plt.xticks(rotation=90)
sns.barplot(data['GEO Summary'],data['Adjusted Passenger Count'],hue='Boarding Area',data=data,ax=ax4,palette='
plt.title('Operating Airline Vs Adjusted Passenger Count')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



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

Out[36]:

Year	Month	Activity Period	Operating Airline	Operating Airline IATA Code	Published Airline	Published Airline IATA Code	GEO Summary	GEO Region	Activity Type Code	Price Category Code	Terminal	Boarding Area	Passenger Count
	July	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Deplaned	Low Fare	Terminal 1	B	27271
	July	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Enplaned	Low Fare	Terminal 1	B	29131
2005	July	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Thru / Transit	Low Fare	Terminal 1	B	5415
	July	200507	Air Canada	AC	Air Canada	AC	International	Canada	Deplaned	Other	Terminal 1	B	35156
	July	200507	Air Canada	AC	Air Canada	AC	International	Canada	Enplaned	Other	Terminal 1	B	34090

In [37]:

```
data.groupby(data['Boarding Area']).sum().max()
```

Out[37]:

```
Activity Period              1040957202
Passenger Count               138526673
Adjusted Passenger Count      139195536
dtype: int64
```

MAXIMUM_SUM, Count in respect of Boarding areas.

In [38]:

```
print('Sum of max passender avaible on the Boarding area A')
print(data.groupby(data['Boarding Area']=='A').sum().max())
print()
print('Sum of max passender avaible on the Boarding area B')
print(data.groupby(data['Boarding Area']=='B').sum().max())
print()
print('Sum of max passender avaible on the Boarding area C')
print(data.groupby(data['Boarding Area']=='C').sum().max())
print()
print('Sum of max passender avaible on the Boarding area D')
print(data.groupby(data['Boarding Area']=='D').sum().max())
print()
print('Sum of max passender avaible on the Boarding area E')
print(data.groupby(data['Boarding Area']=='E').sum().max())
print()
print('Sum of max passender avaible on the Boarding area F')
print(data.groupby(data['Boarding Area']=='F').sum().max())
print()
print('Sum of max passender avaible on the Boarding area G')
print(data.groupby(data['Boarding Area']=='G').sum().max())
```

```
Sum of max passender avaible on the Boarding area A
Activity Period              1965262462
Passenger Count               380732043
Adjusted Passenger Count      381973548
dtype: int64

Sum of max passender avaible on the Boarding area B
Activity Period              2605673358
Passenger Count               371431933
Adjusted Passenger Count      372643209
dtype: int64

Sum of max passender avaible on the Boarding area C
Activity Period              2759400752
Passenger Count               396533401
Adjusted Passenger Count      397878085
dtype: int64

Sum of max passender avaible on the Boarding area D
Activity Period              2940983458
Passenger Count               404744801
Adjusted Passenger Count      406116288
dtype: int64

Sum of max passender avaible on the Boarding area E
Activity Period              2837248334
Passenger Count               397918132
Adjusted Passenger Count      399259312
dtype: int64

Sum of max passender avaible on the Boarding area F
Activity Period              2729393454
Passenger Count               300278368
Adjusted Passenger Count      300980992
dtype: int64

Sum of max passender avaible on the Boarding area G
Activity Period              2204575253
Passenger Count               381191750
Adjusted Passenger Count      382207927
dtype: int64
```

In [39]:

```
data.groupby(data['Boarding Area']).sum().max()
```

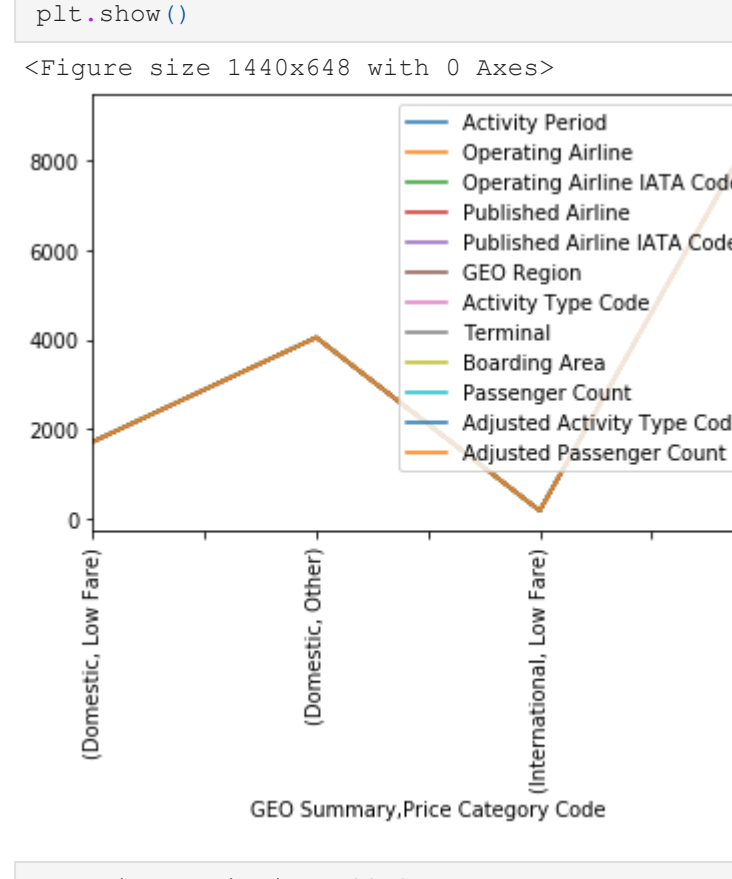
Out[39]:

```
Activity Period              1040957202
Passenger Count               138526673
Adjusted Passenger Count      139195536
dtype: int64
```

In [40]:

```
plt.figure(figsize=(20,9))
data.groupby(['GEO Summary','Price Category Code']).count().plot()
plt.xticks(rotation=90)
plt.show()
```

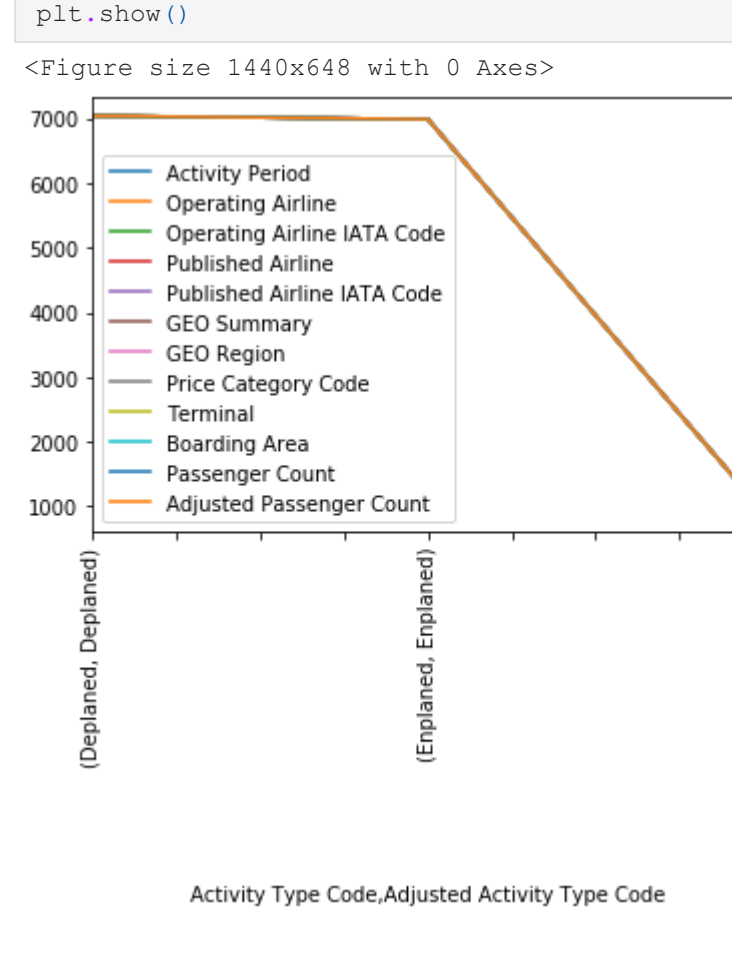
<Figure size 1440x648 with 0 Axes>



In [41]:

```
plt.figure(figsize=(20,9))
data.groupby(['Activity Type Code','Adjusted Activity Type Code']).count().plot()
plt.xticks(rotation=90)
plt.show()
```

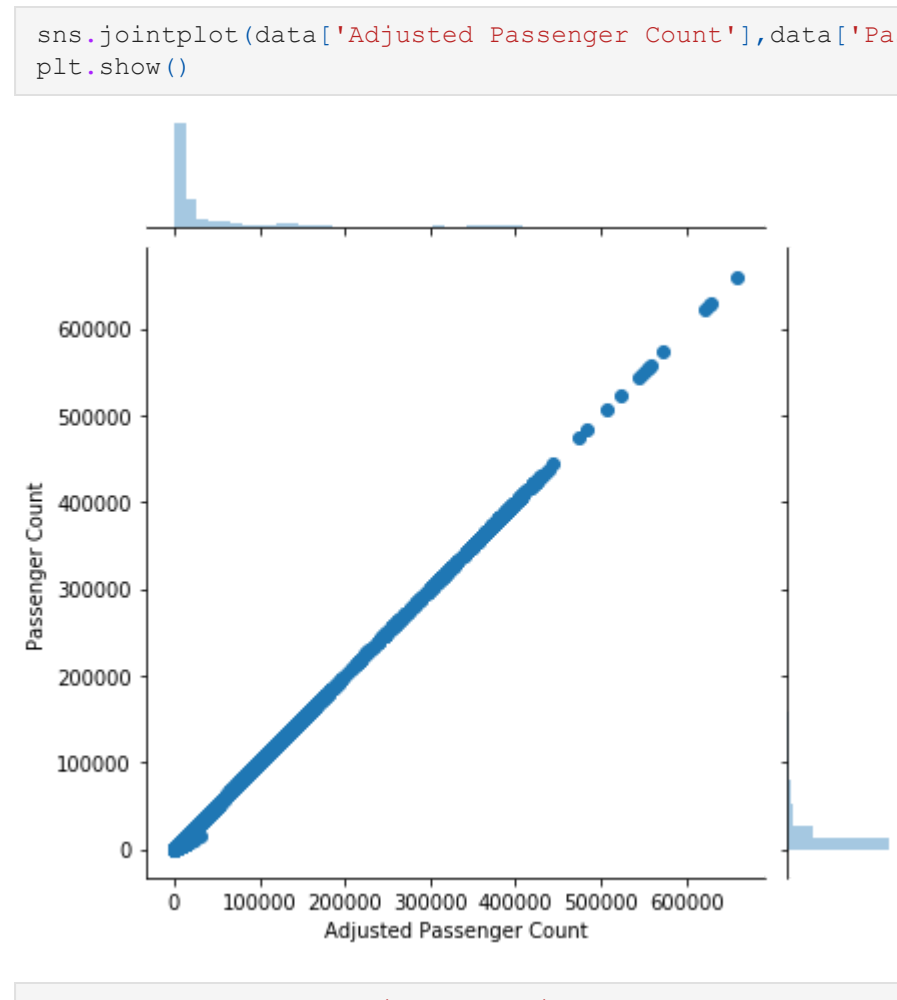
<Figure size 1440x648 with 0 Axes>



Linear Relationship graph

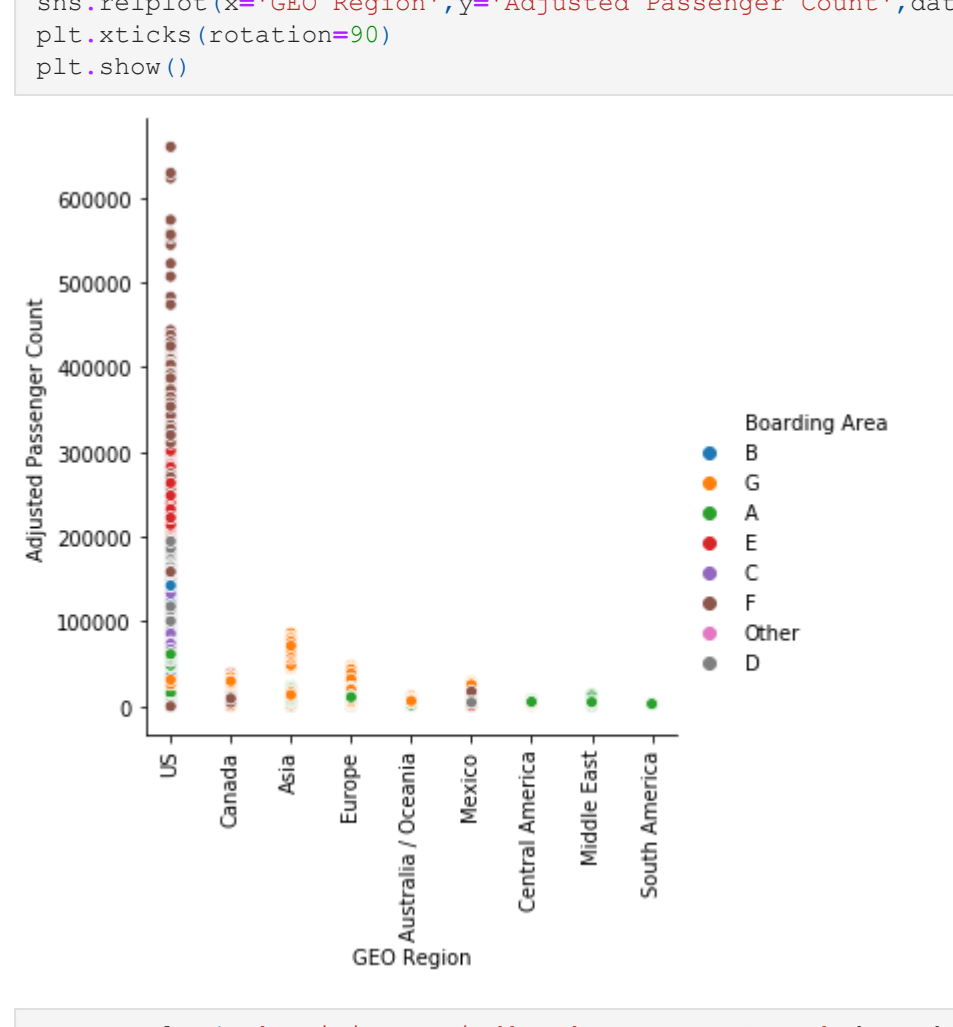
In [42]:

```
sns.jointplot(data['Adjusted Passenger Count'],data['Passenger Count'])
plt.show()
```



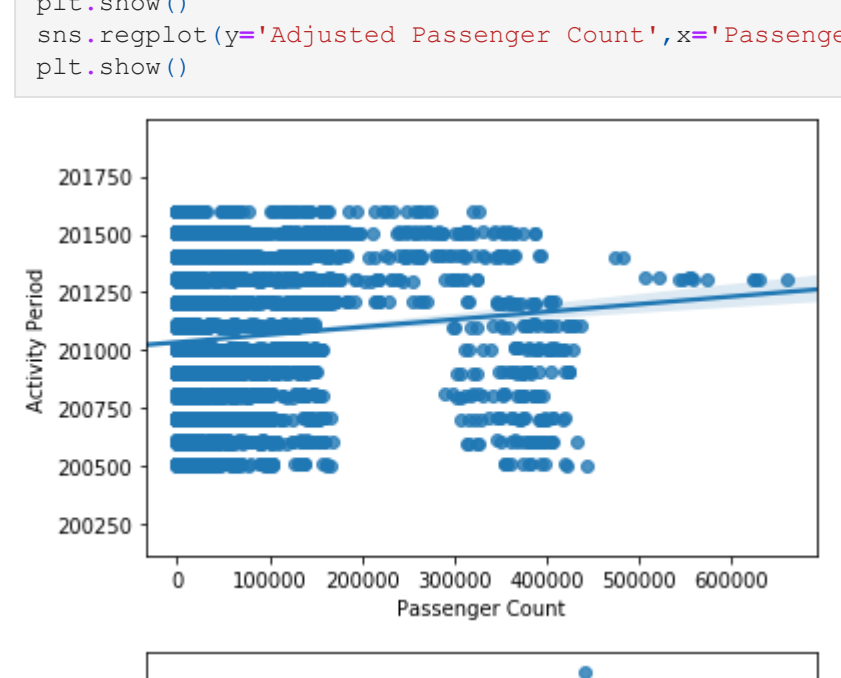
In [43]:

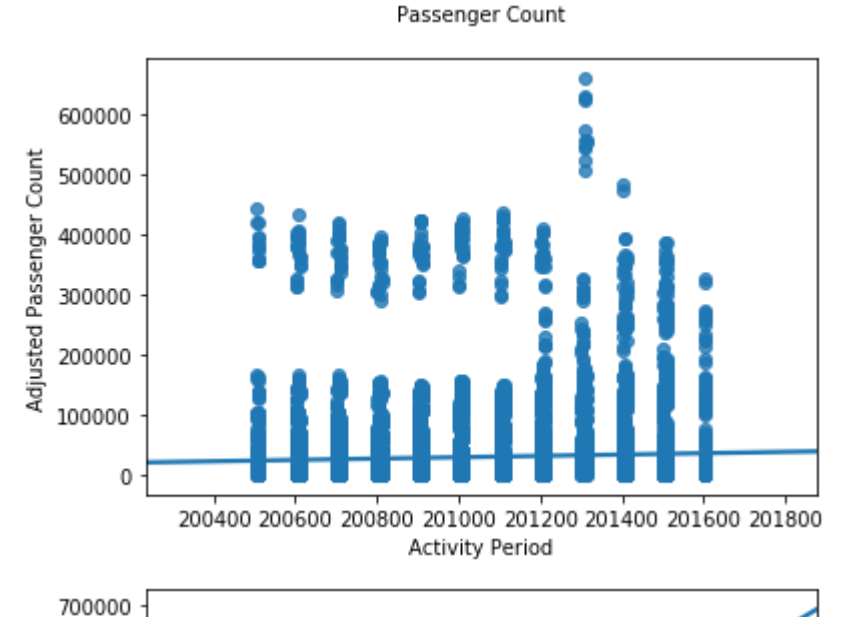
```
sns.relplot(x='GEO Region',y='Adjusted Passenger Count',data=data,hue='Boarding Area')
plt.xticks(rotation=90)
plt.show()
```

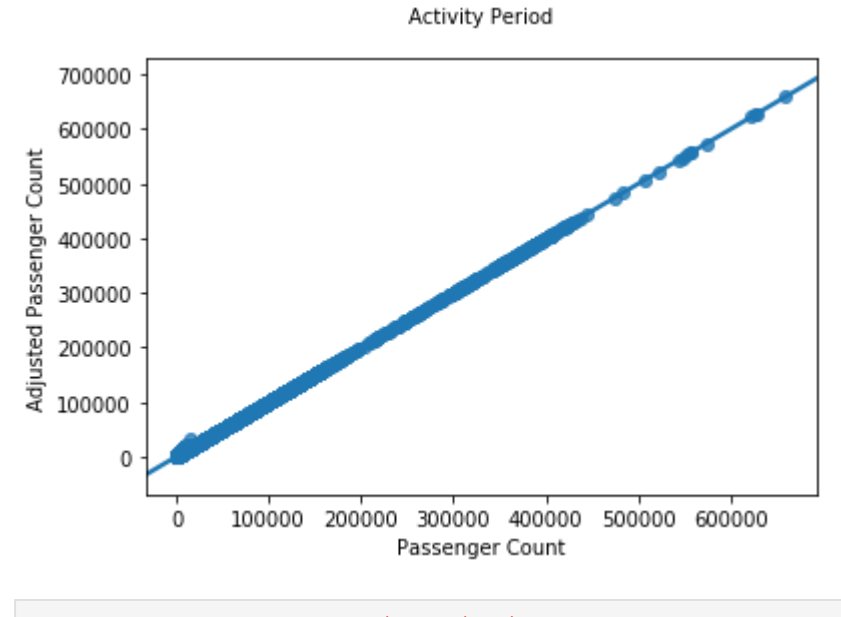


In [44]:

```
sns.regplot(y='Activity Period',x='Passenger Count',data=data)
plt.show()
sns.regplot(y='Adjusted Passenger Count',x='Activity Period',data=data)
plt.show()
sns.regplot(y='Adjusted Passenger Count',x='Passenger Count',data=data)
plt.show()
```

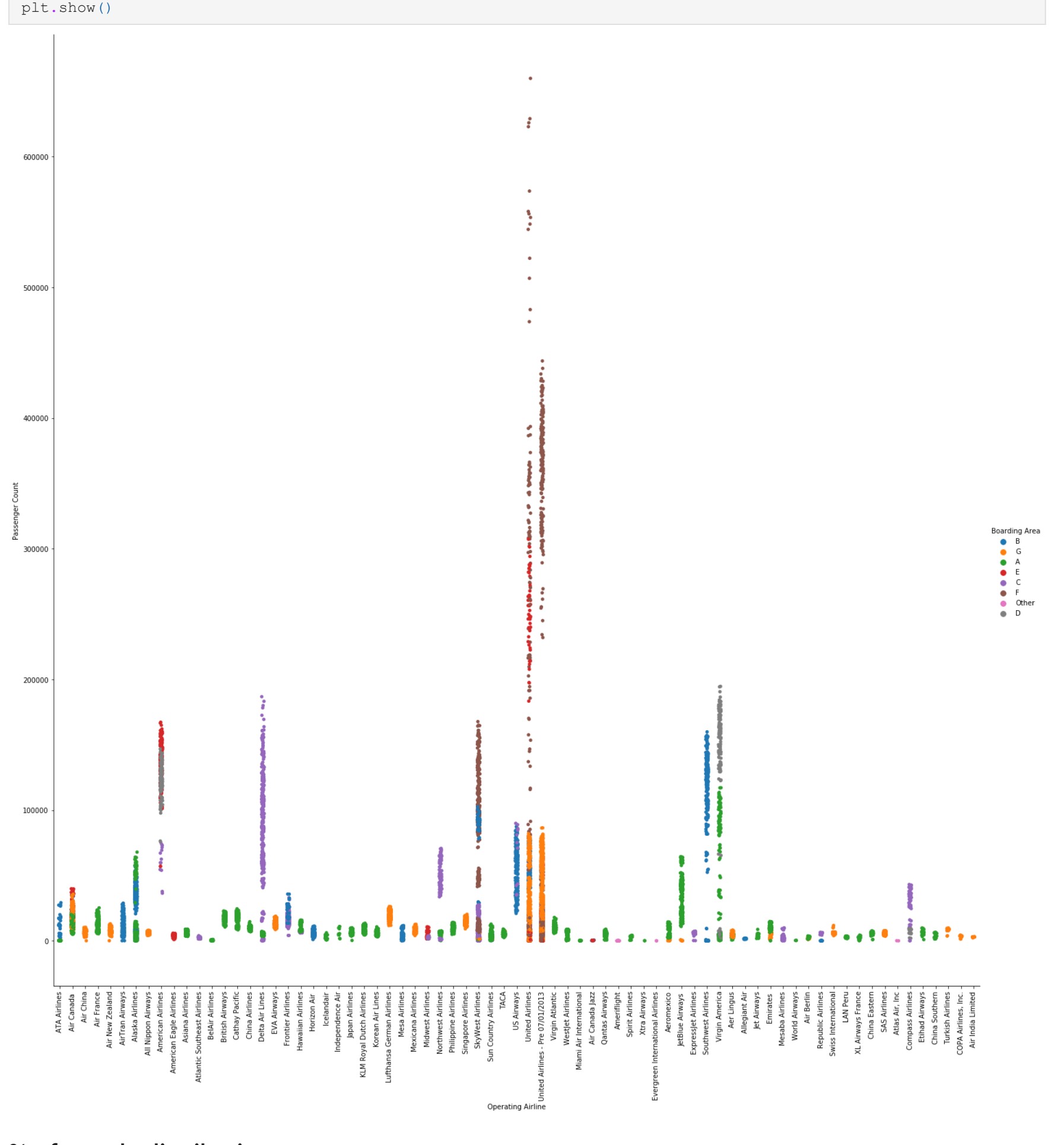






In [45]:

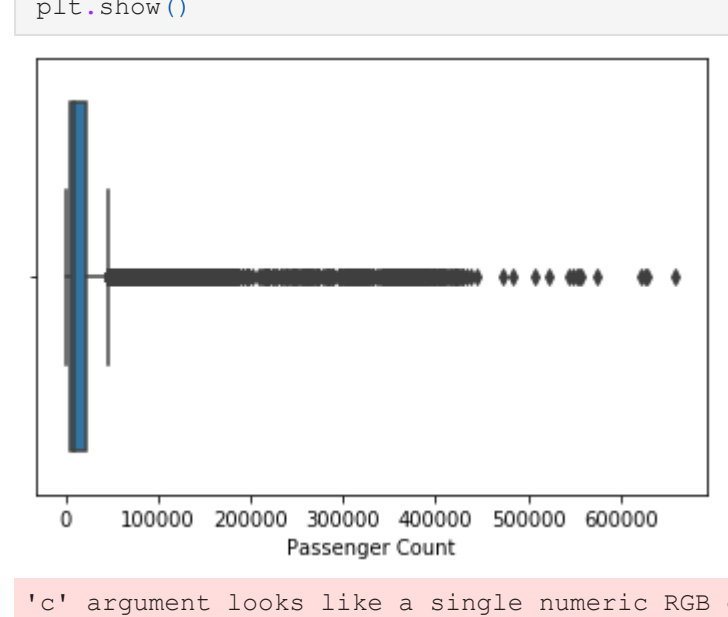
```
sns.catplot(x= 'Operating Airline',y='Passenger Count',data=data,hue='Boarding Area',height=20)
plt.xticks(rotation=90)
plt.show()
```



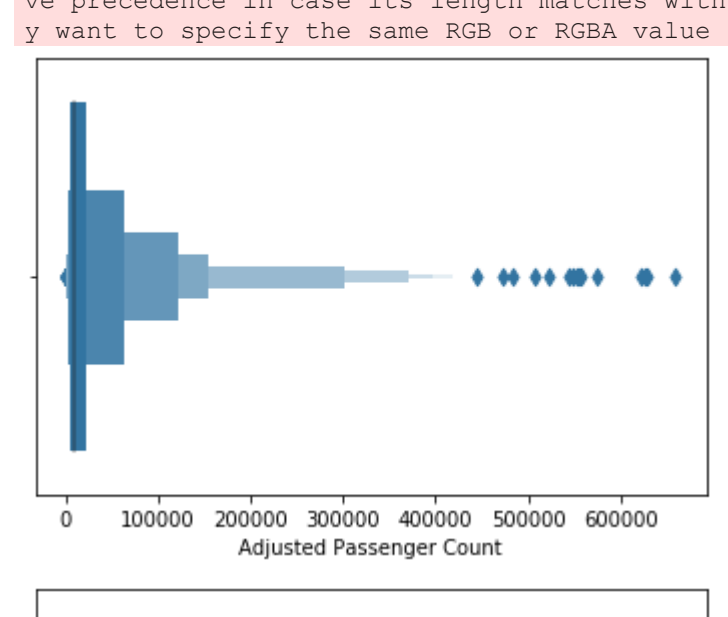
% of people distribution

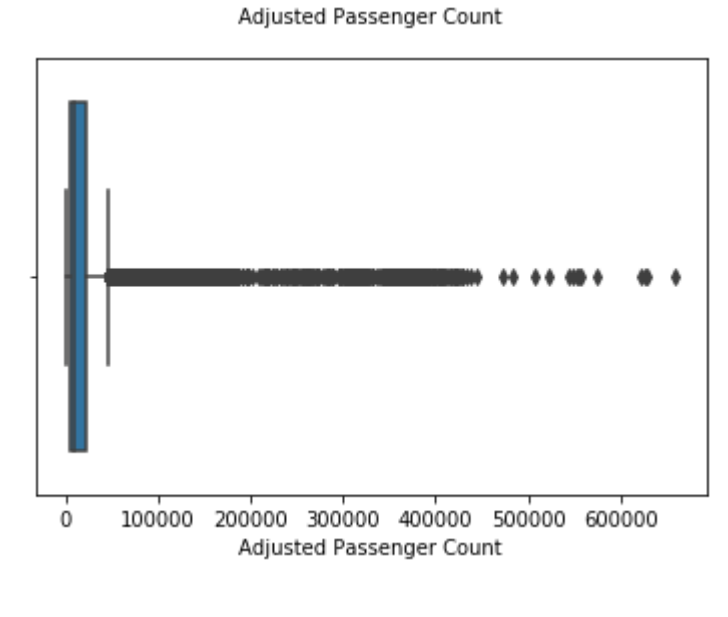
In [46]:

```
sns.boxplot(data['Passenger Count'])
plt.show()
sns.boxenplot(data['Adjusted Passenger Count'])
plt.show()
sns.boxplot(data['Adjusted Passenger Count'])
plt.show()
```



'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with 'x' & 'y'. Please use a 2-D array with a single row if you really want to specify the same RGB or RGBA value for all points.



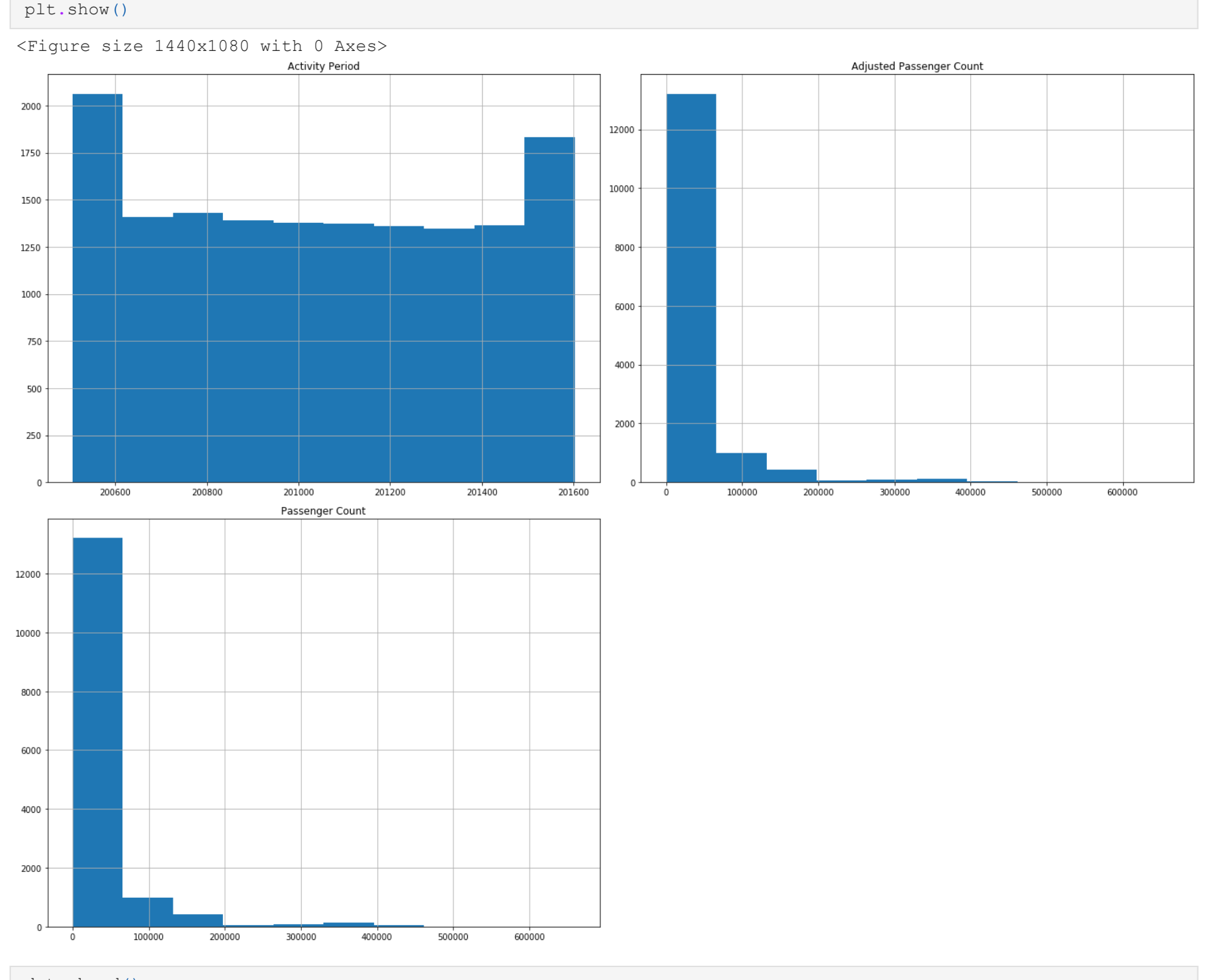


Histogram (distribution of data's columns)

In [47]:

```
plt.figure(figsize=(20,15))
data.hist(figsize=(20,15))
plt.tight_layout()
plt.xticks(rotation=90)
plt.show()
```

<Figure size 1440x1080 with 0 Axes>



In [48]:

```
data.head()
```

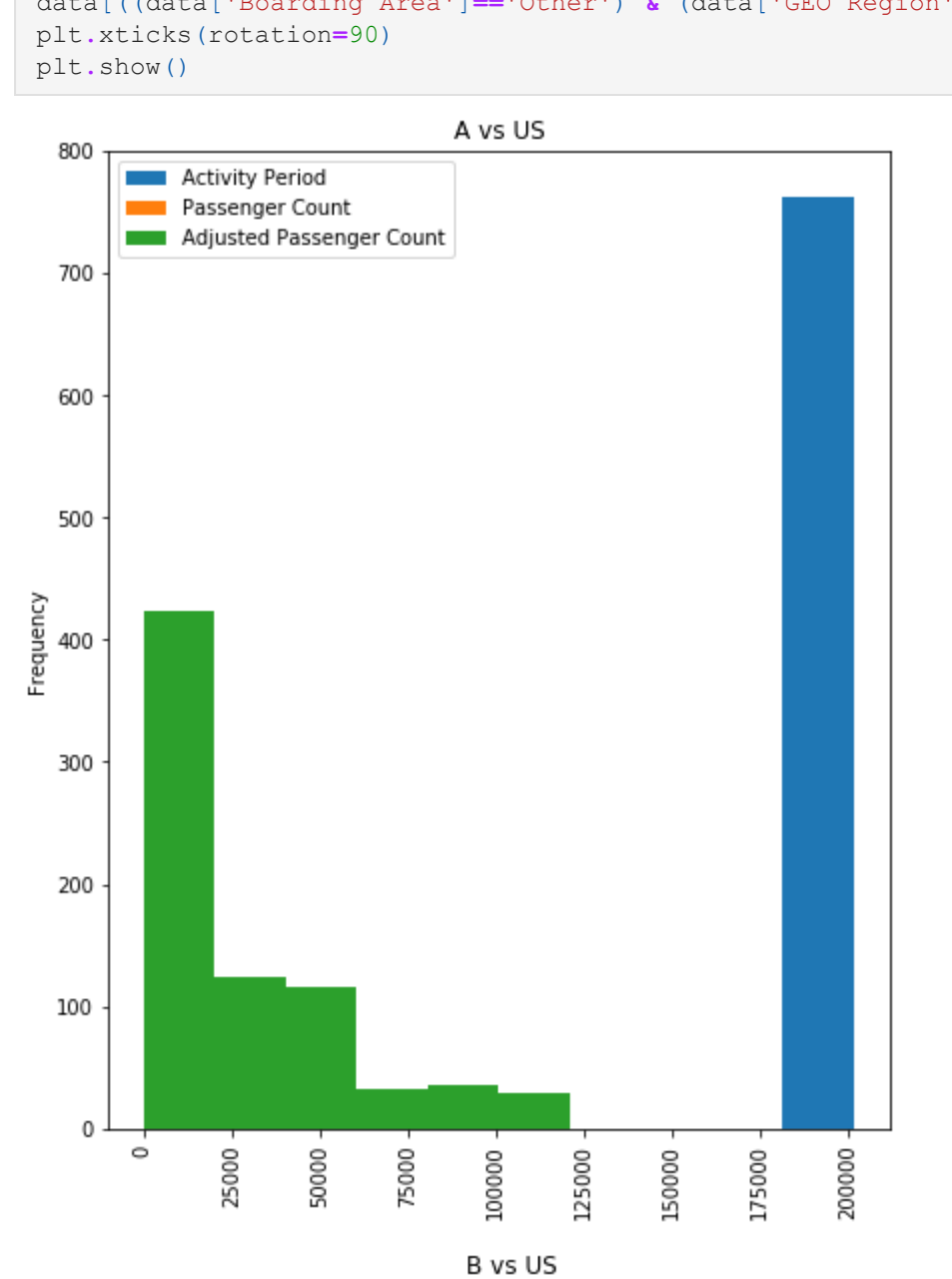
Out[48]:

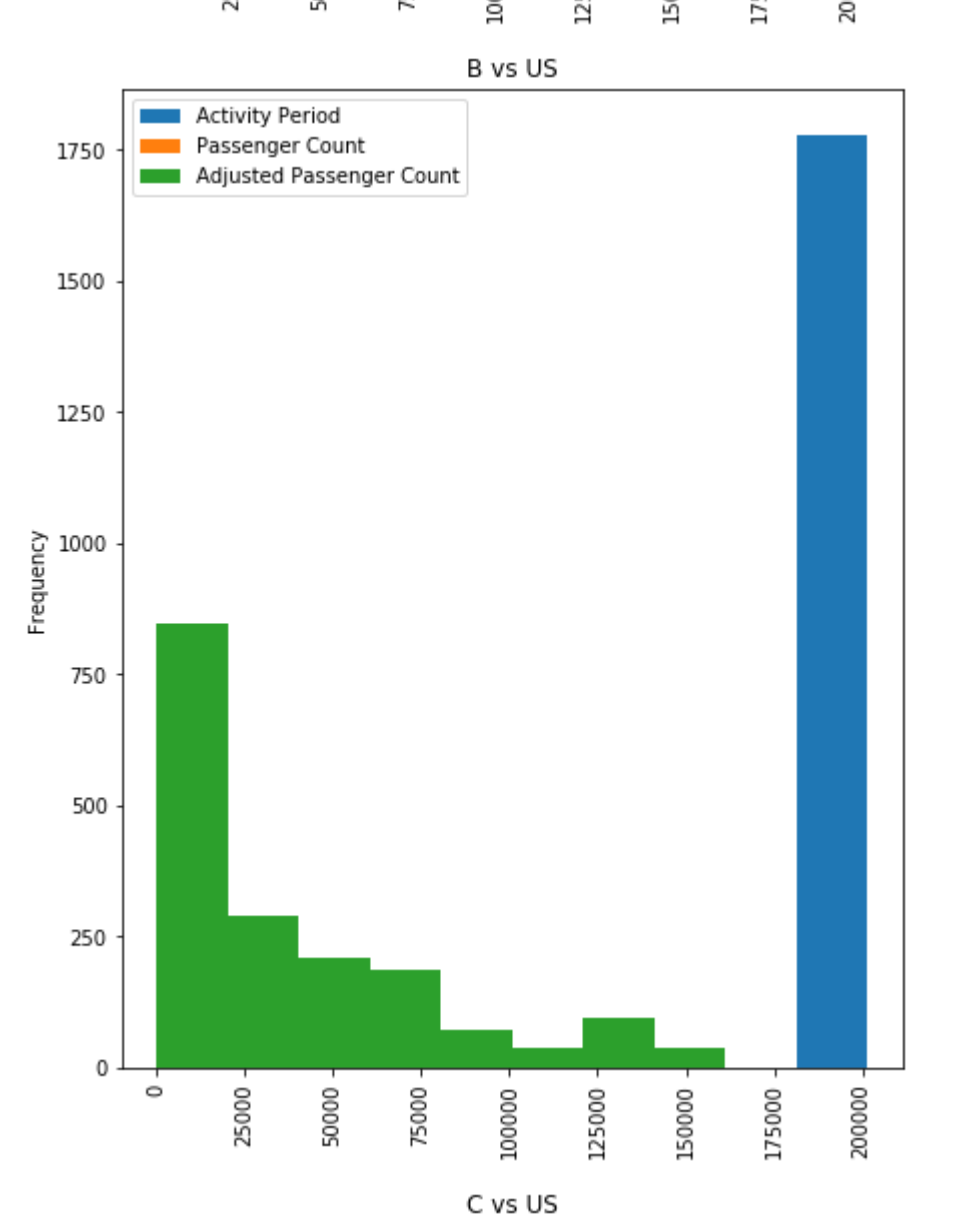
Year	Month	Activity Period	Operating Airline	Operating Airline IATA Code	Published Airline	Published Airline IATA Code	GEO Summary	GEO Region	Activity Type Code	Price Category Code	Terminal	Boarding Area	Passenger Count
	July	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Deplaned	Low Fare	Terminal 1	B	27271
	July	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Enplaned	Low Fare	Terminal 1	B	29131
2005	July	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Thru / Transit	Low Fare	Terminal 1	B	5415
	July	200507	Air Canada	AC	Air Canada	AC	International	Canada	Deplaned	Other	Terminal 1	B	35156
	July	200507	Air Canada	AC	Air Canada	AC	International	Canada	Enplaned	Other	Terminal 1	B	34090

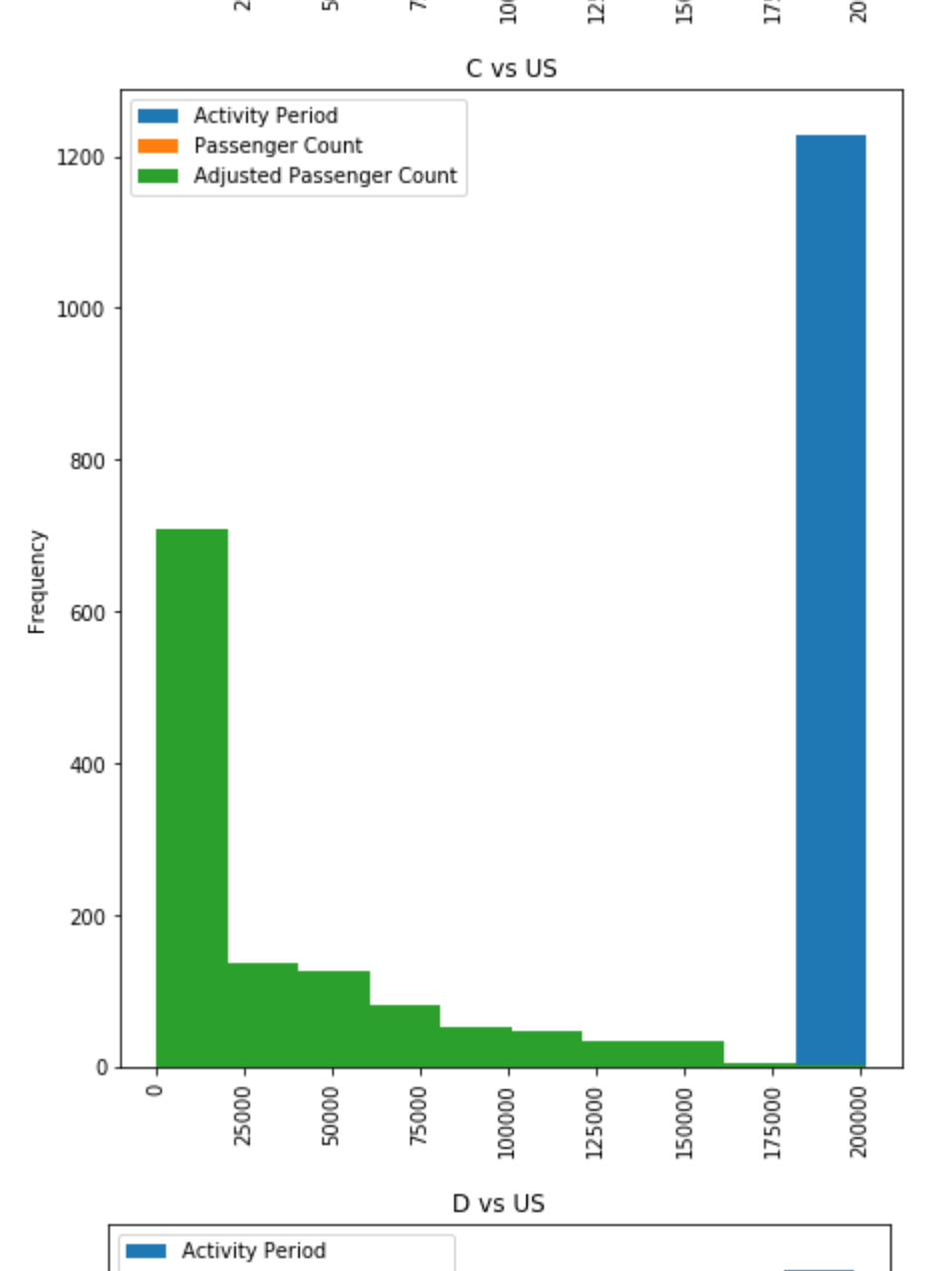
Relational Barplot between the various boarding areas and GEO Region='US'

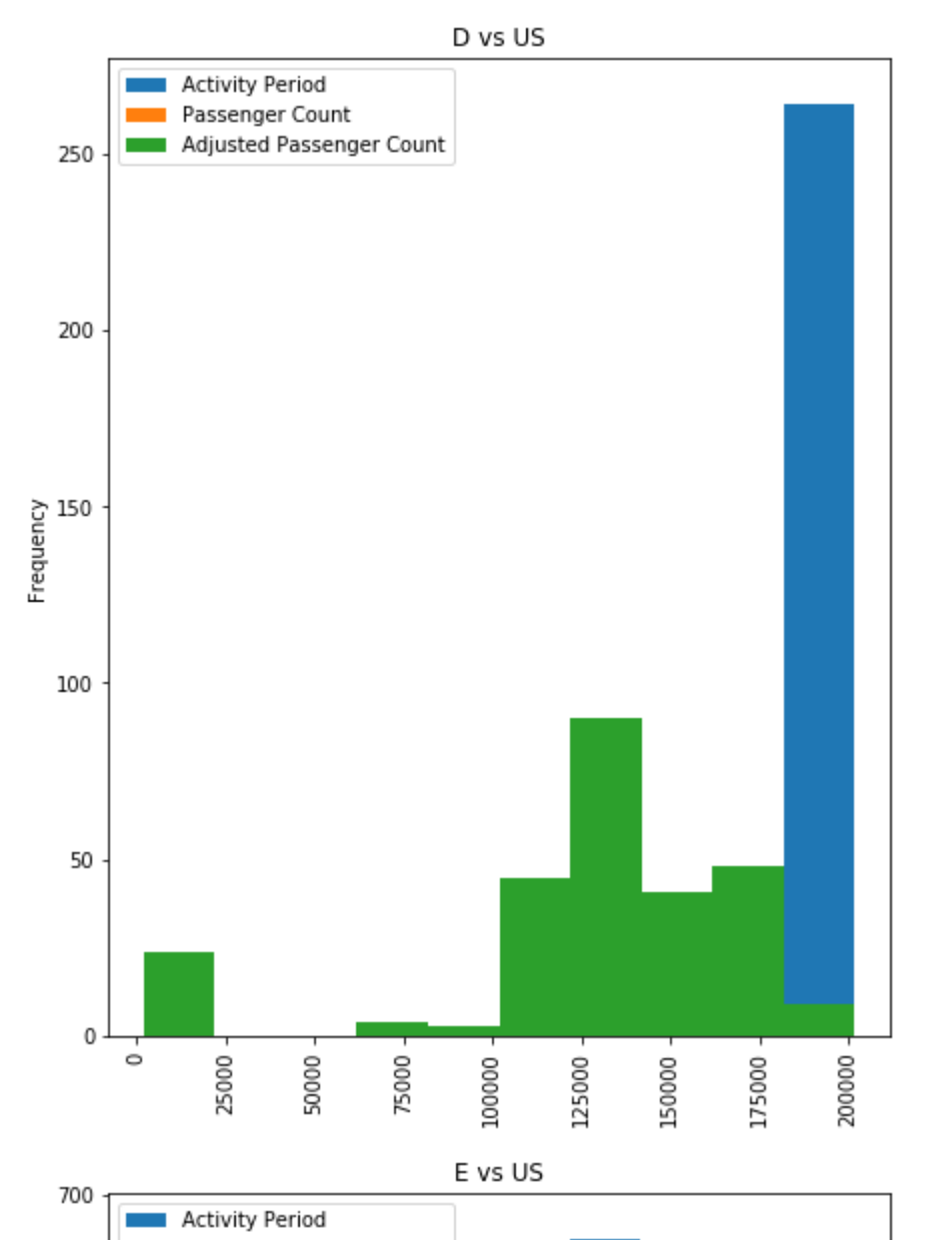
In [49]:

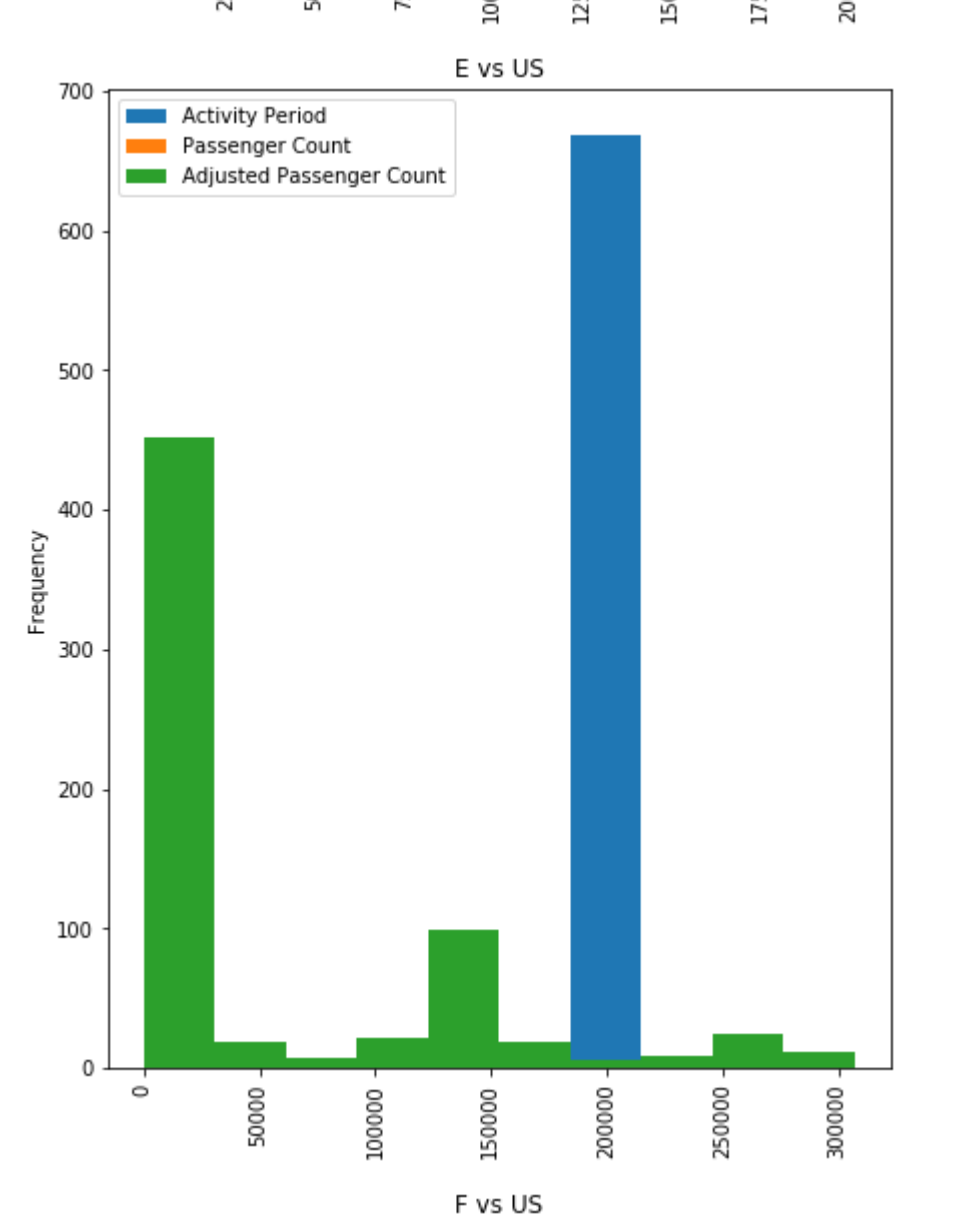
```
data[((data['Boarding Area']=='A') & (data['GEO Region']=='US'))].plot(figsize=(7,9),kind='hist',title='A vs US
plt.xticks(rotation=90)
plt.show()
data[((data['Boarding Area']=='B') & (data['GEO Region']=='US'))].plot(figsize=(7,9),kind='hist',title='B vs US
plt.xticks(rotation=90)
plt.show()
data[((data['Boarding Area']=='C') & (data['GEO Region']=='US'))].plot(figsize=(7,9),kind='hist',title='C vs US
plt.xticks(rotation=90)
plt.show()
data[((data['Boarding Area']=='D') & (data['GEO Region']=='US'))].plot(figsize=(7,9),kind='hist',title='D vs US
plt.xticks(rotation=90)
plt.show()
data[((data['Boarding Area']=='E') & (data['GEO Region']=='US'))].plot(figsize=(7,9),kind='hist',title='E vs US
plt.xticks(rotation=90)
plt.show()
data[((data['Boarding Area']=='F') & (data['GEO Region']=='US'))].plot(figsize=(7,9),kind='hist',title='F vs US
plt.xticks(rotation=90)
plt.show()
data[((data['Boarding Area']=='G') & (data['GEO Region']=='US'))].plot(figsize=(7,9),kind='hist',title='G vs US
plt.xticks(rotation=90)
plt.show()
data[((data['Boarding Area']=='Other') & (data['GEO Region']=='US'))].plot(figsize=(7,9),kind='hist',title='Oth
plt.xticks(rotation=90)
plt.show()
```

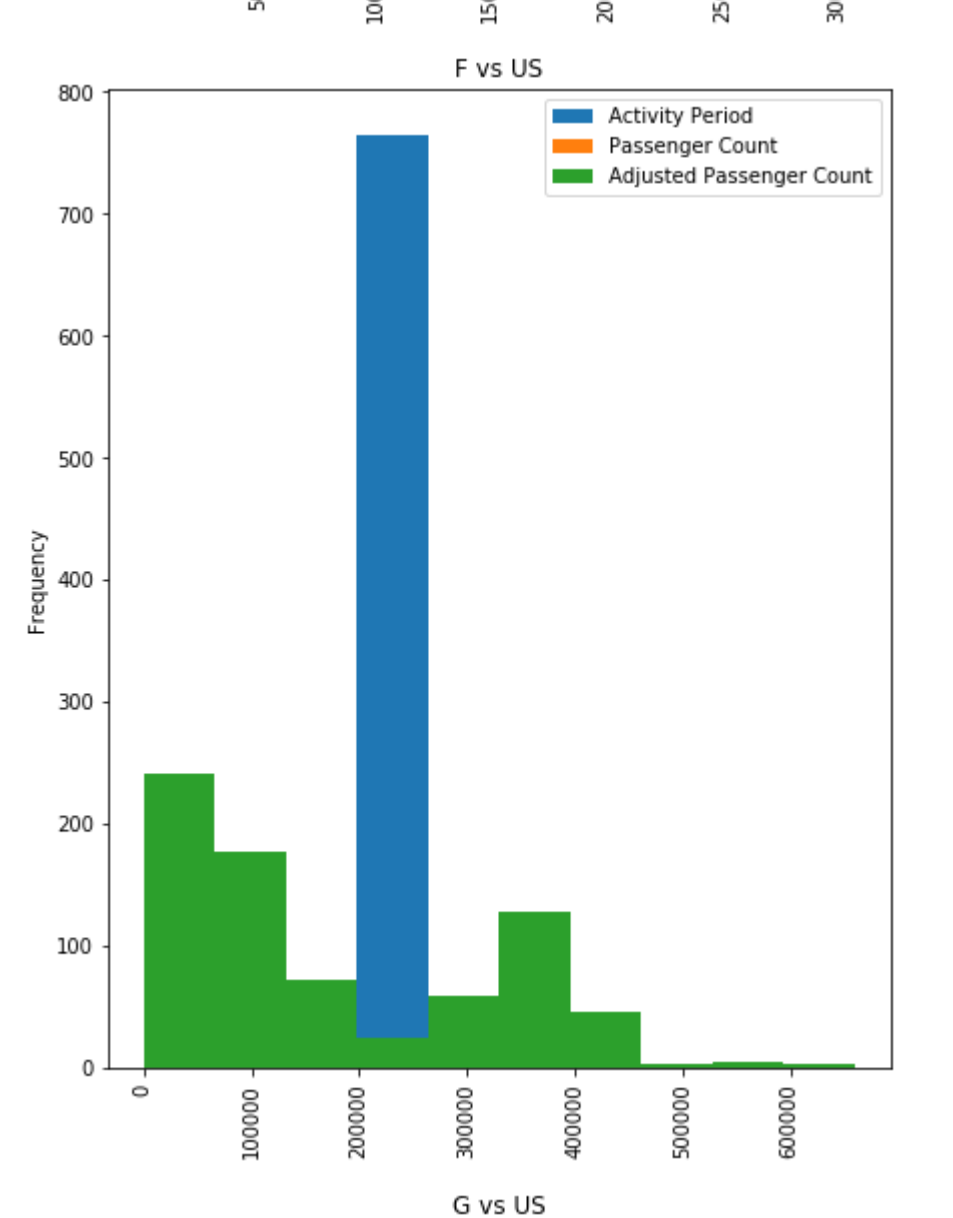


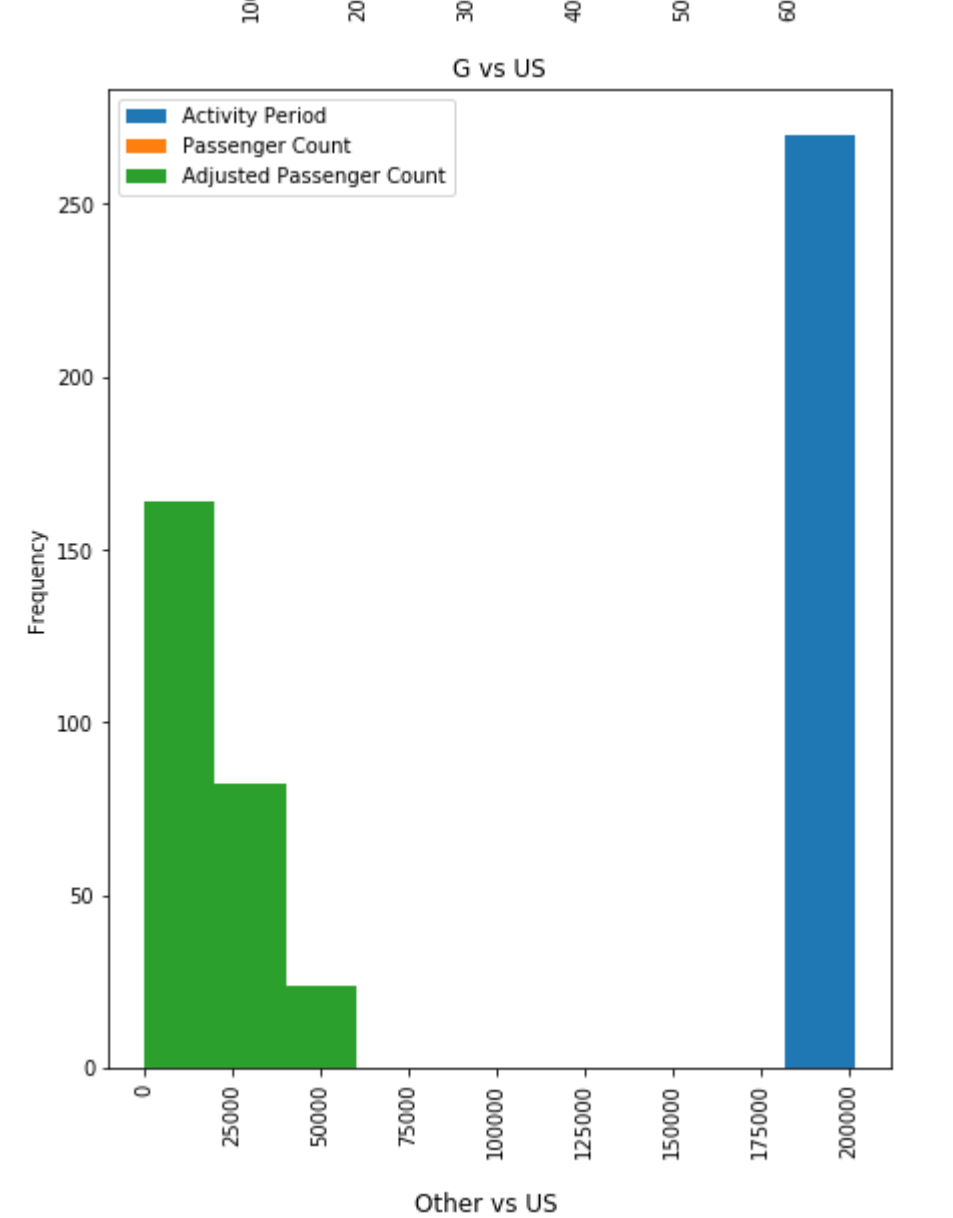


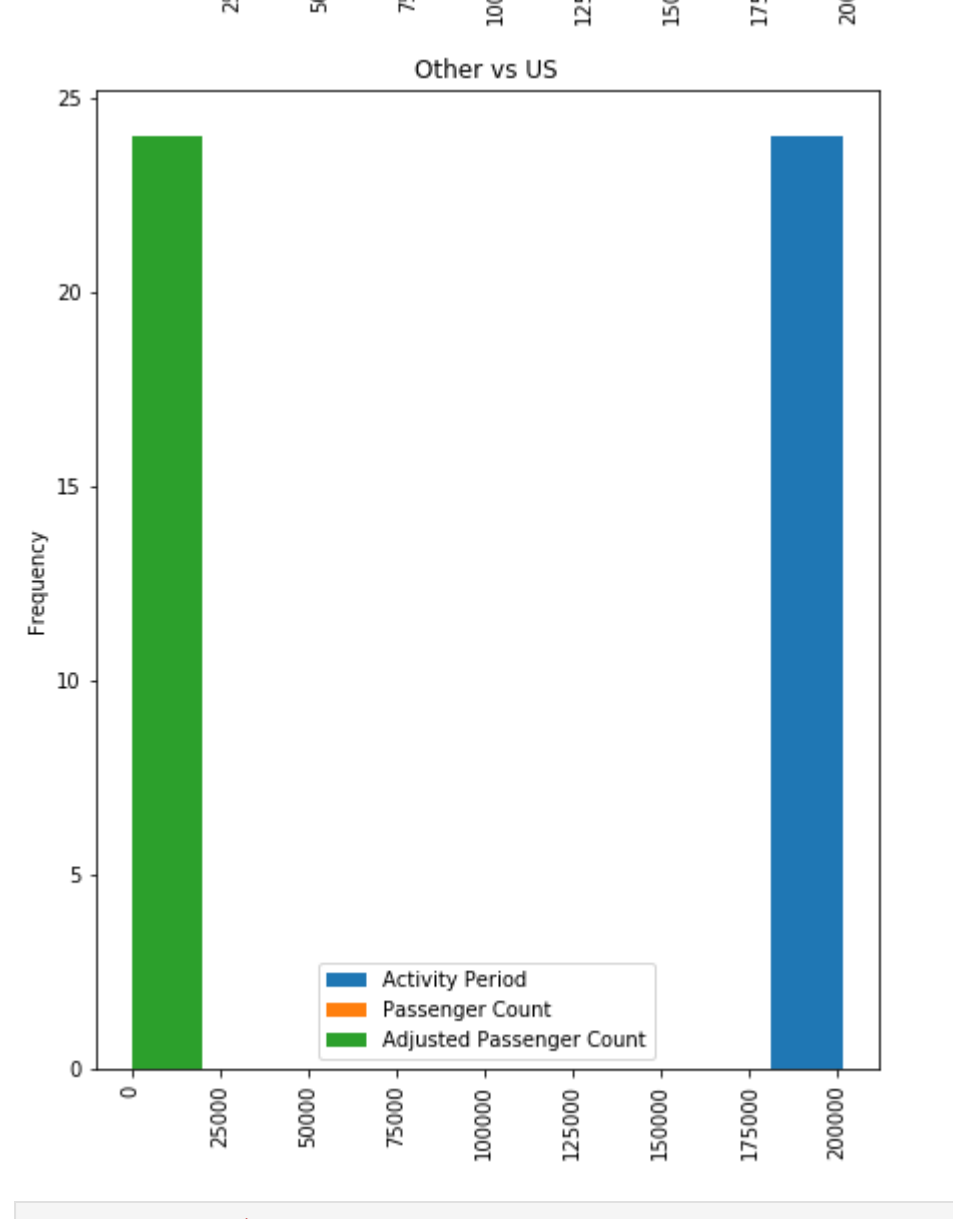












In [50]:

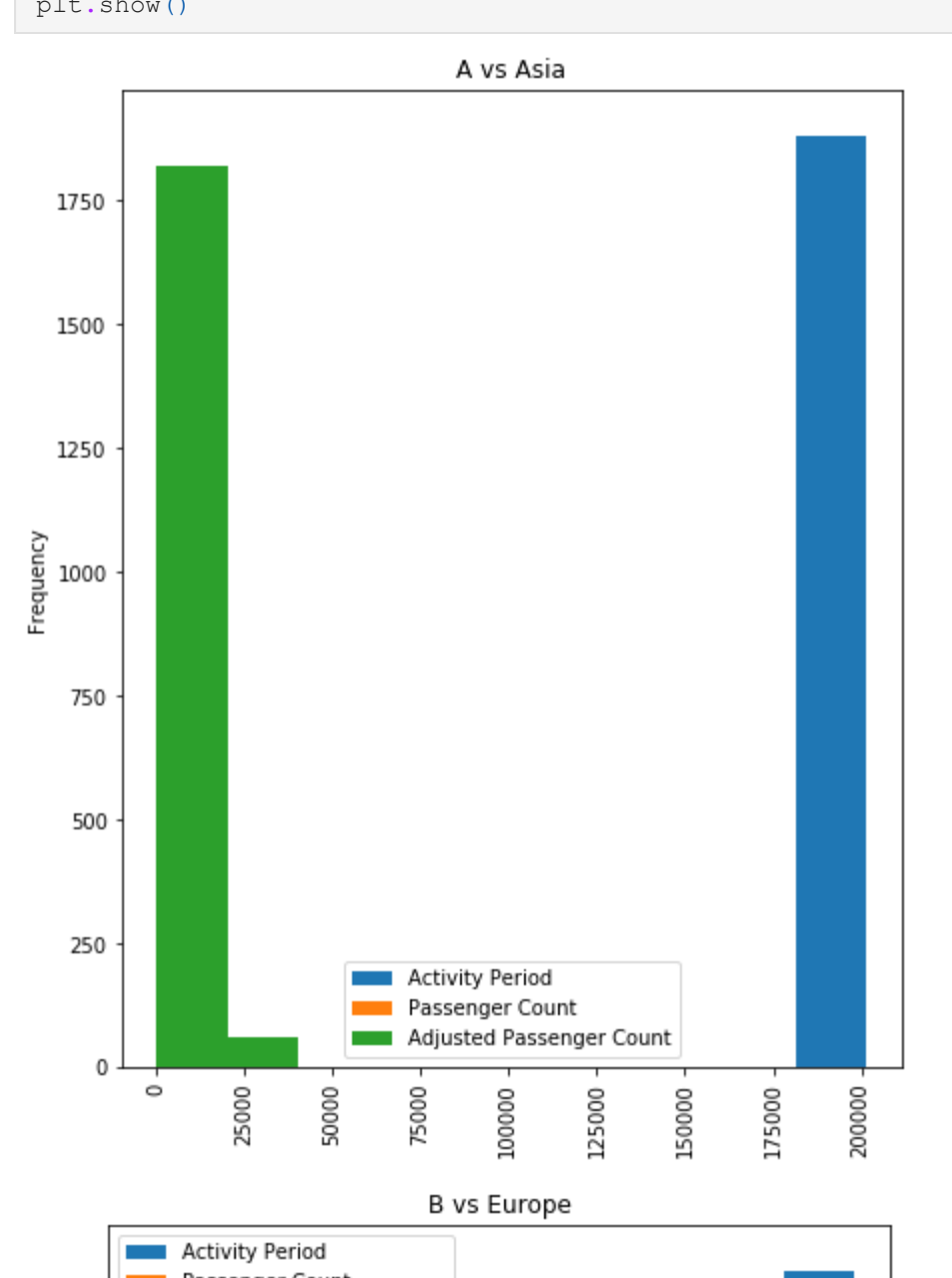
```
data['GEO Region'].value_counts()
```

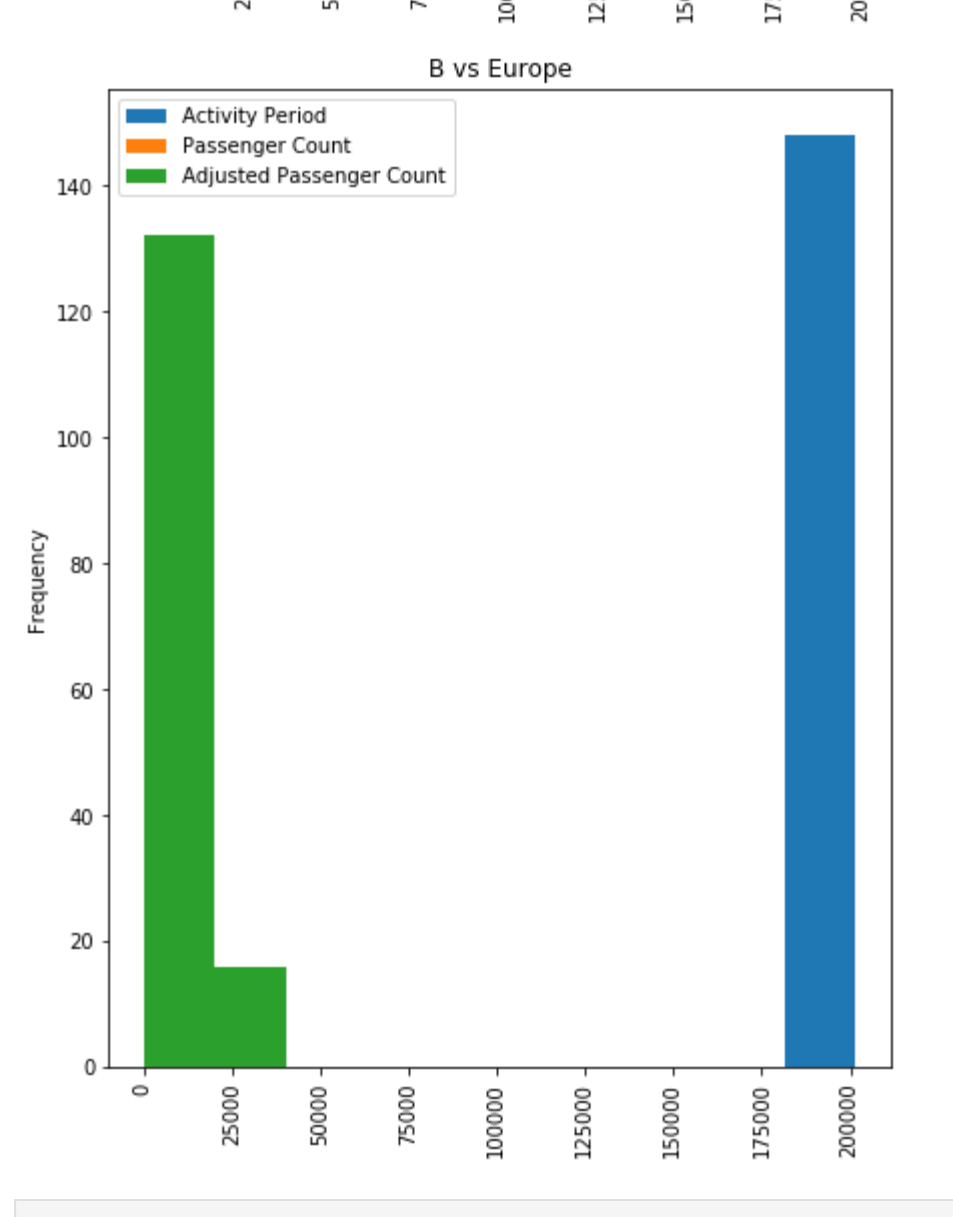
Out[50]:

```
US                     5757
Asia                   3272
Europe                 2078
Canada                 1418
Mexico                 1115
Australia / Oceania     737
Central America         272
Middle East             214
South America            90
Name: GEO Region, dtype: int64
```

In [51]:

```
data[((data['Boarding Area']=='A') & (data['GEO Region']=='Asia'))].plot(figsize=(7,9),kind='hist',title='A vs
plt.xticks(rotation=90)
plt.show()
data[((data['Boarding Area']=='B') & (data['GEO Region']=='Canada'))].plot(figsize=(7,9),kind='hist',title='B v
plt.xticks(rotation=90)
plt.show()
```





In [52]:

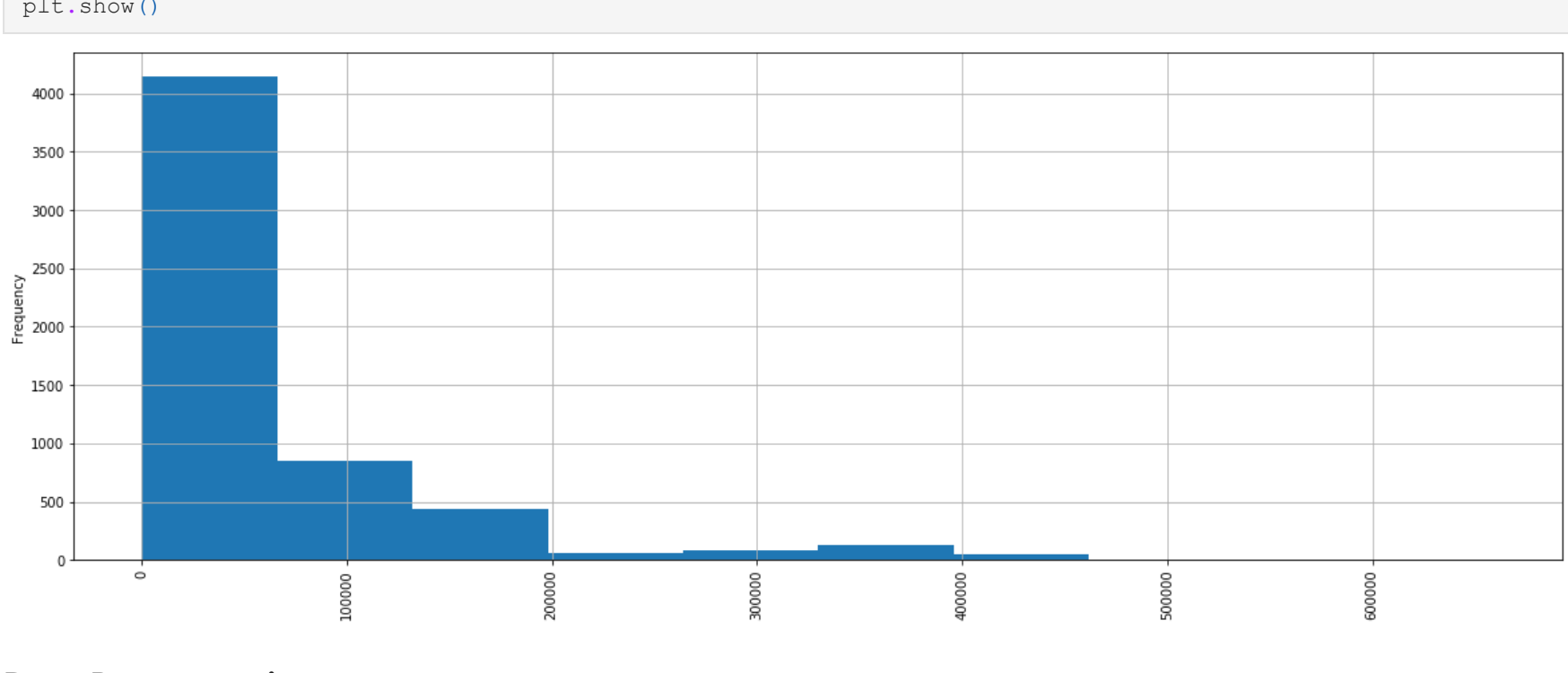
```
data.head()
```

Out[52]:

Year	Month	Activity Period	Operating Airline	Operating Airline IATA Code	Published Airline	Published Airline IATA Code	GEO Summary	GEO Region	Activity Type Code	Price Category Code	Terminal	Boarding Area	Passenger Count
2005	July	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Deplaned	Low Fare	Terminal 1	B	27271
	July	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Enplaned	Low Fare	Terminal 1	B	29131
	July	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Thru / Transit	Low Fare	Terminal 1	B	5415
	July	200507	Air Canada	AC	Air Canada	AC	International	Canada	Deplaned	Other	Terminal 1	B	35156
	July	200507	Air Canada	AC	Air Canada	AC	International	Canada	Enplaned	Other	Terminal 1	B	34090

Distribution of the GEO summary (Domestic) passenger Count.

In [53]:
```
data[data['GEO Summary']=='Domestic']['Passenger Count'].plot(figsize=(20,7),kind='hist')
plt.grid()
plt.xticks(rotation=90)
plt.show()
```



Data Preprocessing.

In [54]:
```
data.head()
```

Out[54]:

Year	Month	Activity Period	Operating Airline	Operating Airline IATA Code	Published Airline	Published Airline IATA Code	GEO Summary	GEO Region	Activity Type Code	Price Category Code	Terminal	Boarding Area	Passenger Count
2005	July	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Deplaned	Low Fare	Terminal 1	B	27271
	July	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Enplaned	Low Fare	Terminal 1	B	29131
	July	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Thru / Transit	Low Fare	Terminal 1	B	5415
	July	200507	Air Canada	AC	Air Canada	AC	International	Canada	Deplaned	Other	Terminal 1	B	35156
	July	200507	Air Canada	AC	Air Canada	AC	International	Canada	Enplaned	Other	Terminal 1	B	34090

In [55]:
```
data.drop(columns=['Activity Period'],inplace=True)
```

In [56]:
```
df =pd.get_dummies(data['Operating Airline'],drop_first=True)
dff = pd.get_dummies(data['Operating Airline IATA Code'],drop_first=True)
dfff = pd.get_dummies(data['Published Airline'],drop_first=True)
df1 = pd.get_dummies(data['GEO Summary'],drop_first=True)
df3 = pd.get_dummies(data['Price Category Code'],drop_first=True)
df5 = pd.get_dummies(data['Adjusted Activity Type Code'],drop_first=True)
df6 = pd.get_dummies(data['Terminal'],drop_first=True)
```

In [57]:
```
f = pd.concat([df,dff,dfff,df1,df3,df5,df6],axis=1)
```

In [58]:
```
data1 = pd.concat([f,data],axis=1)
```

In [59]:
```
data1.drop(columns=['Operating Airline','Operating Airline IATA Code','Published Airline','GEO Summary','Price
```

In [60]:
```
data['Boarding Area'].value_counts()
```

Out[60]:
```
A        5177
G        3987
B        1993
F        1377
C        1228
E         841
D         324
Other      26
Name: Boarding Area, dtype: int64
```

In [61]:
```
c={'A':0,'B':1,'C':2,'D':3,'E':4,'F':5,'G':6,'Other':7}
data1['Boarding Area'] = data1['Boarding Area'].map(c)
```

In [62]:
```
data1.head()
```

Out[62]:

Year	Month	Aer Lingus	Aeromexico	Air Berlin	Air Canada	Air Canada Jazz	Air China	Air France	Air India Limited	Air New Zealand	AirTran Airways	...	Terminal 1	Terminal 2	Terminal 3
2005	July	0	0	0	0	0	0	0	0	0	0	...	1	0	0
	July	0	0	0	0	0	0	0	0	0	0	...	1	0	0
	July	0	0	0	0	0	0	0	0	0	0	...	1	0	0
	July	0	0	0	1	0	0	0	0	0	0	...	1	0	0
	July	0	0	0	1	0	0	0	0	0	0	...	1	0	0

5 rows × 222 columns

In [63]:
```
corr=data1.corr()['Boarding Area']
corr
```

Out[63]:
```
Aer Lingus                 0.008905
Aeromexico                -0.106770
Air Berlin                -0.012748
Air Canada                 0.037858
Air Canada Jazz            0.000654
                             ...
Terminal 2                 0.019900
Terminal 3                 0.321781
Boarding Area              1.000000
Passenger Count            0.130034
Adjusted Passenger Count   0.131088
Name: Boarding Area, Length: 218, dtype: float64
```

In [64]:
```
corr = abs(corr)
```

In [65]:
```
good_corr = corr[corr>0.2]
```

In [66]:
```
len(good_corr)
```

Out[66]: 9

In [67]:
```
good_corr.drop(columns=['Boarding Area'],inplace=True)
```

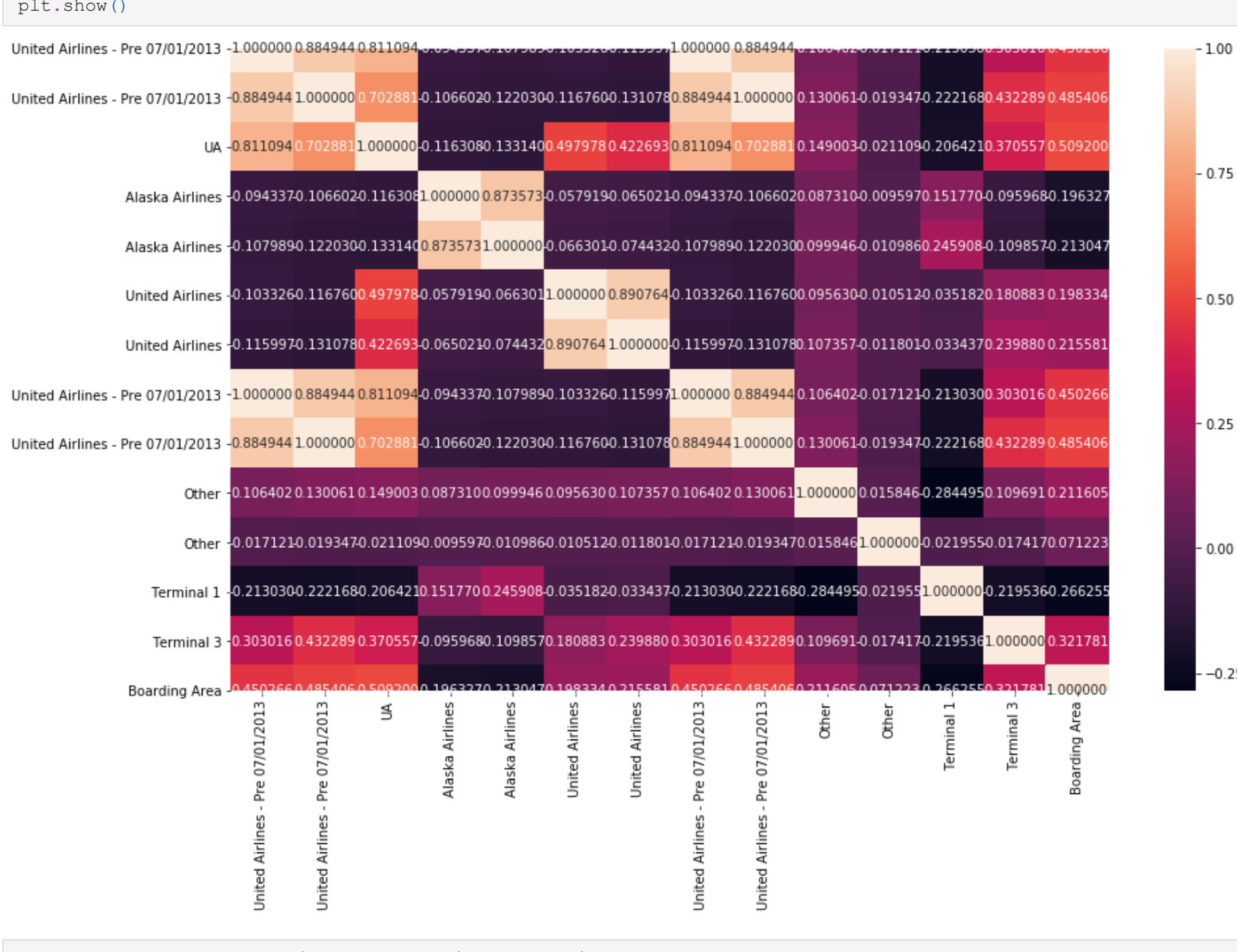
In [68]:
```
corr_data = data1[good_corr.index]
```

In [69]:
```
corr_data.corr()
```

Out[69]:

	United Airlines - Pre 07/01/2013	United Airlines - Pre 07/01/2013	UA	Alaska Airlines	Alaska Airlines	United Airlines	United Airlines	United Airlines - Pre 07/01/2013	United Airlines - Pre 07/01/2013	Other	Other
United Airlines - Pre 07/01/2013	1.000000	0.884944	0.811094	-0.094337	-0.107989	-0.103326	-0.115997	1.000000	0.884944	0.106402	-0.017121
United Airlines - Pre 07/01/2013	0.884944	1.000000	0.702881	-0.106602	-0.122030	-0.116760	-0.131078	0.884944	1.000000	0.130061	-0.019347
UA	0.811094	0.702881	1.000000	-0.116308	-0.133140	0.497978	0.422693	0.811094	0.702881	0.149003	-0.021109
Alaska Airlines	-0.094337	-0.106602	-0.116308	1.000000	0.873573	-0.057919	-0.065021	-0.094337	-0.106602	0.087310	-0.009597
Alaska Airlines	-0.107989	-0.122030	-0.133140	0.873573	1.000000	-0.066301	-0.074432	-0.107989	-0.122030	0.099946	-0.010986
United Airlines	-0.103326	-0.116760	0.497978	-0.057919	-0.066301	1.000000	0.890764	-0.103326	-0.116760	0.095630	-0.010512
United Airlines	-0.115997	-0.131078	0.422693	-0.065021	-0.074432	0.890764	1.000000	-0.115997	-0.131078	0.107357	-0.011801
United Airlines - Pre 07/01/2013	1.000000	0.884944	0.811094	-0.094337	-0.107989	-0.103326	-0.115997	1.000000	0.884944	0.106402	-0.017121
United Airlines - Pre 07/01/2013	0.884944	1.000000	0.702881	-0.106602	-0.122030	-0.116760	-0.131078	0.884944	1.000000	0.130061	-0.019347
Other	0.106402	0.130061	0.149003	0.087310	0.099946	0.095630	0.107357	0.106402	0.130061	1.000000	0.015846
Other	-0.017121	-0.019347	-0.021109	-0.009597	-0.010986	-0.010512	-0.011801	-0.017121	-0.019347	0.015846	1.000000
Terminal 1	-0.213030	-0.222168	-0.206421	0.151770	0.245908	-0.035182	-0.033437	-0.213030	-0.222168	-0.284495	-0.021955
Terminal 3	0.303016	0.432289	0.370557	-0.095968	-0.109857	0.180883	0.239880	0.303016	0.432289	0.109691	-0.017417
Boarding Area	0.450266	0.485406	0.509200	-0.196327	-0.213047	0.198334	0.215581	0.450266	0.485406	0.211605	0.071223

In [70]:
```
plt.figure(figsize=(15,9))
sns.heatmap(corr_data.corr(),annot=True,fmt='f')
plt.show()
```



In [71]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
from sklearn.ensemble import AdaBoostClassifier,GradientBoostingClassifier,RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV,RandomizedSearchCV
```

In [72]:
```
x = corr_data.drop(columns=['Boarding Area'])
y = corr_data['Boarding Area']
```

In [73]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=0.80,random_state =42,stratify=y)
```

In [74]:
```
x_train.shape,y_train.shape
```

Out[74]: ((11962, 13), (11962,))

In [75]:
```
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

In [76]:
```
y_train = y_train.to_numpy()
```

In [77]:
```
print(len(y_train[y_train==0]))
print(len(y_train[y_train==1]))
print(len(y_train[y_train==2]))
print(len(y_train[y_train==3]))
print(len(y_train[y_train==4]))
print(len(y_train[y_train==5]))
print(len(y_train[y_train==6]))
print(len(y_train[y_train==7]))
```

```
4141
1594
982
259
673
1102
3190
21
```

In [78]:
```
c={'A':0,'B':1,'C':2,'D':3,'E':4,'F':5,'G':6,'Other':7}
data['Boarding Area'] = data['Boarding Area'].map(c)
```

In [79]:
```
from imblearn.over_sampling import SMOTE
```

```
Using TensorFlow backend.
```

In [80]:
```
sm = SMOTE(sampling_strategy='all',random_state=42,k_neighbors=5)
```

In [81]:
```
x_train_res,y_train_res = sm.fit_sample(x_train,y_train.ravel())
```

In [82]:
```
print(len(y_train_res[y_train_res==0]))
print(len(y_train_res[y_train_res==1]))
print(len(y_train_res[y_train_res==2]))
print(len(y_train_res[y_train_res==3]))
print(len(y_train_res[y_train_res==4]))
print(len(y_train_res[y_train_res==5]))
print(len(y_train_res[y_train_res==6]))
print(len(y_train_res[y_train_res==7]))
```

```
4141
4141
4141
4141
4141
4141
4141
4141
```

In [83]:
```
model = RandomForestClassifier(n_estimators=200,random_state=42)
model.fit(x_train_res,y_train_res)
y_pred = model.predict(x_test)
print('Accuracy :-',accuracy_score(y_test,y_pred))
```

```
Accuracy :- 0.7372116349047142
```

In [84]:
```
classifier = RandomForestClassifier()
```

In [85]:
```
from scipy.stats import randint
```

In [86]:
```
params = {'max_depth':[3,None],'min_samples_leaf':randint(1,9),'criterion':['gini','entropy']}
```

In [87]:
```
cv_classifier = RandomizedSearchCV(classifier,params,scoring='accuracy',cv=5)
```

In [88]:
```
cv_classifier.fit(x_train_res,y_train_res)
```

Out[88]:
```
RandomizedSearchCV(cv=5, error_score=nan,
                   estimator=RandomForestClassifier(bootstrap=True,
                                                    ccp_alpha=0.0,
                                                    class_weight=None,
                                                    criterion='gini',
                                                    max_depth=None,
                                                    max_features='auto',
                                                    max_leaf_nodes=None,
                                                    max_samples=None,
                                                    min_impurity_decrease=0.0,
                                                    min_impurity_split=None,
                                                    min_samples_leaf=1,
                                                    min_samples_split=2,
                                                    min_weight_fraction_leaf=0.0,
                                                    n_estimators=100,
                                                    n_jobs...
                                                    oob_score=False,
                                                    random_state=None,
                                                    verbose=0,
                                                    warm_start=False),
                   iid='deprecated', n_iter=10, n_jobs=None,
                   param_distributions={'criterion': ['gini', 'entropy'],
                                        'max_depth': [3, None],
                                        'min_samples_leaf': <scipy.stats._distn_infrastructure.rv_frozen object
at 0x0D977870>},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=False, scoring='accuracy', verbose=0)
```

In [91]:
```
y_prediction=cv_classifier.predict(x_test)
```

In [92]:
```
cv_classifier.best_params_
```

Out[92]: {'criterion': 'gini', 'max_depth': None, 'min_samples_leaf': 1}

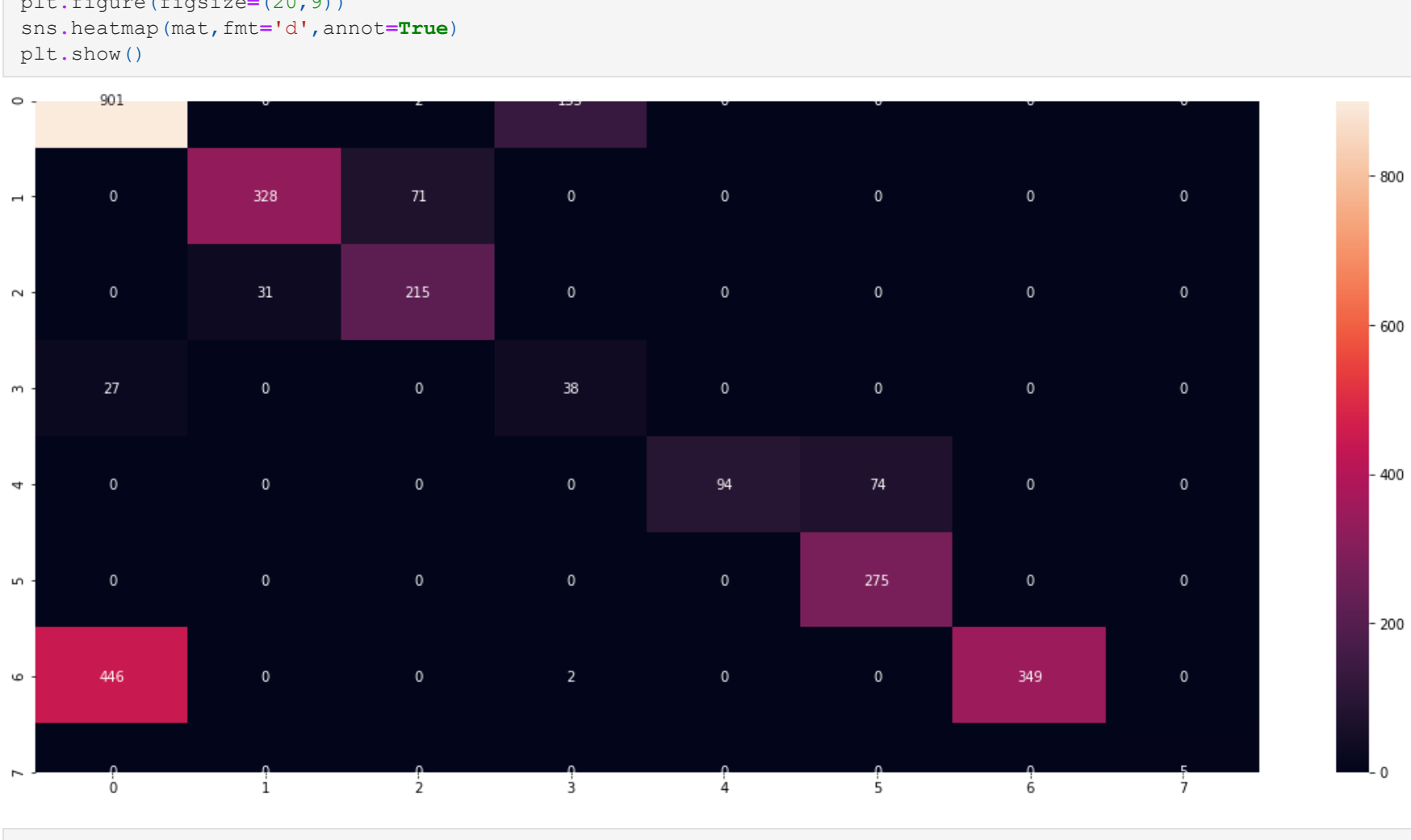
In [93]:
```
cv_classifier.best_score_
```

Out[93]: 0.75226421700676

In [95]:
```
mat=confusion_matrix(y_test,y_prediction)
mat
```

Out[95]:
```
array([[901,   0,   2, 133,   0,   0,   0,   0],
       [  0, 328,  71,   0,   0,   0,   0,   0],
       [  0,  31, 215,   0,   0,   0,   0,   0],
       [ 27,   0,   0,  38,   0,   0,   0,   0],
       [  0,   0,   0,   0,  94,  74,   0,   0],
       [  0,   0,   0,   0,   0, 275,   0,   0],
       [446,   0,   0,   2,   0,   0, 349,   0],
       [  0,   0,   0,   0,   0,   0,   0,   5]], dtype=int64)
```

In [96]:
```
plt.figure(figsize=(20,9))
sns.heatmap(mat,fmt='d',annot=True)
plt.show()
```



In [97]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.66      0.87      0.75      1036
           1       0.91      0.82      0.87       399
           2       0.75      0.87      0.81       246
           3       0.22      0.58      0.32        65
           4       1.00      0.56      0.72       168
           5       0.79      1.00      0.88       275
           6       1.00      0.44      0.61       797
           7       1.00      1.00      1.00         5

    accuracy                           0.74      2991
   macro avg       0.79      0.77      0.74      2991
weighted avg       0.81      0.74      0.73      2991
```

To check whether model is overfitted or not.

- From the we can say that model is performing well on training as testing data also especificaly GradientBoosting Algorithm.Since this is algorithm is neither overfitted nor the underfitted.

In [98]:
```
print('accuarcy for training data :-',cv_classifier.score(x_train_res,y_train_res))
print('accuarcy for testing data :-',cv_classifier.score(x_test,y_test))
```

```
accuarcy for training data :- 0.7522639459067858
accuarcy for testing data :- 0.7372116349047142
```

- Since,our model is neither overfitted nor underfitted.

RandomForesrClassifier

In [101...
```
model = RandomForestClassifier(criterion='gini',max_depth=None,min_samples_leaf=1,random_state=42,n_estimators=
model.fit(x_train_res,y_train_res)
Y_predict = model.predict(x_test)
print('Accuarcy :',accuracy_score(y_test,Y_predict))
```

```
Accuarcy : 0.7372116349047142
```

In [102...
```
print('accuarcy for training data :-',model.score(x_train_res,y_train_res))
print('accuarcy for testing data :-',model.score(x_test,y_test))
```

```
accuarcy for training data :- 0.7522639459067858
accuarcy for testing data :- 0.7372116349047142
```

In [103...
```
from sklearn.ensemble import GradientBoostingClassifier,AdaBoostClassifier
```

GradientBoostingClassifier

In [104...
```
model_gradient = GradientBoostingClassifier(max_leaf_nodes=6,random_state=42,n_estimators=200)
model_gradient.fit(x_train_res,y_train_res)
Y_predi = model_gradient.predict(x_test)
print('Accuarcy :',accuracy_score(y_test,Y_predict))
```

```
Accuarcy : 0.7372116349047142
```

In [105...
```
print('accuarcy for training data :-',model_gradient.score(x_train_res,y_train_res))
print('accuarcy for testing data :-',model_gradient.score(x_test,y_test))
```

```
accuarcy for training data :- 0.7522639459067858
accuarcy for testing data :- 0.7372116349047142
```

AdaBoostClassifier

In [110...
```
model_ada =AdaBoostClassifier(n_estimators=300,random_state=42,learning_rate=0.0015)
model_ada.fit(x_train_res,y_train_res)
Y_pre = model_ada.predict(x_test)
print('Accuarcy :',accuracy_score(y_test,Y_pre))
```

```
Accuarcy : 0.6496155132062855
```

In [112...
```
print('accuarcy for training data :-',model_ada.score(x_train_res,y_train_res))
print('accuarcy for testing data :-',model_ada.score(x_test,y_test))
```

```
accuarcy for training data :- 0.6598345810190775
accuarcy for testing data :- 0.6496155132062855
```

From the Above we can say the accuaracy has been occure during the training is 65% and during testing the accuarcy occure is 64% that means model is neither overfitted nor underfitted.