In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set_theme(color_codes=True)
```

In [2]:
```
df = pd.read_csv('airlines_delay.csv')
df.head()
```

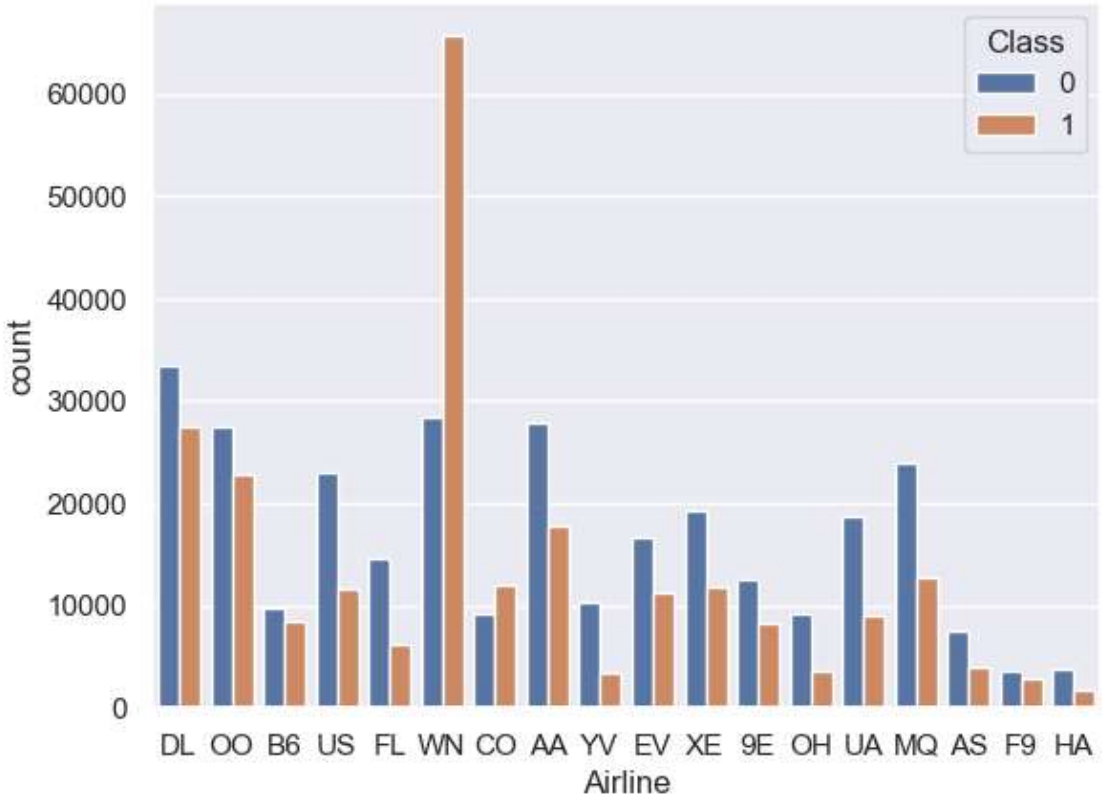
Out[2]:

	Flight	Time	Length	Airline	AirportFrom	AirportTo	DayOfWeek	Class
0	2313.0	1296.0	141.0	DL	ATL	HOU	1	0
1	6948.0	360.0	146.0	OO	COS	ORD	4	0
2	1247.0	1170.0	143.0	B6	BOS	CLT	3	0
3	31.0	1410.0	344.0	US	OGG	PHX	6	0
4	563.0	692.0	98.0	FL	BMI	ATL	4	0

Exploratory Data Analysis

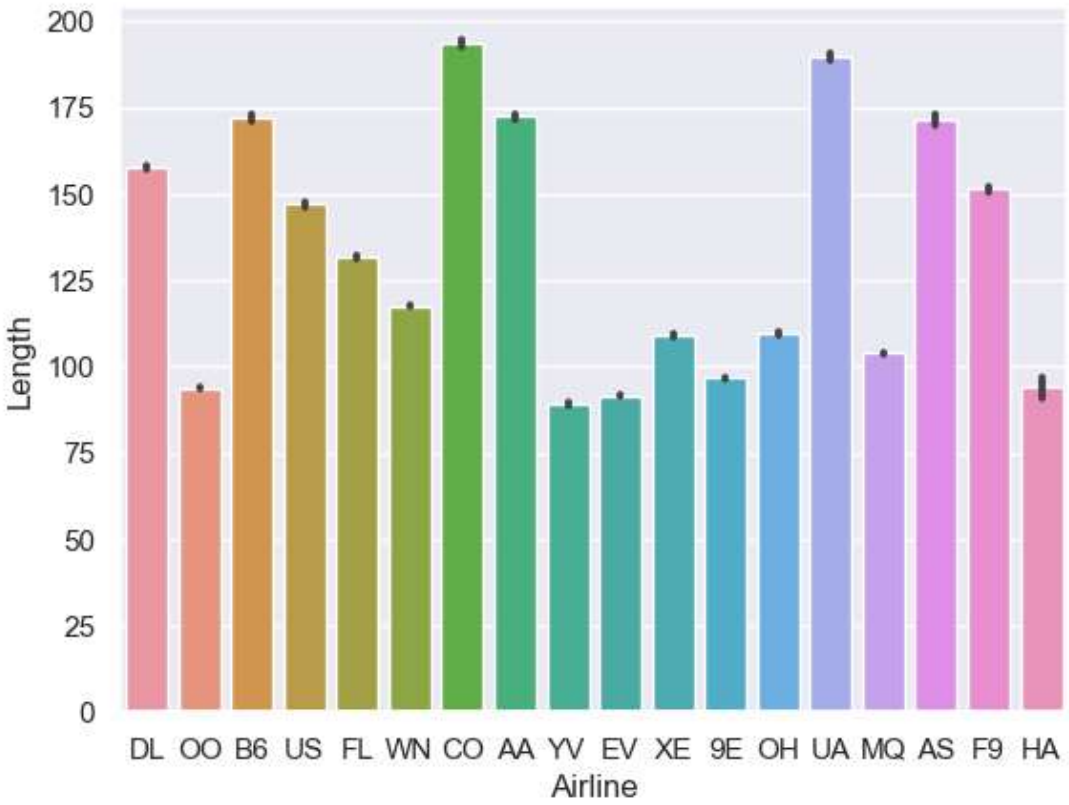
In [3]:
```
sns.countplot(data=df, x="Airline", hue="Class")
#WN Airlines are the only airline that often delayed than other airlines
```

Out[3]:
```
<AxesSubplot:xlabel='Airline', ylabel='count'>
```



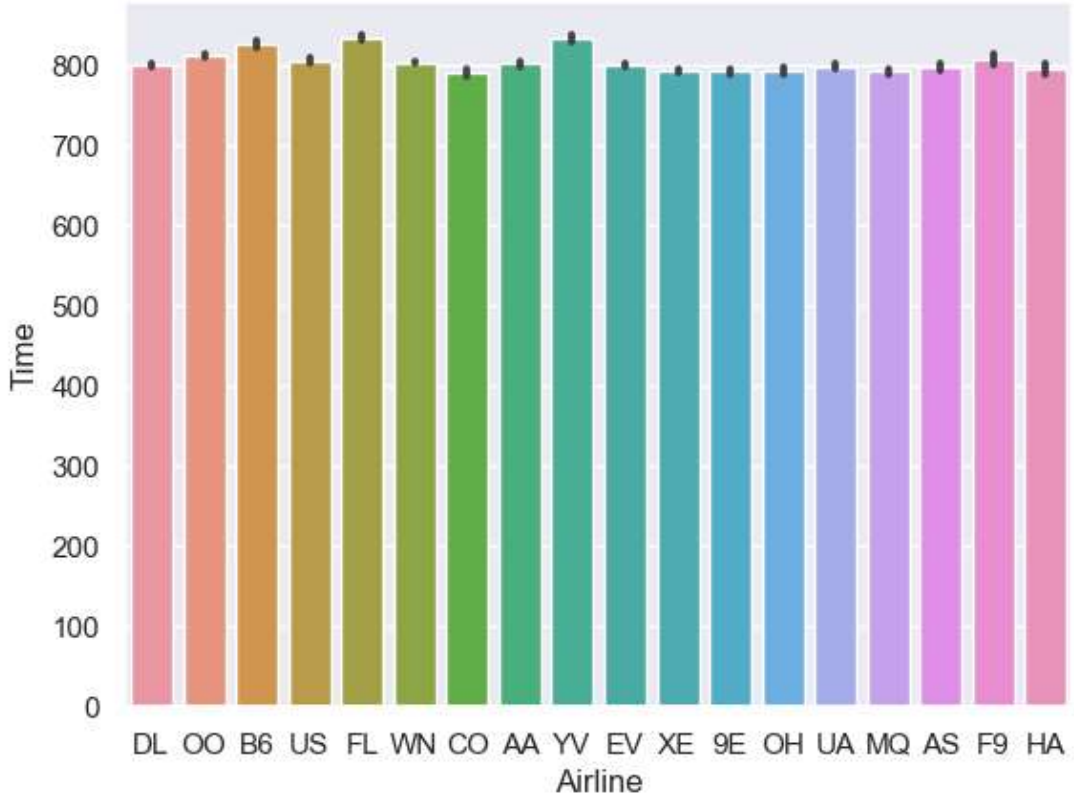
In [4]:
```
sns.barplot(data=df, x="Airline", y="Length")
#CO and UA have the longest average flight time
```

Out[4]:
```
<AxesSubplot:xlabel='Airline', ylabel='Length'>
```



In [5]:
```
sns.barplot(data=df, x="Airline", y="Time")
#Almost all of the airplane have the same departure time
```

Out[5]:
```
<AxesSubplot:xlabel='Airline', ylabel='Time'>
```



Data Preprocessing

In [6]:
```
#Remove flight ID
df = df.drop('Flight', axis=1)
df.head()
```

Out[6]:

	Time	Length	Airline	AirportFrom	AirportTo	DayOfWeek	Class
0	1296.0	141.0	DL	ATL	HOU	1	0
1	360.0	146.0	OO	COS	ORD	4	0
2	1170.0	143.0	B6	BOS	CLT	3	0
3	1410.0	344.0	US	OGG	PHX	6	0
4	692.0	98.0	FL	BMI	ATL	4	0

In [7]:
```
df['Airline'].unique()
```

Out[7]:
```
array(['DL', 'OO', 'B6', 'US', 'FL', 'WN', 'CO', 'AA', 'YV', 'EV', 'XE',
       '9E', 'OH', 'UA', 'MQ', 'AS', 'F9', 'HA'], dtype=object)
```

In [8]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
df['Airline']= label_encoder.fit_transform(df['Airline'])
df['Airline'].unique()
```

Out[8]:
```
array([ 5, 12,  3, 14,  8, 15,  4,  1, 17,  6, 16,  0, 11, 13, 10,  2,  7,
        9])
```

In [9]:
```
df['AirportFrom'].unique()
```

Out[9]:
```
array(['ATL', 'COS', 'BOS', 'OGG', 'BMI', 'MSY', 'EWR', 'DFW', 'BWI',
       'CRW', 'LGB', 'BIS', 'CLT', 'IAH', 'LAX', 'JAX', 'SAV', 'CLE',
       'FLL', 'SAN', 'BHM', 'ROC', 'DTW', 'STT', 'AUS', 'DCA', 'PHX',
       'EYW', 'IND', 'JFK', 'ORD', 'PBI', 'SFO', 'MIA', 'DSM', 'SLC',
       'PHL', 'BZN', 'GRB', 'MBS', 'SBA', 'TYS', 'MSP', 'DEN', 'SAT',
       'BUF', 'RIC', 'SEA', 'PDX', 'LAS', 'IAD', 'HNL', 'BDL', 'MOT',
       'PSE', 'CPR', 'SNA', 'STL', 'CVG', 'PIT', 'HSV', 'SGF', 'RDU',
       'MEM', 'KOA', 'ELP', 'SJU', 'JAN', 'AEX', 'LGA', 'RSW', 'MDT',
       'GUC', 'MKE', 'CAE', 'GRR', 'FAR', 'LIT', 'OMA', 'BNA', 'EVV',
       'RDD', 'OKC', 'ITO', 'SJC', 'MCO', 'LBB', 'CSG', 'OAK', 'PHF',
       'ABQ', 'SMF', 'FAY', 'ABI', 'MSO', 'MFE', 'GEG', 'MSN', 'TPA',
       'DAY', 'RNO', 'PVD', 'ALB', 'CHO', 'ONT', 'LIH', 'PSP', 'LAN',
       'LEX', 'XNA', 'GJT', 'CMH', 'GSO', 'PSC', 'SYR', 'AVL', 'MHT',
       'GRK', 'MCI', 'TXK', 'LRD', 'ABE', 'LWB', 'ERI', 'DAL', 'ANC',
       'TUS', 'ROA', 'MOD', 'JNU', 'SBP', 'CDV', 'TUL', 'FSD', 'FNT',
       'BTV', 'FCA', 'GNV', 'RAP', 'MDW', 'FWA', 'BUR', 'PNS', 'RST',
       'HOU', 'BOI', 'CRP', 'BRO', 'ATW', 'SHV', 'SMX', 'RDM', 'ORF',
       'GPT', 'KTN', 'ICT', 'SAF', 'CAK', 'IDA', 'MQT', 'VPS', 'CHS',
       'MAF', 'HPN', 'AVP', 'AZO', 'TRI', 'GSP', 'HDN', 'MLU', 'EUG',
       'AMA', 'MHK', 'ISP', 'CID', 'MOB', 'BGR', 'SRQ', 'MLI', 'EKO',
       'LFT', 'TOL', 'ECP', 'PSG', 'SBN', 'FAT', 'ELM', 'YUM', 'CLD',
       'FAI', 'ASE', 'BTR', 'BQK', 'COU', 'MRY', 'CEC', 'CWA', 'PWM',
       'FLG', 'TLH', 'SDF', 'BFL', 'CHA', 'ACV', 'MGM', 'ROW', 'GTR',
       'EWN', 'ILM', 'OTZ', 'SGU', 'OTH', 'CMX', 'SWF', 'BET', 'GTF',
       'CMI', 'MFR', 'JAC', 'DLH', 'ABY', 'MTJ', 'SCC', 'DRO', 'TEX',
       'FSM', 'COD', 'GGG', 'DBQ', 'GFK', 'BKG', 'AGS', 'BTM', 'DHN',
       'TYR', 'EGE', 'PIH', 'VLD', 'MEI', 'SIT', 'MLB', 'PAH', 'YAK',
       'DAB', 'HLN', 'PIA', 'SPI', 'GCC', 'IPL', 'TVC', 'OAJ', 'EAU',
       'BGM', 'MYR', 'HRL', 'MKG', 'SUN', 'LSE', 'CIC', 'OME', 'ITH',
       'LNK', 'BIL', 'CYS', 'LCH', 'BQN', 'WRG', 'BRW', 'SPS', 'RKS',
       'TWF', 'LMT', 'ACT', 'PLN', 'ACY', 'ADK', 'SJT', 'IYK', 'LWS',
       'BLI', 'SCE', 'MMH', 'LYH', 'GUM', 'CDC', 'ADQ', 'HTS', 'PIE',
       'STX', 'FLO', 'UTM', 'CLL', 'ABR'], dtype=object)
```

In [10]:
```
df['AirportFrom']= label_encoder.fit_transform(df['AirportFrom'])
df['AirportFrom'].unique()
```

Out[10]:
```
array([ 16,  65,  35, 203,  32, 198,  96,  80,  45,  69, 160,  29,  60,
       135, 154, 147, 245,  58, 104, 243,  27, 238,  85, 269,  18,  78,
       217,  97, 139, 148, 208, 213, 253, 183,  84, 261, 216,  46, 117,
       171, 246, 285, 197,  79, 244,  43, 234, 252, 214, 153, 134, 128,
        22, 192, 225,  67, 264, 268,  71, 221, 132, 254, 233, 177, 150,
        91, 260, 146,  10, 159, 241, 174, 124, 184,  47, 119,  99, 162,
       205,  33,  94, 231, 204, 143, 258, 173, 155,  70, 202, 215,   2,
       262, 101,   1, 196, 178, 111, 195, 277,  76, 236, 228,  12,  53,
       207, 161, 227, 152, 157, 290, 114,  61, 120, 224, 273,  19, 182,
       118, 172, 283, 165,   0, 167,  92,  75,  14, 280, 237, 191, 149,
       248,  50, 279, 107, 106,  42, 102, 115, 230, 175, 109,  44, 223,
       240, 129,  34,  68,  38,  17, 256, 263, 232, 209, 116, 151, 136,
       242,  48, 137, 193, 288,  54, 170, 130,  20,  21, 278, 121, 126,
       188,  93,  13, 181, 141,  56, 190,  26, 267, 187,  89, 158, 276,
        87, 226, 247, 100,  90, 292,  57,  98,  15,  41,  36,  66, 194,
        51,  72, 229, 103, 275, 251,  24,  52,   6, 180, 239, 123,  95,
       138, 211, 255, 210,  63, 272,  23, 122,  62, 179, 145,  82,   4,
       199, 249,  83, 274, 108,  64, 113,  77, 112,  30,  11,  40,  81,
       284,  88, 220, 287, 176, 257, 186, 212, 291,  74, 127, 218, 265,
       110, 140, 281, 201,  86,  25, 200, 131, 185, 271, 166,  55, 206,
       142, 164,  28,  73, 156,  37, 289,  39, 266, 235, 282, 163,   5,
       222,   7,   8, 259, 144, 168,  31, 250, 189, 169, 125,  49,   9,
       133, 219, 270, 105, 286,  59,   3])
```

In [11]:
```
df['AirportTo'].unique()
```

Out[11]:
```
array(['HOU', 'ORD', 'CLT', 'PHX', 'ATL', 'BHM', 'DFW', 'MEM', 'GRR',
       'PBI', 'MCO', 'SFO', 'DEN', 'YUM', 'BWI', 'HPN', 'EWR', 'JFK',
       'MKE', 'OAK', 'IAH', 'CLE', 'SYR', 'SJU', 'BDL', 'SAN', 'DTW',
       'PSP', 'DCA', 'LGA', 'STL', 'FAY', 'MSP', 'BUF', 'LAS', 'SGU',
       'SLC', 'GJT', 'LAX', 'VPS', 'FAR', 'RKS', 'BOS', 'ANC', 'SNA',
       'ONT', 'RNO', 'JAX', 'GSP', 'CVG', 'TPA', 'SEA', 'LEX', 'SMF',
       'CAE', 'STT', 'DAY', 'MDW', 'RSW', 'ITO', 'IAD', 'ICT', 'HNL',
       'MIA', 'CRW', 'RDU', 'MHT', 'FAT', 'CAK', 'COS', 'DAL', 'TYS',
       'PHL', 'ABI', 'MOB', 'SDF', 'SAV', 'MDT', 'LIT', 'TUL', 'ACV',
       'BNA', 'MCI', 'MSY', 'FLL', 'PVD', 'OKC', 'ECP', 'PHF', 'AUS',
       'RIC', 'LIH', 'ABQ', 'JAN', 'PIT', 'BMI', 'BTV', 'RAP', 'MRY',
       'CSG', 'SHV', 'FAI', 'SJC', 'PIA', 'SBN', 'IND', 'SGF', 'ACT',
       'SRQ', 'ROC', 'CHO', 'JAC', 'SAT', 'FWA', 'OMA', 'PDX', 'CMH',
       'PWM', 'CID', 'TRI', 'ORF', 'GTF', 'TUS', 'MHK', 'BUR', 'MLU',
       'CEC', 'TEX', 'MBS', 'DSM', 'HRL', 'LFT', 'ELP', 'AEX', 'CPR',
       'LBB', 'MYR', 'ALB', 'COU', 'LSE', 'CHA', 'MLI', 'GEG', 'AZO',
       'MFR', 'BTR', 'FLG', 'KTN', 'PSC', 'GSO', 'OGG', 'MSN', 'GPT',
       'PNS', 'RDM', 'BZN', 'DLH', 'CRP', 'TXK', 'KOA', 'MQT', 'MAF',
       'TLH', 'XNA', 'CWA', 'SBP', 'BFL', 'DRO', 'WRG', 'DHN', 'SPS',
       'AMA', 'EGE', 'BET', 'FCA', 'EUG', 'EVV', 'AVL', 'HSV', 'PIE',
       'MLB', 'SWF', 'ASE', 'BGM', 'MSO', 'ADK', 'GRK', 'SUN', 'SBA',
       'LGB', 'CHS', 'GNV', 'MOT', 'LAN', 'LNK', 'OME', 'OTH', 'ISP',
       'FNT', 'EAU', 'ILM', 'BRW', 'LCH', 'IYK', 'MKG', 'HDN', 'BRO',
       'GRB', 'FSD', 'LRD', 'RDD', 'SPI', 'ROA', 'IPL', 'EYW', 'SAF',
       'LWS', 'AGS', 'CMX', 'ATW', 'MGM', 'GGG', 'BOI', 'FLO', 'COD',
       'ACY', 'CMI', 'JNU', 'AVP', 'ERI', 'TYR', 'DAB', 'TVC', 'FSM',
       'IDA', 'MFE', 'EKO', 'ABE', 'PAH', 'LMT', 'YAK', 'HLN', 'MMH',
       'ITH', 'LYH', 'BIL', 'EWN', 'SMX', 'MEI', 'OAJ', 'SCE', 'CLD',
       'BIS', 'GFK', 'MTJ', 'BQN', 'BQK', 'GTR', 'CDV', 'BKG', 'PIH',
       'ROW', 'PLN', 'TWF', 'ELM', 'GCC', 'CYS', 'CDC', 'ABY', 'VLD',
       'MOD', 'STX', 'OTZ', 'HTS', 'BTM', 'PSE', 'SCC', 'RST', 'DBQ',
       'ADQ', 'PSG', 'SIT', 'GUC', 'LWB', 'BGR', 'CIC', 'GUM', 'UTM',
       'CLL', 'TOL', 'SJT', 'BLI', 'ABR'], dtype=object)
```

In [12]:
```
df['AirportTo']= label_encoder.fit_transform(df['AirportTo'])
df['AirportTo'].unique()
```

Out[12]:
```
array([129, 208,  60, 217,  16,  27,  80, 177, 119, 213, 173, 253,  79,
       292,  45, 130,  96, 148, 184, 202, 135,  58, 273, 260,  22, 243,
        85, 227,  78, 159, 268, 101, 197,  43, 153, 255, 261, 114, 154,
       288,  99, 235,  35,  14, 264, 207, 236, 147, 121,  71, 277, 252,
       157, 262,  47, 269,  76, 175, 241, 143, 134, 136, 128, 183,  69,
       233, 182, 100,  48,  65,  75, 285, 216,   1, 190, 251, 245, 174,
       162, 279,   6,  33, 172, 198, 104, 228, 204,  87, 215,  18, 234,
       161,   2, 146, 221,  32,  42, 230, 194,  70, 256,  98, 258, 218,
       247, 139, 254,   5, 267, 238,  53, 145, 244, 109, 205, 214,  61,
       229,  56, 278, 209, 122, 280, 181,  44, 188,  51, 274, 171,  84,
       131, 158,  91,  10,  67, 155, 200,  12,  66, 166,  52, 187, 111,
        21, 179,  41, 103, 151, 224, 120, 203, 195, 116, 223, 232,  46,
        82,  68, 283, 150, 193, 170, 275, 290,  72, 248,  24,  83, 289,
        81, 266,  13,  88,  23, 102,  93,  94,  19, 132, 219, 186, 272,
        15,  25, 196,   8, 118, 271, 246, 160,  54, 115, 192, 152, 164,
       206, 210, 141, 106,  86, 138,  39, 156, 144, 185, 126,  38, 117,
       107, 165, 231, 265, 237, 140,  97, 242, 168,  11,  63,  17, 180,
       113,  34, 105,  64,   7,  62, 149,  20,  92, 284,  74, 281, 108,
       137, 178,  89,   0, 212, 163, 291, 127, 189, 142, 169,  28,  95,
       263, 176, 201, 250,  57,  29, 112, 199,  37,  36, 123,  50,  30,
       220, 239, 222, 282,  90, 110,  73,  49,   4, 287, 191, 270, 211,
       133,  40, 225, 249, 240,  77,   9, 226, 257, 124, 167,  26,  55,
       125, 286,  59, 276, 259,  31,   3])
```

In [13]:
```
df.head()
```

Out[13]:

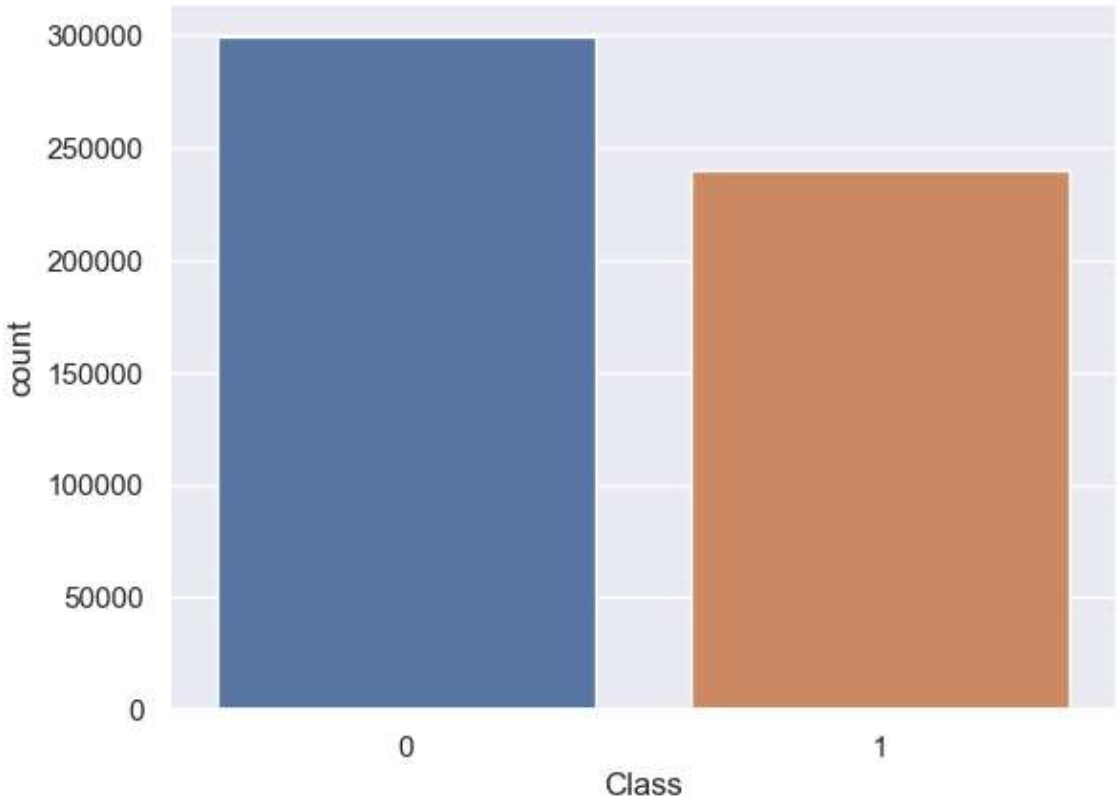
	Time	Length	Airline	AirportFrom	AirportTo	DayOfWeek	Class
0	1296.0	141.0	5	16	129	1	0
1	360.0	146.0	12	65	208	4	0
2	1170.0	143.0	3	35	60	3	0
3	1410.0	344.0	14	203	217	6	0
4	692.0	98.0	8	32	16	4	0

Check the Class Value

In [14]:
```
sns.countplot(df['Class'])
df['Class'].value_counts()
```

```
D:\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the fo
llowing variable as a keyword arg: x. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

Out[14]:
```
0    299118
1    240264
Name: Class, dtype: int64
```

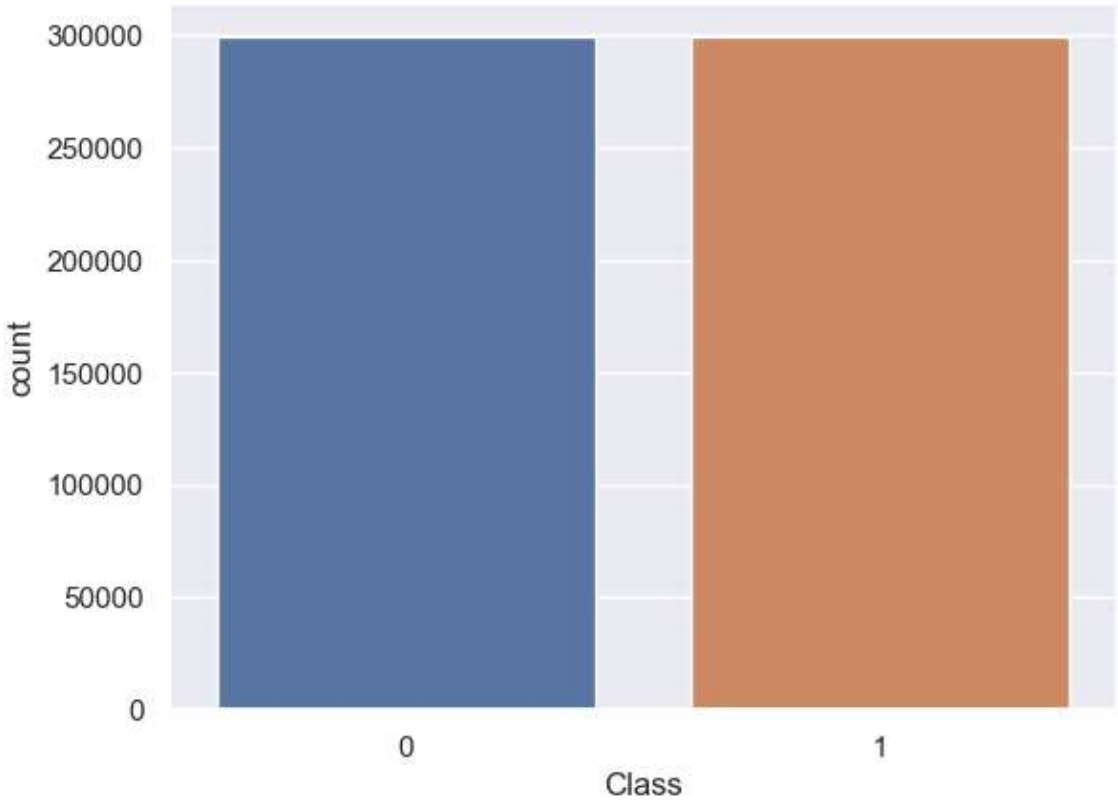


In [15]:
```
from sklearn.utils import resample
#create two different dataframe of majority and minority class
df_majority = df[(df['Class']==0)]
df_minority = df[(df['Class']==1)]
# upsample minority class
df_minority_upsampled = resample(df_minority,
                                 replace=True,    # sample with replacement
                                 n_samples= 299118, # to match majority class
                                 random_state=0)  # reproducible results
# Combine majority class with upsampled minority class
df_upsampled = pd.concat([df_minority_upsampled, df_majority])
```

In [16]:
```
sns.countplot(df_upsampled['Class'])
df_upsampled['Class'].value_counts()
```

```
D:\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the fo
llowing variable as a keyword arg: x. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

Out[16]:
```
1    299118
0    299118
Name: Class, dtype: int64
```



Check the Outliers

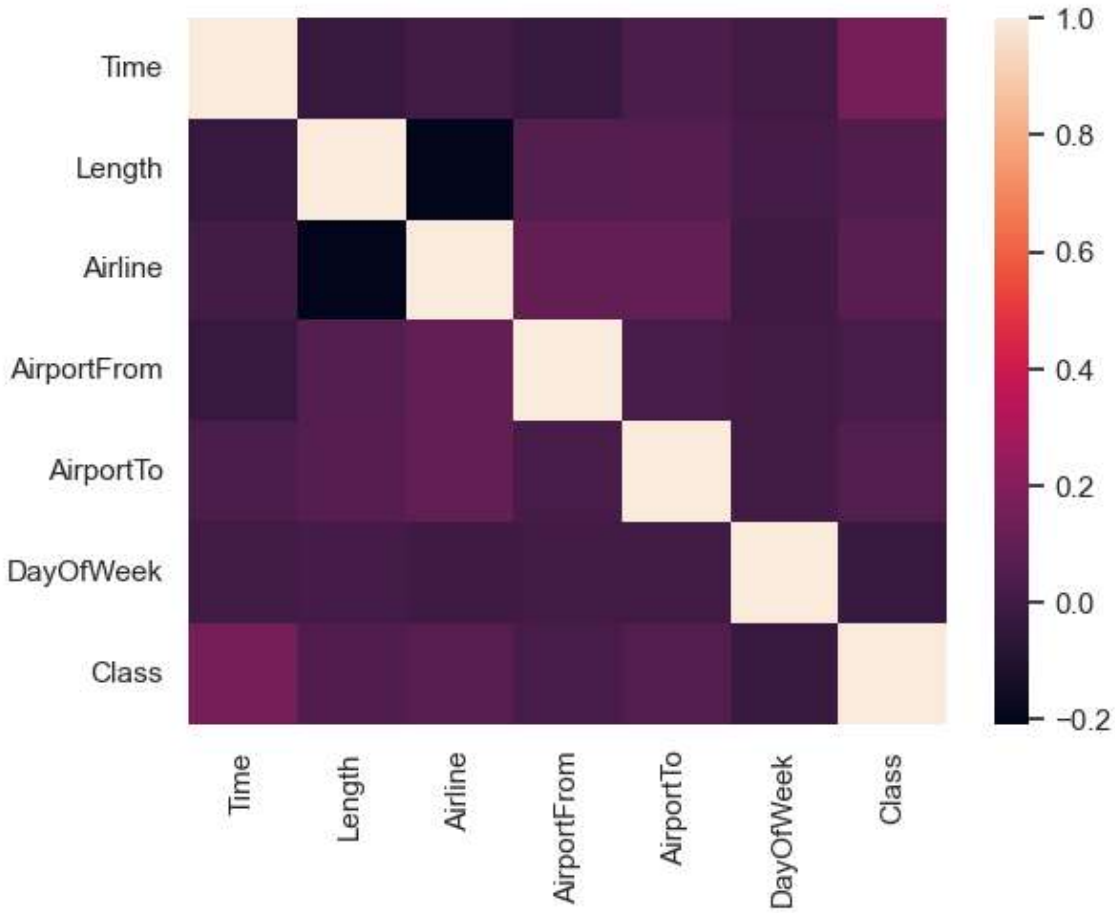
In [17]:
```
#Remove Outlier using Z-Score Method
import scipy.stats as stats
z = np.abs(stats.zscore(df_upsampled))
data_clean = df_upsampled[(z<3).all(axis = 1)]
data_clean.shape
```

Out[17]:
```
(587262, 7)
```

Attribute Correlation

In [18]:
```
sns.heatmap(data_clean.corr(), fmt='.2g')
```

Out[18]: <AxesSubplot:>



In [19]:
```
X = data_clean.drop('Class', axis=1)
y = data_clean['Class']
```

In [20]:
```
#test size 20% and train size 80%
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2,random_state=0
```

Decision Tree Classifier

In [21]:
```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(random_state=0)
dtree.fit(X_train, y_train)
```

Out[21]:
```
DecisionTreeClassifier(random_state=0)
```

In [22]:
```
y_pred = dtree.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

```
Accuracy Score : 68.09 %
```

In [23]:
```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
print('F-1 Score : ',(f1_score(y_test, y_pred)))
print('Precision Score : ',(precision_score(y_test, y_pred)))
print('Recall Score : ',(recall_score(y_test, y_pred)))
```

```
F-1 Score :  0.6744489030173536
Precision Score :  0.686760413902892
Recall Score :  0.6625710336353863
```

Random Forest Classifier

In [34]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(random_state=0)
rfc.fit(X_train, y_train)
```

Out[34]:
```
RandomForestClassifier(random_state=0)
```

In [35]:
```
y_pred = rfc.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

```
Accuracy Score : 69.89 %
```

In [36]:
```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
print('F-1 Score : ',(f1_score(y_test, y_pred)))
print('Precision Score : ',(precision_score(y_test, y_pred)))
print('Recall Score : ',(recall_score(y_test, y_pred)))
```

```
F-1 Score :  0.7072126886366834
Precision Score :  0.6867303302144625
Recall Score :  0.7289544190173893
```

Logistic Regression

In [27]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(random_state=0)
lr.fit(X_train, y_train)
```

Out[27]:
```
LogisticRegression(random_state=0)
```

In [28]:
```
y_pred = lr.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

```
Accuracy Score : 57.52 %
```

In [29]:
```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
print('F-1 Score : ',(f1_score(y_test, y_pred)))
print('Precision Score : ',(precision_score(y_test, y_pred)))
print('Recall Score : ',(recall_score(y_test, y_pred)))
```

```
F-1 Score :  0.572576568858428
Precision Score :  0.5749062381722465
Recall Score :  0.5702657041929043
```

AdaBoost Classifier

In [30]:
```
from sklearn.ensemble import AdaBoostClassifier
ada = AdaBoostClassifier(random_state=0)
ada.fit(X_train, y_train)
```

Out[30]:
```
AdaBoostClassifier(random_state=0)
```

In [31]:
```
y_pred = ada.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

```
Accuracy Score : 62.38 %
```

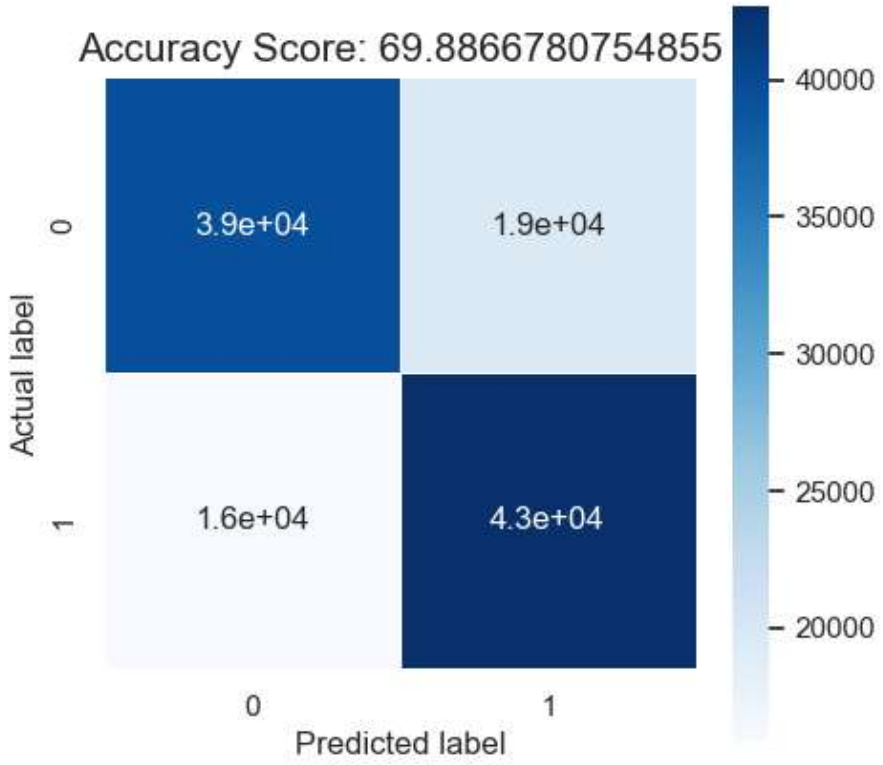
In [32]:
```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
print('F-1 Score : ',(f1_score(y_test, y_pred)))
print('Precision Score : ',(precision_score(y_test, y_pred)))
print('Recall Score : ',(recall_score(y_test, y_pred)))
```

```
F-1 Score :  0.6156910151935504
Precision Score :  0.6277844778660613
Recall Score :  0.6040546767009676
```

Visualize Random Forest Classifier

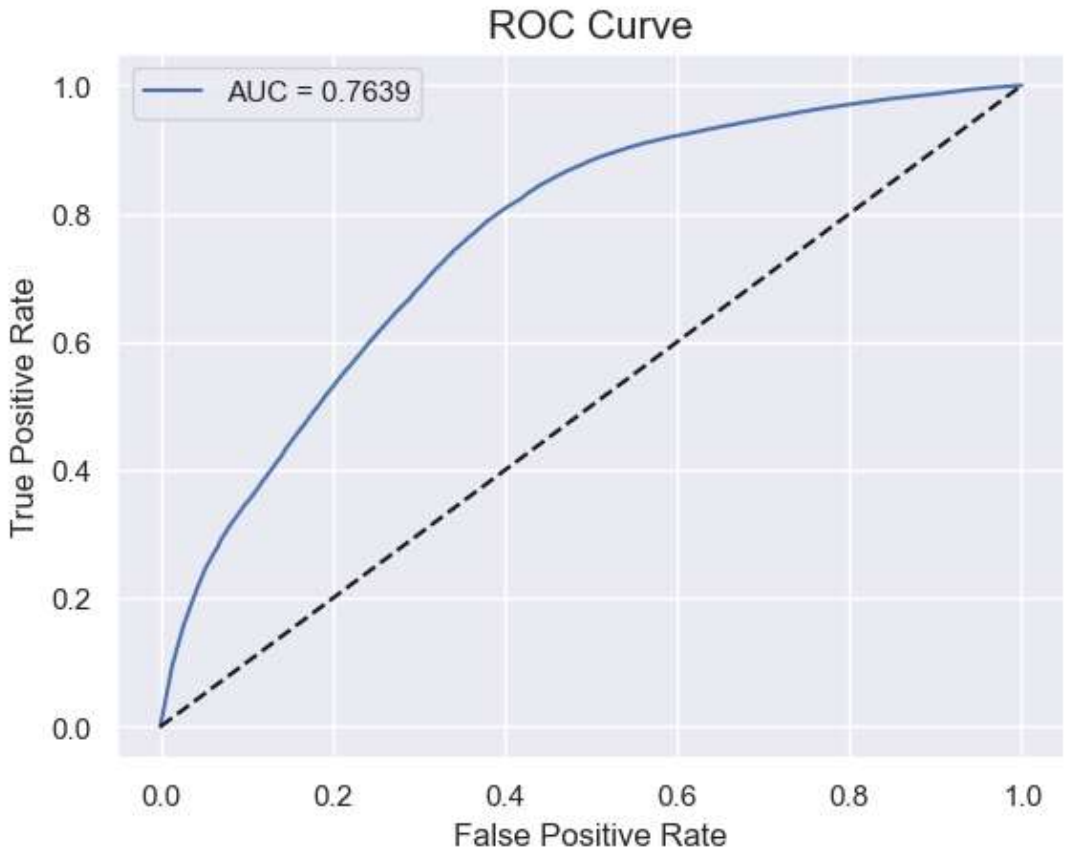
In [37]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=.5, annot=True,square = True,  cmap = 'Blues')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Accuracy Score: {0}'.format(rfc.score(X_test, y_test)*100)
plt.title(all_sample_title, size = 15)
```

Out[37]:
```
Text(0.5, 1.0, 'Accuracy Score: 69.8866780754855')
```



In [38]:
```
from sklearn.metrics import roc_curve, roc_auc_score
y_pred_proba = rfc.predict_proba(X_test)[:][:,1]
df_actual_predicted = pd.concat([pd.DataFrame(np.array(y_test), columns=['y_actual'])
df_actual_predicted.index = y_test.index
fpr, tpr, tr = roc_curve(df_actual_predicted['y_actual'], df_actual_predicted['y_pred
auc = roc_auc_score(df_actual_predicted['y_actual'], df_actual_predicted['y_pred_proba
plt.plot(fpr, tpr, label='AUC = %0.4f' %auc)
plt.plot(fpr, fpr, linestyle = '--', color='k')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve', size = 15)
plt.legend()
```

Out[38]: <matplotlib.legend.Legend at 0x26d567e2130>



In [39]:
```
#Feature Importance
imp_df = pd.DataFrame({
    "Feature Name": X_train.columns,
    "Importance": rfc.feature_importances_
})
fi = imp_df.sort_values(by="Importance", ascending=False)
fi
```

Out[39]:

	Feature Name	Importance
0	Time	0.264537
5	DayOfWeek	0.239242
1	Length	0.177447
3	AirportFrom	0.110214
2	Airline	0.108437
4	AirportTo	0.100123

In [41]:
```
fi2 = fi.head(10)
plt.figure(figsize=(10,8))
sns.barplot(data=fi2, x='Importance', y='Feature Name')
plt.title('Top Feature Importance Each Attributes (Random Forest)', fontsize=18)
plt.xlabel ('Importance', fontsize=16)
plt.ylabel ('Feature Name', fontsize=16)
plt.show()
```