# -\*- coding: utf-8 -\*-

"""

Spyder Editor

This is a temporary script file.

"""

###############################################################################

###############################################################################

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DATA PREPARATION AND CLEANING\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

###############################################################################

###############################################################################

import os

import pandas as pd

import numpy as np

df = pd.read\_excel('2010to2018.xlsx')

#print the column names

print (df.columns)

df.head()

df.tail()

df.isna()

#delete columns NA or invalid data

df=df.dropna(how='all', axis=1)

#df2=df1.drop(df1.index[33266:35969], inplace=False)

#new columns names

dffinal = df.rename(columns={'Unnamed: 0': 'Date', 'Unnamed: 1': 'flightN','Unnamed: 3': 'AircraftType','Unnamed: 4': 'AircraftReg','Unnamed: 5':'OperationType','Unnamed: 6':'Departure' })

dffinal = dffinal.rename(columns={'Unnamed: 7': 'Arrival', 'Unnamed: 8': 'EstimTD','Unnamed: 9': 'ActualTD','Unnamed: 10': 'EstimTA','Unnamed: 11':'ActualTA','Unnamed: 12':'DepartDiff' })

dffinal = dffinal.rename(columns={'Unnamed: 13': 'ArrivalDiff', 'Unnamed: 15': 'AirbornH','Unnamed: 16': 'BlockH','Unnamed: 17': 'Passengers','Flight Statistics Report': 'Arrival'})

#Detect missing values in an Index and nulls

dffinal.isna().sum()

dffinal.isnull().sum()

# dropping NA value observations

dffinal = dffinal.dropna(how='any',axis=0)

#delete column names by default on

dffinal=dffinal.drop(df.index[6], inplace=False)

#split and convertion to datetime Date

dffinal['Date'] = pd.to\_datetime(dffinal['Date'], format='%d%b%y', utc=True)

dffinal['Year'] = pd.DatetimeIndex(dffinal['Date']).year

dffinal['Month'] = pd.DatetimeIndex(dffinal['Date']).month

dffinal['Day'] = pd.DatetimeIndex(dffinal['Date']).day

dffinal['Weekday'] = dffinal['Date'].dt.day\_name

dffinal['Weekday\_num'] = dffinal['Date'].dt.dayofweek

#split STD

dffinal['HEstimTD'] = pd.DatetimeIndex(dffinal['EstimTD']).hour

dffinal['mtsEstimTD'] = pd.DatetimeIndex(dffinal['EstimTD']).minute

#split ATD

dffinal['HActualTD'] = pd.DatetimeIndex(dffinal['ActualTD']).hour

dffinal['mtsActualTD'] = pd.DatetimeIndex(dffinal['ActualTD']).minute

#split STA

dffinal['EstimTA'] = dffinal['EstimTA'].str.replace('24:','00:')

dffinal['HEstimTA'] = pd.DatetimeIndex(dffinal['EstimTA']).hour

dffinal['mtsEstimTA'] = pd.DatetimeIndex(dffinal['EstimTA']).minute

#split ATA

dffinal['ActualTA'] = dffinal['ActualTA'].str.replace('24:','00:')

#dffinal['ActualTA'] = pd.to\_datetime(dffinal['ActualTA'], format='%H:%m', utc=True)

dffinal['HActualTA'] = pd.DatetimeIndex(dffinal['ActualTA']).hour

dffinal['mtsActualTA'] = pd.DatetimeIndex(dffinal['ActualTA']).minute

dffinal.head()

#sort date

dffinal=dffinal.sort\_values(by=['Date'])

#evaluation of negative of positive sign in string DepartDiff

dffinal['stateflightDepartDiff']=dffinal['DepartDiff'].str.contains('-')

#remane true and false per Early or delayed flight

dffinal['stateflightDepartDiff']= np.where (dffinal['stateflightDepartDiff'], 'Early', 'Delayed')

#split minutes and hours DepartDiff

dffinal['HDepartDiff'],dffinal['mtsDepartDiff']= dffinal['DepartDiff'].str.split(":", n = 1).str

#convertion to numeric

dffinal['HDepartDiff']= pd.to\_numeric(dffinal['HDepartDiff'], errors='raise', downcast=None)

dffinal['mtsDepartDiff']= pd.to\_numeric(dffinal['mtsDepartDiff'], errors='raise', downcast=None)

dffinal['Passengers']= pd.to\_numeric(dffinal['Passengers'], errors='raise', downcast=None)

#convertion to minutes

dffinal.loc[:,'HDepartDiff'] \*= 60

#sum mts

dffinal['mtstotaisDepartDiff']=dffinal['HDepartDiff']+dffinal['mtsDepartDiff']

dffinal['mtstotaisDepartDiff']=dffinal['mtstotaisDepartDiff'].abs()

#evaluation of negative of positive sign in string ArrivalDiff

dffinal['stateflightArrivalDiff']=dffinal['ArrivalDiff'].str.contains('-')

#remane true and false per Early or delayed flight

dffinal['stateflightArrivalDiff']= np.where (dffinal['stateflightArrivalDiff'], 'Early', 'Delayed')

#split minutes and hours ArrivalDiff

dffinal['HArrivalDiff'],dffinal['mtsArrivalDiff']= dffinal['ArrivalDiff'].str.split(":", n = 1).str

#convertion to numeric

dffinal['HArrivalDiff']= pd.to\_numeric(dffinal['HArrivalDiff'], errors='raise', downcast=None)

dffinal['mtsArrivalDiff']= pd.to\_numeric(dffinal['mtsArrivalDiff'], errors='raise', downcast=None)

#convertion to minutes

dffinal.loc[:,'HArrivalDiff'] \*= 60

#sum mts

dffinal['mtstotaisArrivalDiff']=dffinal['HArrivalDiff']+dffinal['mtsArrivalDiff']

dffinal['mtstotaisArrivalDiff']=dffinal['mtstotaisArrivalDiff'].abs()

#Airborn

dffinal['HAirborn'],dffinal['mtsAirborn']= dffinal['AirbornH'].str.split(":", n = 1).str

dffinal['HAirborn']= pd.to\_numeric(dffinal['HAirborn'], errors='raise', downcast=None)

dffinal['mtsAirborn']= pd.to\_numeric(dffinal['mtsAirborn'], errors='raise', downcast=None)

dffinal.loc[:,'HAirborn'] \*= 60

dffinal['mtstotaisAirborn']=dffinal['HAirborn']+dffinal['mtsAirborn']

# Blockhours

dffinal['HBlockH'],dffinal['mtsBlockH']= dffinal['BlockH'].str.split(":", n = 1).str

dffinal['HBlockH']= pd.to\_numeric(dffinal['HBlockH'], errors='raise', downcast=None)

dffinal['mtsBlockH']= pd.to\_numeric(dffinal['mtsBlockH'], errors='raise', downcast=None)

dffinal.loc[:,'HBlockH'] \*= 60

dffinal['mtstotaisBlockH']=dffinal['HBlockH']+dffinal['mtsBlockH']

dffinal['CorrectOperationType']=dffinal['OperationType'].str.replace('T', 'Test').replace('C', 'Charter').replace('J', 'ACMI').replace('K', 'Train').replace('N', 'Private').replace('G', 'ACMI').replace('W', 'ACMI').replace('P', 'Position')

dffinal['Airline'] = 0

dffinal['Airline'] = dffinal.flightN.str.extract(r'(\w+?)(\d+)', expand=True)

#Anonymization

dffinal['Airline\_anon'] = dffinal['Airline'].astype('category').cat.codes

###############################################################################

###############################################################################

#\*\*\*\*\*\*\*\*Exploratory Data Analysis\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

###############################################################################

###############################################################################

import matplotlib.pyplot as plt

import seaborn as sns

#count/sandard deviation,mean,min, max, percentiles per categories

statsorig = dffinal.describe(include='all')

print(statsorig)

# revelate the most important categorical

#bar chart Type of Operation

count\_operation = dffinal["CorrectOperationType"].value\_counts()

x\_axis=count\_operation.index

y\_axis=dffinal["CorrectOperationType"].value\_counts()

g=sns.barplot(x\_axis, y\_axis, color="red")

ax=g

plt.title('Operations Types')

plt.xlabel('operation')

plt.ylabel('number of operations')

#annotate axis = seaborn axis

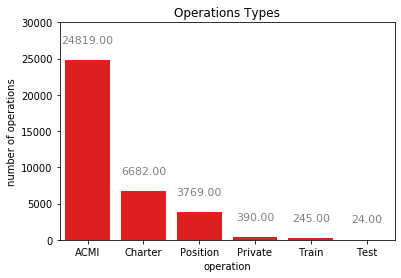
for p in ax.patches:

ax.annotate("%.2f" % p.get\_height(), (p.get\_x() + p.get\_width() / 2., p.get\_height()),

ha='center', va='center', fontsize=11, color='gray', xytext=(0, 20),

textcoords='offset points')

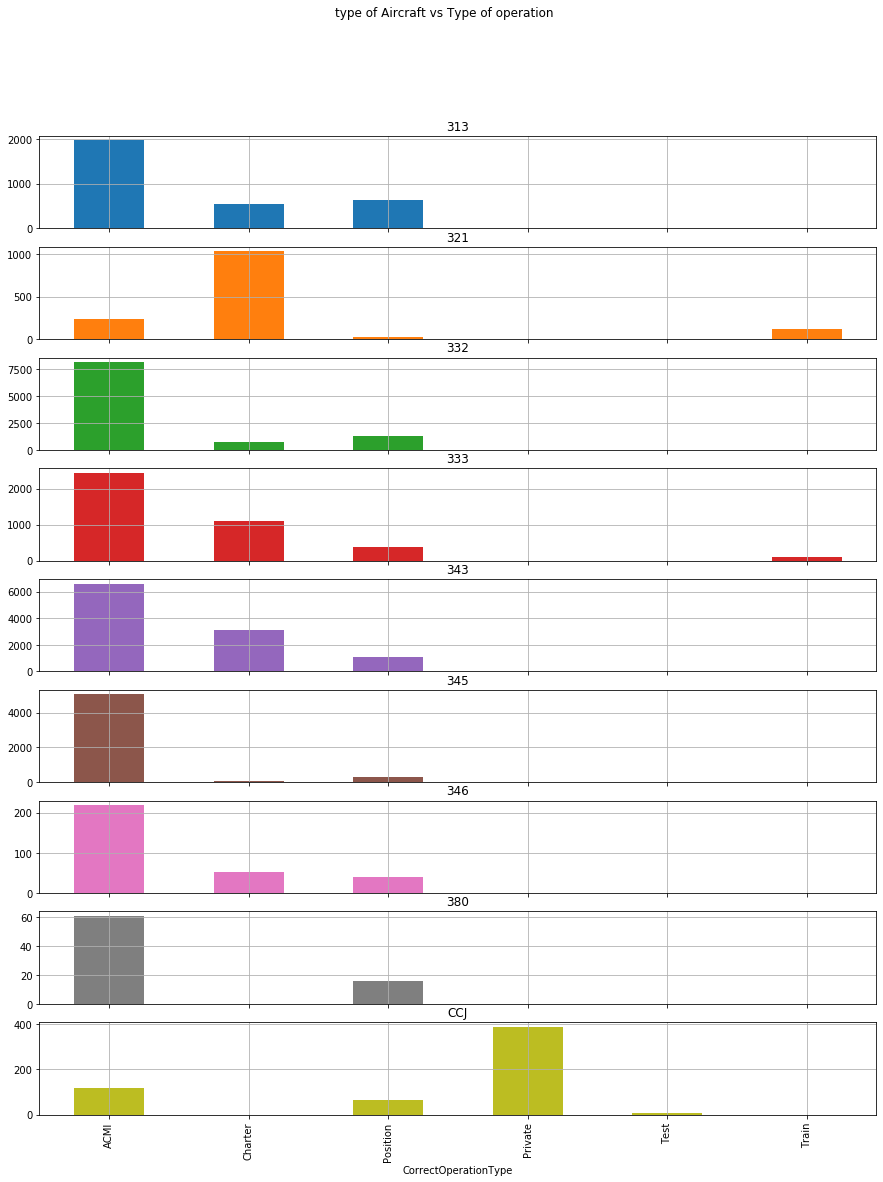
\_ = g.set\_ylim(0,30000)



# Subplot with type of Aircraft vs type of operation

dfsubplot = dffinal.groupby(['CorrectOperationType','AircraftType']).size().unstack(fill\_value=0)

dfsubplot.plot.bar(subplots=True,title='type of Aircraft vs Type of operation',figsize=(15, 18),sharex=True,grid=True,legend= False)

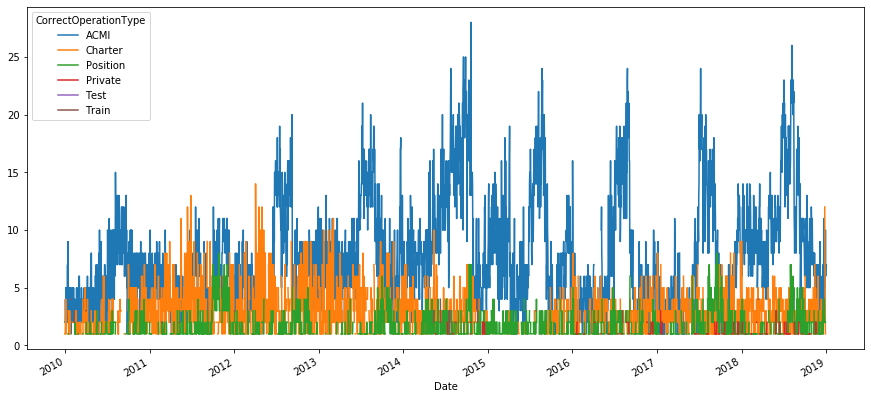


#plot between type of operation and type of aircraft per date

#plot data

fig, ax = plt.subplots(figsize=(15,7))

dffinal.groupby(['Date','CorrectOperationType']).count()['AircraftType'].unstack().plot(ax=ax)



dfplot2=dffinal[dffinal['AircraftType'] == '343']

dfplot3=dfplot2[['Date','AircraftType','CorrectOperationType']]

fig, ax = plt.subplots(figsize=(15,7))

dfplot3.groupby(['Date','CorrectOperationType']).count()['AircraftType'].unstack().plot(grid=True,ax=ax, title='A343')

