Analysis of Flights and Delay of flights from Newyork Airports

---

Table of Contents

- 1. Problem Statement
  - 1.1 Introduction
  - 1.2 Data source and data set
  - 1.3 Data Description or Meta Data
- 2. Load the packages and data
  - 2.1 Import required packages
  - 2.2 Loading of Dataset
- 3. Data profiling
  - 3.1 dtypes
  - 3.2 head
  - 3.3 describe
  - 3.4 sample
  - 3.5 Pandas profiling
- 4. Data normalization
  - 4.1 Standardize column headers to lower case
  - 4.2 Handling Missing Data
  - 4.3 Insert "date" columns using Year, month and date from flights dataset
  - 4.4 Name of Months in Categorical format
  - 4.5 Insert new column and Encode Airlines two digit code
- 5. Identify patterns in the data
  - 5.1 What is total number of scheduled departure by different carrier in Year 2013 ?
  - 5.2 What is count of Airlines flew from Newyork in year 2013 with annotation ?
  - 5.3 What is total scheduled departure from unique origin ?
  - 5.4 What is % of flight opearation from 3 diffierent origin ?

    - 5.5 Distribution pattern of Airlines from 3 airports (EWR/JFK/LGA) of Newyork.
- 6. Analysis of Destination
    - 6.1 How many unique destination and flight network from 3 different origin ?
    - 6.2 What is top 10 destination from Newyork Airport ?
    - 6.3 What is Monthwise top 10 destinations flight from Newyork ?
    - 6.4 Top 10 destination using hue parameters.
    - 6.5 What is maximum numbers of flights headed to unique destination from origin ?
    - 6.6 As Chicago has maximum number of flights so what is total number of unique airlines headed towards chicago ?
    - 6.7 Which aircraft is opearting for chicago and what are their tailnumber ?
- 7. Analysis of Delay of Flights
    - 7.1 What is average monthly departure delay by diffierent carriers ?
    - 7.2 Pictoral representation of average monthly departure delay
    - 7.3 What are arrival and departure delay by carrier in year 2013 ?
    - 7.4 Pictorial average arrival delayed and departure delayed by carrier in year 2013 .
    - 7.5 Relation between departure delay and arrival delay of airlines ?
    - 7.6 Average departure delay by carrier with origin .
- 8. Analysis Ontime departure and arrival of Flights
    - 8.1 Departure delay status classification .
    - 8.2 Arrival delay status classification.
    - 8.3 Average arrival delay by carrier.
    - 8.4 Average departure delay by carrier .
- 9. Performance / speed analysis of Aircrafts
    - 9.1 Ratio of scheduled flights and cancelled flights
    - 9.2 In which months have highest average departure delay.
    - 9.3 In which months have highest number of flight delay
    - 9.4 Top 5 destination where flight arrived earlier than scheduled arrival.
    - 9.5 The Best airport in term of early departure from NYC.
    - 9.6 Graphical analysis of average monthly delay of aircraft and number of minute delayed by months
    - 9.7 Other explanation like speed ananlysis / heatmap for correlation
- 10. Conclusion

1. Problem Statement !

"This dataset is from a 2013 survey that how often flight is operated and delayed from Newyork three airports (Newark(EWR)/ NewYork (JFK) and Laguardia (LGA)"

1.1. Introduction

This Exploratory Data Analysis is to practice Python skills learned till now on a structured data set including loading, inspecting, wrangling, exploring, and drawing conclusions from data. The notebook has observations with each step in order to explain thoroughly how to approach the data set. Based on the observation some questions also are answered in the notebook for the reference though not all of them are explored in the analysis.

1.2. Data source and dataset

How was it collected?

- **Name**: "Delay and of flight analysis of USA airlines origin from all three airport of NewYork "
- **Year**: 2013
- **Description**: "There are three airports of Newyork and flight of Various airlines are operating so need to analyse that how often flight is delayed and what is frequency of delay ".

Data Source: [https://github.com/insaid2018/Term-1/blob/master/Data/Projects/NYC_Flight_Data.rar (https://github.com/insaid2018/Term-1/blob/master/Data/Projects/NYC_Flight_Data.rar)](https://github.com/insaid2018/Term-1/blob/master/Data/Projects/NYC_Flight_Data.rar)

1.3 Description/metadata of the Dataset

- Dataset have 19 columns which is as below

Dataset Details

1. year - Year in which flight was operated.
2. month - Month of the year in which Flight was operated.
3. day - Day (number of Flight)
4. dep_time - Departure Time
5. sched_dep_time - Scheduled Departure Time
6. dep_delay - Departure Delay (in Minutes)
7. arr_time - Arrival Time
8. sched_arr_time - Scheduled Arrival Time
9. arr_delay - Arrival Delay (in Minutes)
10. carrier- Carrier Details of Flight
11. flight - Flight Name
12. tailnum - Tailnumber of Flight
13. origin - Origin of Flight
14. dest - Destination Airport Name
15. air_time - Air Time (Time in Air - travel Time (in minutes))
16. distance - Distance (assuming in miles?)
17. hour - Hour
18. minute - minute
19. time_hour - time hour

2. Load the packages and data

2.1 Import required packages

In [2]:

```
# ! pip install https://github.com/pandas-profiling/pandas-profiling/archive/master.zip
```

In [3]:

```
#-----------------------------------------------------------------------------------------
# import numpy
import numpy as np                                                    # Impport numerical
np.set_printoptions(precision = 4)                                    # To display value
#-----------------------------------------------------------------------------------------

import pandas as pd                                                  # Import Pandas for
from pandas_profiling import ProfileReport                            # Import pandas pro
pd.set_option('display.max_columns', None)                            # Unfolding hidden
pd.set_option('display.max_colwidth', None)                           # Unfolding the max
pd.set_option('display.max_rows', None)                               # Unfolding hidden
pd.set_option('mode.chained_assignment', None)                        # Removing restrict
pd.set_option('display.float_format', lambda x: '%.f' % x)           # To suppress scient
#-----------------------------------------------------------------------------------------

from matplotlib import pyplot as plt ; plt.rcdefaults()               # Import Matplot j
plt.style.use('seaborn-whitegrid')                                    # to apply seaborn
%matplotlib inline
#-----------------------------------------------------------------------------------------

import seaborn as sns                                                  # Import seaborn j
sns.set(style='whitegrid',font_scale=1.3, color_codes= True)           # To adjust seabor

#-----------------------------------------------------------------------------------------
import plotly.express as px
#-----------------------------------------------------------------------------------------
import datetime                                                         # import datetime
#-----------------------------------------------------------------------------------------
import networkx as nx
#-----------------------------------------------------------------------------------------
import warnings                                              # To Supress all the warning
warnings.filterwarnings('ignore')

```

2.1 Loading of Dataset

In [4]:

```
flights = pd.read_csv('C:/Users/ITQLAP/Desktop/Data/project/NYC_Flight_Data.csv')
print('Shape of the dataset : ',flights.shape)

# Incase use excel
# flights = pd.read_excel("C:/Users/ITQLAP/Desktop/Data/project/NYC_Flight_Data.xls",'j
```

```
Shape of the dataset :  (336776, 19)
```

3. Data Profiling

3.1 Get the data information and data type for further action

In [5]:

```
flights.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 336776 entries, 0 to 336775
Data columns (total 19 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   year            336776 non-null  int64  
 1   month           336776 non-null  int64  
 2   day             336776 non-null  int64  
 3   dep_time        328521 non-null  float64
 4   sched_dep_time  336776 non-null  int64  
 5   dep_delay       328521 non-null  float64
 6   arr_time        328063 non-null  float64
 7   sched_arr_time  336776 non-null  int64  
 8   arr_delay       327346 non-null  float64
 9   carrier         336776 non-null  object 
 10  flight          336776 non-null  int64  
 11  tailnum         334264 non-null  object 
 12  origin          336776 non-null  object 
 13  dest            336776 non-null  object 
 14  air_time        327346 non-null  float64
 15  distance        336776 non-null  int64  
 16  hour            336776 non-null  int64  
 17  minute          336776 non-null  int64  
 18  time_hour       336776 non-null  object 
dtypes: float64(5), int64(9), object(5)
memory usage: 48.8+ MB
```

3.2 Getting head and tail of dataset.

In [6]:

```
flights.head()  # Show head of the Data Set
```

Out[6]:

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_dela
0	2013	1	1	517	515	2	830	819	1
1	2013	1	1	533	529	4	850	830	2
2	2013	1	1	542	540	2	923	850	3
3	2013	1	1	544	545	-1	1004	1022	-1
4	2013	1	1	554	600	-6	812	837	-2

In [7]:

```
flights.tail(5)
```

Out[7]:

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr
336771	2013	9	30	nan	1455	nan	nan	1634	
336772	2013	9	30	nan	2200	nan	nan	2312	
336773	2013	9	30	nan	1210	nan	nan	1330	
336774	2013	9	30	nan	1159	nan	nan	1344	
336775	2013	9	30	nan	840	nan	nan	1020	

3.3 Get the description of data set using describe.

- In this section we will get **information about the data** and see some observations.

In [8]:

```
# flights.describe(include = 'all')
flights.describe()
```

Out[8]:

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time
count	336776	336776	336776	328521	336776	328521	328063	336776
mean	2013	7	16	1349	1344	13	1502	1536
std	0	3	9	488	467	40	533	497
min	2013	1	1	1	106	-43	1	1
25%	2013	4	8	907	906	-5	1104	1124
50%	2013	7	16	1401	1359	-2	1535	1556
75%	2013	10	23	1744	1729	11	1940	1945
max	2013	12	31	2400	2359	1301	2400	2359

3.4 Taking sample of data to know about data set.

In [9]:

```
flights.sample(6)
```

Out[9]:

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr
88887	2013	12	6	2038	2010	28	2213	2130	
304213	2013	8	26	1500	1456	4	1727	1747	
264326	2013	7	15	2055	2050	5	34	42	
136781	2013	3	1	1519	1520	-1	1854	1842	
65268	2013	11	11	1000	1000	0	1242	1245	
61319	2013	11	6	2023	2029	-6	2155	2216	

3.5 Pandas Profiling

In [10]:

```
profile = ProfileReport(flights, title ='Pandas Profiling Report', html = {'style':{'fu
profile.to_notebook_iframe()
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

Overview

Dataset statistics

Number of variables	19
Number of observations	336776
Missing cells	46595
Missing cells (%)	0.7%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	48.8 MiB
Average record size in memory	152.0 B

Variable types

Categorical	6
Numeric	13

Warnings

`year` has constant value "2013"	**Constant**
`tailnum` has a high cardinality: 4043 distinct values	**High cardinality**
`dest` has a high cardinality: 105 distinct values	**High cardinality**
`time_hour` has a high cardinality: 6936 distinct values	**High cardinality**

4. Data Normalisation

4.1 Standardize column headers to lower case.

In [11]:

```
type(flights.columns)
```

Out[11]:

```
pandas.core.indexes.base.Index
```

In [12]:

```
flights.columns
```

Out[12]:

```
Index(['year', 'month', 'day', 'dep_time', 'sched_dep_time', 'dep_delay',
       'arr_time', 'sched_arr_time', 'arr_delay', 'carrier', 'flight',
       'tailnum', 'origin', 'dest', 'air_time', 'distance', 'hour', 'minut
e',
       'time_hour'],
      dtype='object')
```

Observation

- As all columns header are already in lower case hence no action is required.

4.2 Handling Missing Data

In [13]:

```
flights.isna().sum()        # Total Missing vallue in each column
```

Out[13]:

```
year                 0
month                0
day                  0
dep_time          8255
sched_dep_time       0
dep_delay         8255
arr_time          8713
sched_arr_time       0
arr_delay         9430
carrier              0
flight               0
tailnum           2512
origin               0
dest                 0
air_time          9430
distance             0
hour                 0
minute               0
time_hour            0
dtype: int64
```

Observation

- Droping all NA to get normalise the data

In [14]:

```
flights = flights.dropna()                    # Droping NA to make regularise
flights.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 327346 entries, 0 to 336769
Data columns (total 19 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   year            327346 non-null  int64
 1   month           327346 non-null  int64
 2   day             327346 non-null  int64
 3   dep_time        327346 non-null  float64
 4   sched_dep_time  327346 non-null  int64
 5   dep_delay       327346 non-null  float64
 6   arr_time        327346 non-null  float64
 7   sched_arr_time  327346 non-null  int64
 8   arr_delay       327346 non-null  float64
 9   carrier         327346 non-null  object
 10  flight          327346 non-null  int64
 11  tailnum         327346 non-null  object
 12  origin          327346 non-null  object
 13  dest            327346 non-null  object
 14  air_time        327346 non-null  float64
 15  distance        327346 non-null  int64
 16  hour            327346 non-null  int64
 17  minute          327346 non-null  int64
 18  time_hour       327346 non-null  object
dtypes: float64(5), int64(9), object(5)
memory usage: 49.9+ MB
```

In [15]:

```
flights.isna().sum()
```

Out[15]:

```
year              0
month             0
day               0
dep_time          0
sched_dep_time    0
dep_delay         0
arr_time          0
sched_arr_time    0
arr_delay         0
carrier           0
flight            0
tailnum           0
origin            0
dest              0
air_time          0
distance          0
hour              0
minute            0
time_hour         0
dtype: int64
```

In [16]:

```
flights.describe()
```

Out[16]:

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time
count	327346	327346	327346	327346	327346	327346	327346	327346
mean	2013	7	16	1349	1340	13	1502	1533
std	0	3	9	488	467	40	533	498
min	2013	1	1	1	500	-43	1	1
25%	2013	4	8	907	905	-5	1104	1122
50%	2013	7	16	1400	1355	-2	1535	1554
75%	2013	10	23	1744	1729	11	1940	1944
max	2013	12	31	2400	2359	1301	2400	2359

4.3 Insert "date" columns using Year, month and date from flights dataset

In [17]:

```
# Create date columns using Year, month and date
flights['date'] = pd.to_datetime(flights[['year','month','day']],yearfirst = True)
```

In [18]:

```
flights.head()
```

Out[18]:

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_dela
0	2013	1	1	517	515	2	830	819	1
1	2013	1	1	533	529	4	850	830	2
2	2013	1	1	542	540	2	923	850	3
3	2013	1	1	544	545	-1	1004	1022	-1
4	2013	1	1	554	600	-6	812	837	-2

4.4 Insert new column "months_name" in Categorical format like 'Jan','Feb' etc.**

In [19]:

```
# Months name in the original form Jan,feb
month_dict = {1:'Jan',2:'Feb',3:'Mar',4:'Apr',5:'May',6:'Jun',7:'Jul',8:'Aug',9:'Sep',1
flights['months_name'] = flights['month'].apply(lambda m:month_dict[m])
```

In [20]:

```
flights.tail()
```

Out[20]:

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	ar
336765	2013	9	30	2240	2245	-5	2334	2351	
336766	2013	9	30	2240	2250	-10	2347	7	
336767	2013	9	30	2241	2246	-5	2345	1	
336768	2013	9	30	2307	2255	12	2359	2358	
336769	2013	9	30	2349	2359	-10	325	350	

In [21]:

```
flights.months_name.unique()
```

Out[21]:

```
array(['Jan', 'Oct', 'Nov', 'Dec', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
       'Jul', 'Aug', 'Sep'], dtype=object)
```

4.5 Insert new column 'carrier_name' and Encode Airlines two digit code.

In [22]:

```
carrier_dict = {'UA':'UNITED AIRLINES(UA)', 'B6':'JETBLUE AIRWAYS(B6)', 'EV':'EXPRESSJE
                'AA':'AMERICAN AIRLINES(AA)','MQ':'ENVOY AIR(MQ)','US':'US AIRWAYS(US)'
                'WN':'SOUTHWEST AIRLINES(WN)','VX':'SOUTHWEST AIRLINES(VX)','FL':'AIR L
                'F9':'FRONTIER AIRLINES(F9)','YV':'AVIANCA(YV)','HA':'HAWAIIAN AIRLINES
flights['carrier_name'] =flights['carrier'].apply(lambda c: carrier_dict[c])
flights.carrier_name.unique()
```

Out[22]:

```
array(['UNITED AIRLINES(UA)', 'AMERICAN AIRLINES(AA)',
       'JETBLUE AIRWAYS(B6)', 'DELTA AIR LINES',
       'EXPRESSJET AIRLINES(EV)', 'ENVOY AIR(MQ)', 'US AIRWAYS(US)',
       'SOUTHWEST AIRLINES(WN)', 'SOUTHWEST AIRLINES(VX)', 'AIR LEAP(FL)',
       'ALASKA AIRLINES', 'ENDEAVOR AIR(9E)', 'FRONTIER AIRLINES(F9)',
       'HAWAIIAN AIRLINES(HA)', 'AVIANCA(YV)', 'SKYWEST AIRLINES(OO)'],
      dtype=object)
```

5. Identify the pattern in the Data.

5.1 What is total number of scheduled departure by different carrier in Year 2013 ?

In [23]:

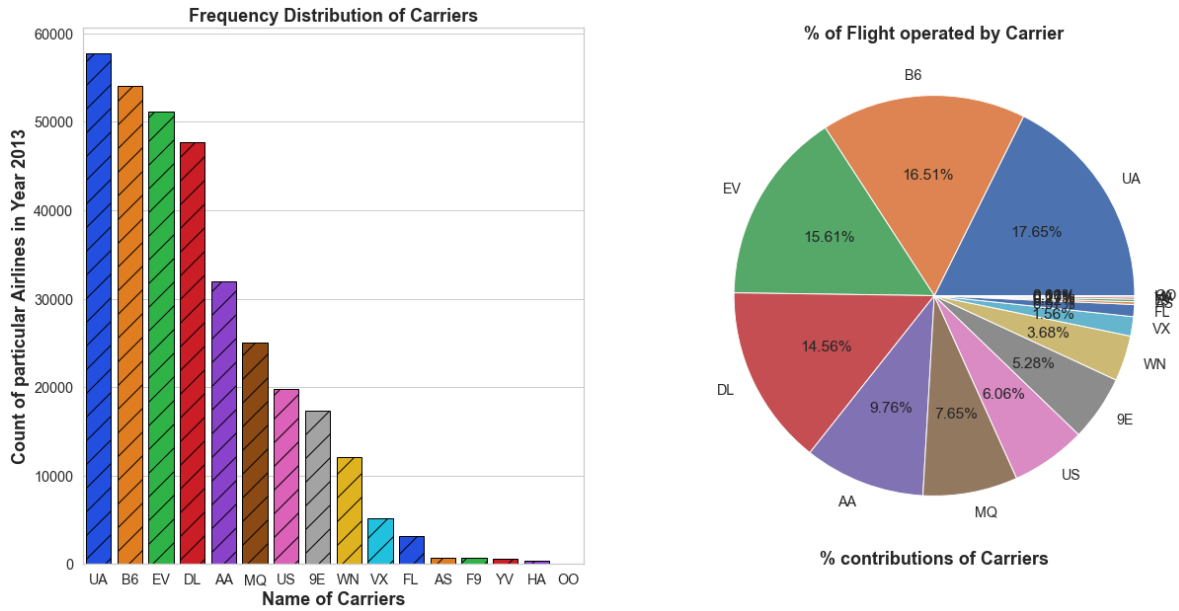
```
# Value counts of carrier
flights.carrier.value_counts()
```

Out[23]:

```
UA    57782
B6    54049
EV    51108
DL    47658
AA    31947
MQ    25037
US    19831
9E    17294
WN    12044
VX     5116
FL     3175
AS      709
F9      681
YV      544
HA      342
OO       29
Name: carrier, dtype: int64
```

In [24]:

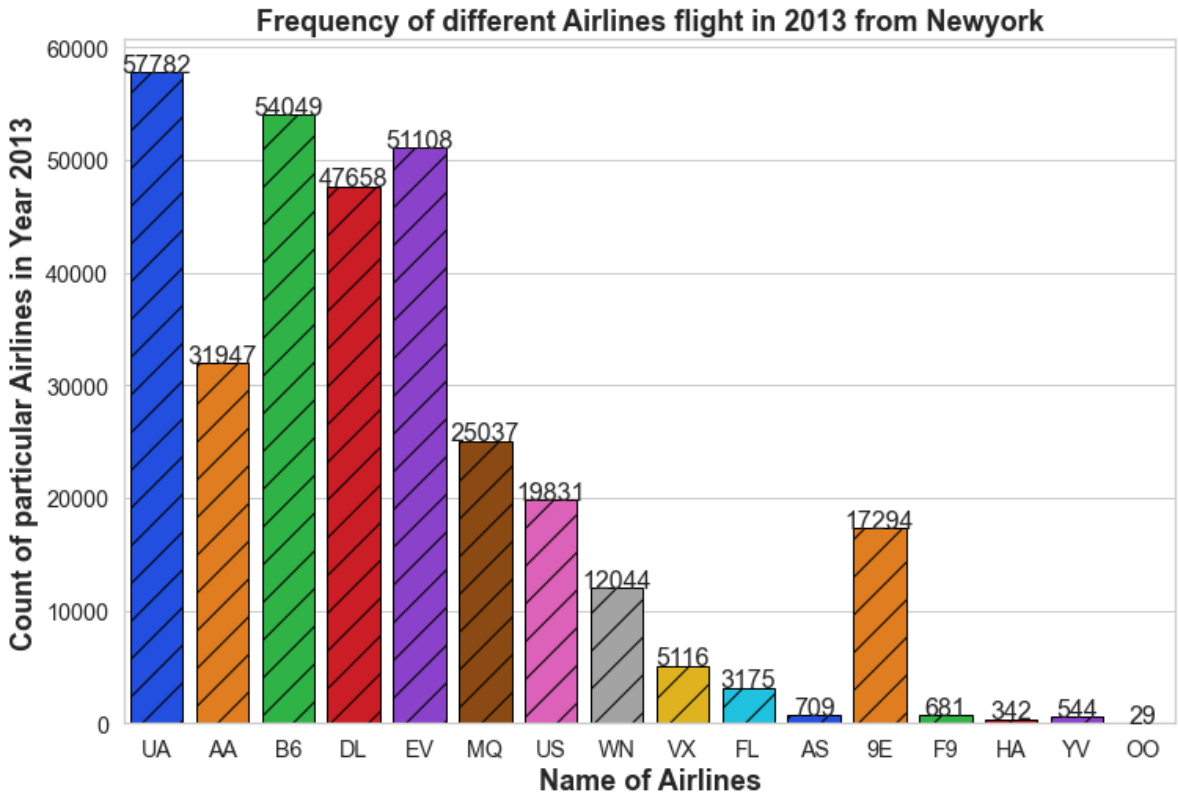
```
# Graphical representation of carrier scheduled flights in number and in Percentage .
f,ax=plt.subplots(1,2,figsize=(20,10))                    # Set subplot size
#-----------------------------------------------------------------------------------------
# sns.countplot('carrier',order = flights['carrier'].value_counts().index, data=flights
sns.countplot('carrier',order = flights['carrier'].value_counts().index,data = flights,
              palette = 'bright', lw = 1, ec = 'black', hatch = '/',ax = ax[0])
ax[0].set_title('Frequency Distribution of Carriers',fontsize = 18, fontweight='bold')
ax[0].set_ylabel('Count of particular Airlines in Year 2013',fontsize = 18, fontweight=
ax[0].set_xlabel('Name of Carriers',fontsize = 18,fontweight = 'bold')

#-----------------------------------------------------------------------------------------

flights['carrier'].value_counts().plot.pie(autopct='%1.2f%%',ax=ax[1],shadow=False)
ax[1].set_title('% of Flight operated by Carrier',fontsize = 18, fontweight='bold')
ax[1].set_ylabel('')
ax[1].set_xlabel('% contributions of Carriers',fontsize = 18,fontweight = 'bold')

#-----------------------------------------------------------------------------------------
plt.show()
```



5.2 What is count of Airlines flew from Newyork in year 2013 with annotation ?

In [25]:

```
# fig size
plt.figure(figsize=(12,8))

# Plot count plot
ax = sns.countplot(x ='carrier',data = flights,palette = 'bright', lw = 1, ec = 'black'

ax.set_title('Frequency of different Airlines flight in 2013 from Newyork',fontsize = 1
ax.set_xlabel('Name of Airlines',fontsize = 18, fontweight='bold')
ax.set_ylabel('Count of particular Airlines in Year 2013',fontsize = 18, fontweight='bo


for rect in ax.patches:
    ax.text (rect.get_x() + rect.get_width() / 2,rect.get_height()+ 20,rect.get_height(

plt.show()
```



5.3 What is total scheduled departure from unique origin ?

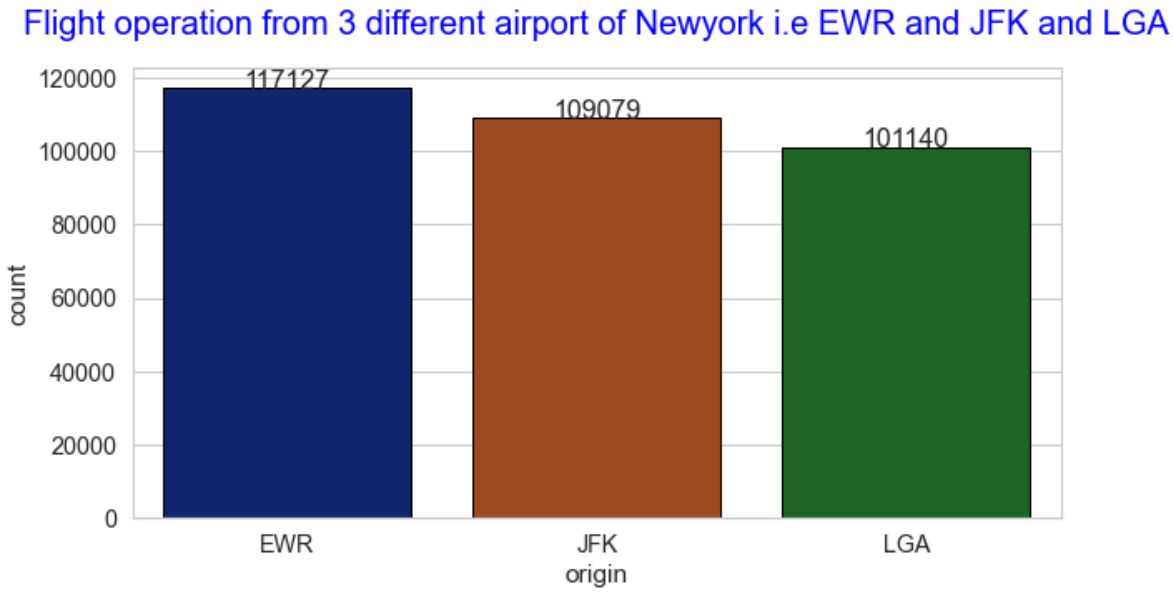
In [26]:

```
# Count of Destination
destination_count = flights.dest.unique()              # Total number of unique destinat
des_count = len(destination_count)                     # Length of count
print('The total number of destination from all three airport of NewYork are : ',des_co

#-----------------------------------------------------------------------------------------
origin_count = flights.origin.value_counts()      # Count of flight from 3 newyork airp
print(origin_count)
```

```
The total number of destination from all three airport of NewYork are :  104
EWR    117127
JFK    109079
LGA    101140
Name: origin, dtype: int64
```

In [27]:

```
plt.figure(figsize = (10,5))
ax = sns.countplot(x ='origin',data = flights,order = flights['origin'].value_counts().
ax.set_title('Flight operation from 3 different airport of Newyork i.e EWR and JFK and

for rect in ax.patches:
    ax.text (rect.get_x() + rect.get_width() / 2,rect.get_height()+ 0.25,rect.get_heigh
```



Explanation :

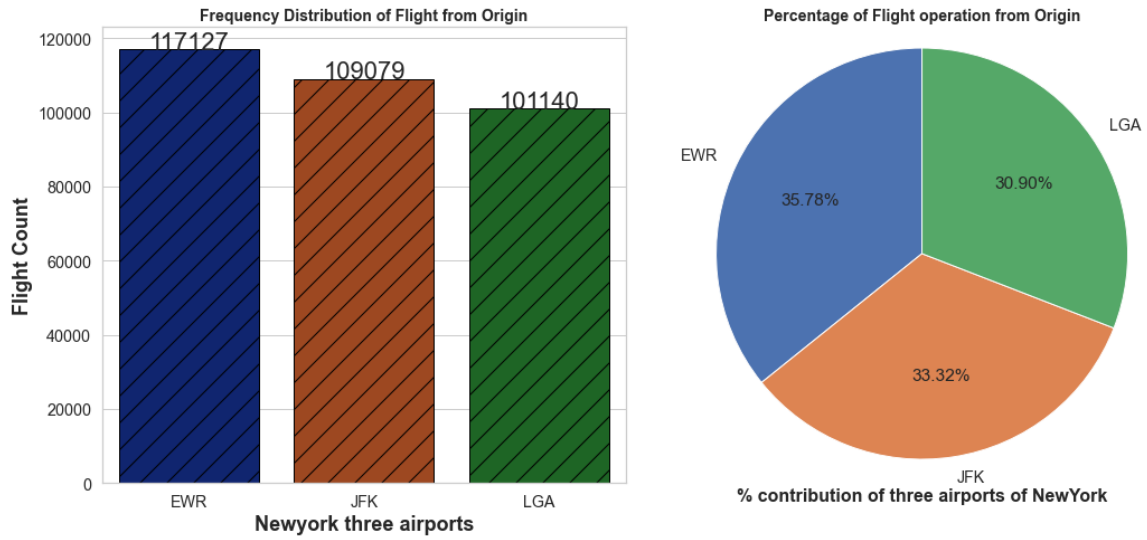
Origin : Three are three airports EWR,JFK and LGA of Newyork and 120835 flights are operating from EWR airport and 111279 flights from JFK airport and 104662 flights from LGA.

Destination : There are 105 destination where flights are operting from EWR,JFK and LGA airports.

5.4 What is % of flight opearation from 3 diffierent origin ?

In [28]:

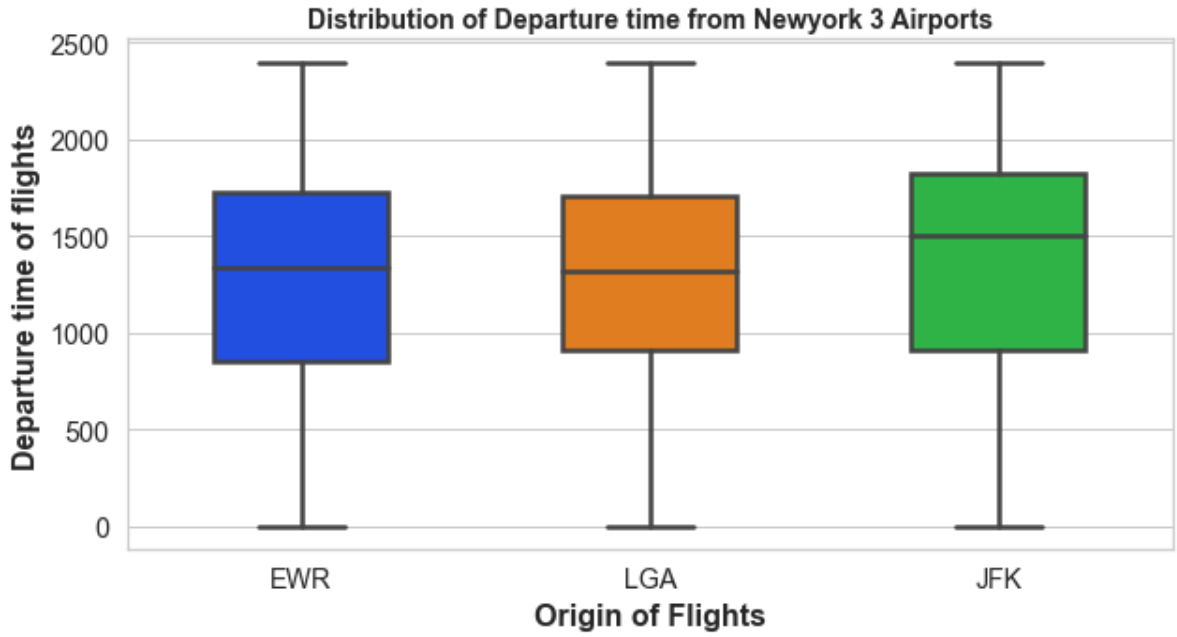
```
f,ax=plt.subplots(1,2,figsize=(15,7))
#-----------------------------------------------------------------------------------------
sns.countplot('origin',order = flights['origin'].value_counts().index, palette = 'dark'
ax[0].set_title('Frequency Distribution of Flight from Origin',fontsize = 14, fontweigh
ax[0].set_ylabel('Flight Count',fontsize = 18, fontweight='bold')
ax[0].set_xlabel('Newyork three airports',fontsize = 18,fontweight = 'bold')

for rect in ax[0].patches:
    ax[0].text (rect.get_x() + rect.get_width() / 2,rect.get_height()+ 0.50,rect.get_he
#-----------------------------------------------------------------------------------------
flights['origin'].value_counts().plot.pie(autopct='%1.2f%%',ax=ax[1],shadow=False,start
ax[1].set_title('Percentage of Flight operation from Origin', fontsize = 14, fontweight
ax[1].set_ylabel('')
ax[1].set_xlabel('% contribution of three airports of NewYork',fontsize = 16,fontweight
ax[1].axis('equal')
plt.tight_layout()
#-----------------------------------------------------------------------------------------
plt.show()
```



5.5 Distribution pattern of Airlines from 3 airports (EWR/JFK/LGA) of Newyork.

In [29]:

```
#Box plot - Two Categorical variables along with Continuous
plt.figure(figsize = (10,5))
ax = sns.boxplot(data=flights,x='origin',y='dep_time',palette ='bright',width = 0.5,lir
ax.set_title('Distribution of Departure time from Newyork 3 Airports',fontsize = 14,for
ax.set_xlabel('Origin of Flights',fontsize = 16,fontweight = 'bold')
ax.set_ylabel('Departure time of flights',fontsize = 16,fontweight = 'bold')
plt.show()
```



Observation:

1. Interquartile range is the highest for JFK airport where 75% of flight are operating till 1800 hrs and middle point of departure is at 1500 hrs.

6. Analysis of Destination

6.1 How many unique destination and flight network from 3 different origin ?

In [30]:

```
# All name of destination which have 3 digit IATA code.
flights['dest'].unique()
```

Out[30]:

```
array(['IAH', 'MIA', 'BQN', 'ATL', 'ORD', 'FLL', 'IAD', 'MCO', 'PBI',
       'TPA', 'LAX', 'SFO', 'DFW', 'BOS', 'LAS', 'MSP', 'DTW', 'RSW',
       'SJU', 'PHX', 'BWI', 'CLT', 'BUF', 'DEN', 'SNA', 'MSY', 'SLC',
       'XNA', 'MKE', 'SEA', 'ROC', 'SYR', 'SRQ', 'RDU', 'CMH', 'JAX',
       'CHS', 'MEM', 'PIT', 'SAN', 'DCA', 'CLE', 'STL', 'MYR', 'JAC',
       'MDW', 'HNL', 'BNA', 'AUS', 'BTV', 'PHL', 'STT', 'EGE', 'AVL',
       'PWM', 'IND', 'SAV', 'CAK', 'HOU', 'LGB', 'DAY', 'ALB', 'BDL',
       'MHT', 'MSN', 'GSO', 'CVG', 'BUR', 'RIC', 'GSP', 'GRR', 'MCI',
       'ORF', 'SAT', 'SDF', 'PDX', 'SJC', 'OMA', 'CRW', 'OAK', 'SMF',
       'TYS', 'PVD', 'DSM', 'PSE', 'TUL', 'BHM', 'OKC', 'CAE', 'HDN',
       'BZN', 'MTJ', 'EYW', 'PSP', 'ACK', 'BGR', 'ABQ', 'ILM', 'MVY',
       'SBN', 'LEX', 'CHO', 'TVC', 'ANC'], dtype=object)
```

In [31]:

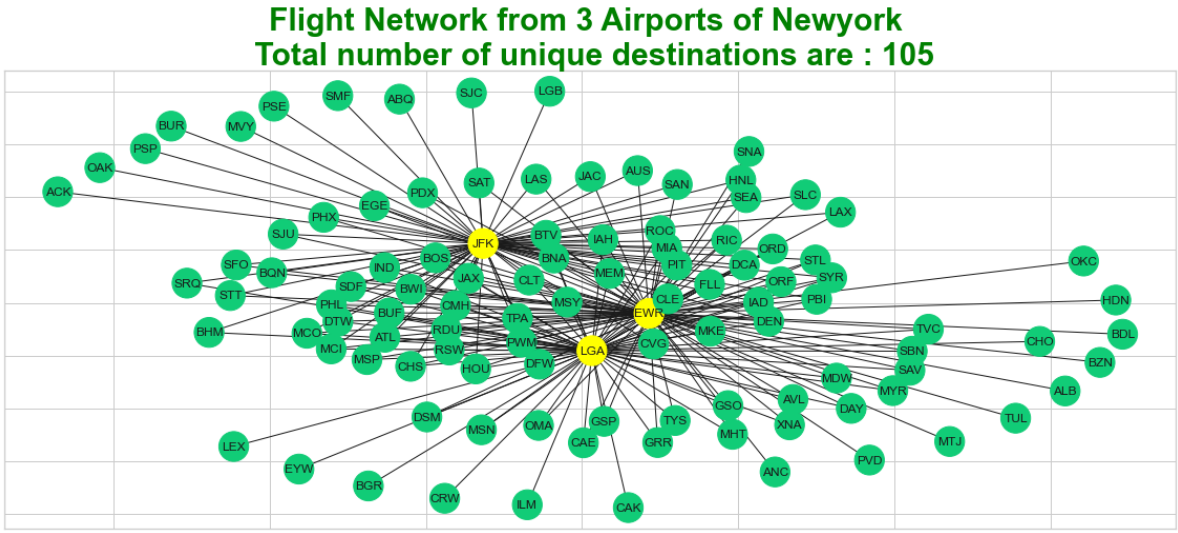
```
# Print count of unique detination
print("Total number of unique destinations are : ", len(flights['dest'].unique()))
```

```
Total number of unique destinations are :  104
```

Route from EWR , JFK and LGA to different destination

In [32]:

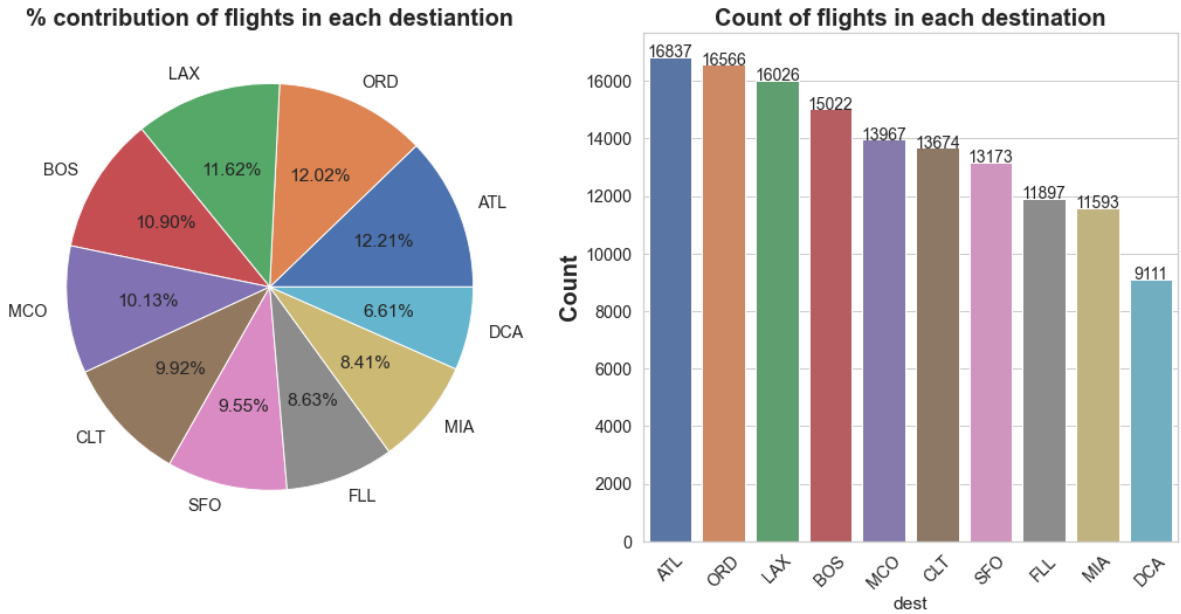
```
plt.figure(figsize=(20,8))
G=nx.Graph()
for index, row in flights.iterrows():
    G.add_edge(row['origin'], row['dest'])

# Give the starting and ending airports a different color
colors = []

for label in G:
    if label in['EWR','JFK','LGA']:
        colors.append('yellow')
    else:
        colors.append('#11cc77')

nx.draw_networkx(G,node_color=colors, node_size=800, with_labels=True )
plt.title(label = 'Flight Network from 3 Airports of Newyork \n Total number of unique
plt.show()
```



6.2 What is top 10 destination from Newyork Airports ?

In [33]:

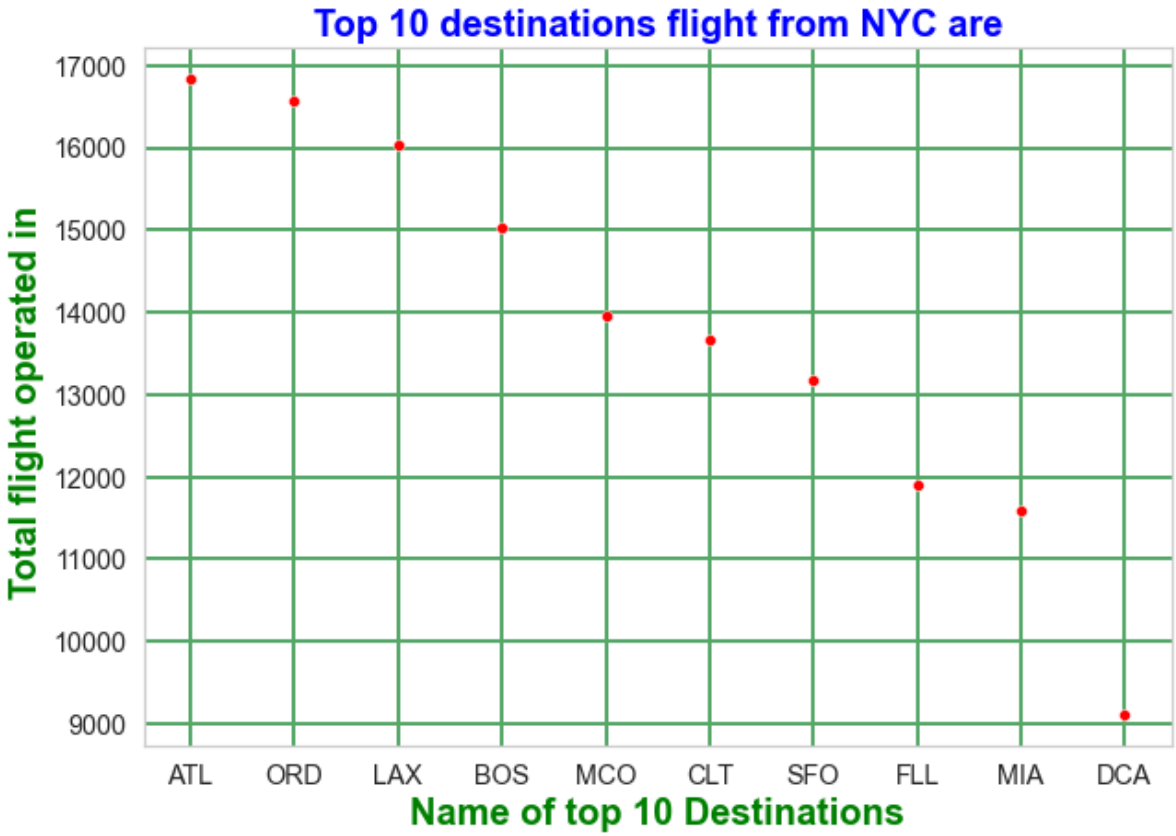
```
f,ax = plt.subplots(1,2,figsize= (18,8))                                    # Set figure size ar

top_10_destination = flights['dest'].value_counts().head(10).sort_values(ascending = Fa
#-----------------------------------------------------------------------------------------
# Use ax[0] for % of flight from origin to 10 different destination
top_10_destination.plot.pie(autopct = '%1.2f%%',ax = ax[0],shadow = False)
ax[0].set_title('% contribution of flights in each destiantion',fontweight = 'bold',for
ax[0].set_ylabel('')

#-----------------------------------------------------------------------------------------
sns.countplot('dest',order = top_10_destination.index,data = flights,ax = ax[1])
ax[1].set_title('Count of flights in each destination',fontweight = 'bold',fontsize = 2
plt.xticks(rotation=45)
ax[1].set_ylabel('Count',fontweight = 'bold',fontsize = 20)

for rect in ax[1].patches:
    ax[1].text (rect.get_x() + rect.get_width() / 2,rect.get_height()+ 0.25,rect.get_he

plt.show()
```



6.3 What is Monthwise top 10 destinations flight from Newyork ?

In [34]:

```
plt.figure(figsize= (10,7))
plt.grid(True, color='g', linewidth=2)

# Groupby destination and aggregate count of months
dest_count = flights.groupby(['dest'],as_index = False).agg({'month':'count'})

# Maximum distance count using sort_values in descending order
max_dest_count = dest_count.sort_values(['month'],ascending = False)

# Head of top 10 Destination
top_dest_flight = max_dest_count.head(10)

# Draw scatter plot where marker color is red
sns.scatterplot(top_dest_flight.dest,top_dest_flight.month,color='red')

# Styling like title , xlabel , ylabel
plt.title('Top 10 destinations flight from NYC are',fontsize = 20,fontweight = 'bold',
plt.xlabel('Name of top 10 Destinations',color = 'green', fontsize = 20, fontweight = '
plt.ylabel('Total flight operated in ',color = 'green', fontsize = 20, fontweight = 'bo

# Show the plot
plt.show()

# Name of top 10 destination
top_dest_flight
```



Out[34]:

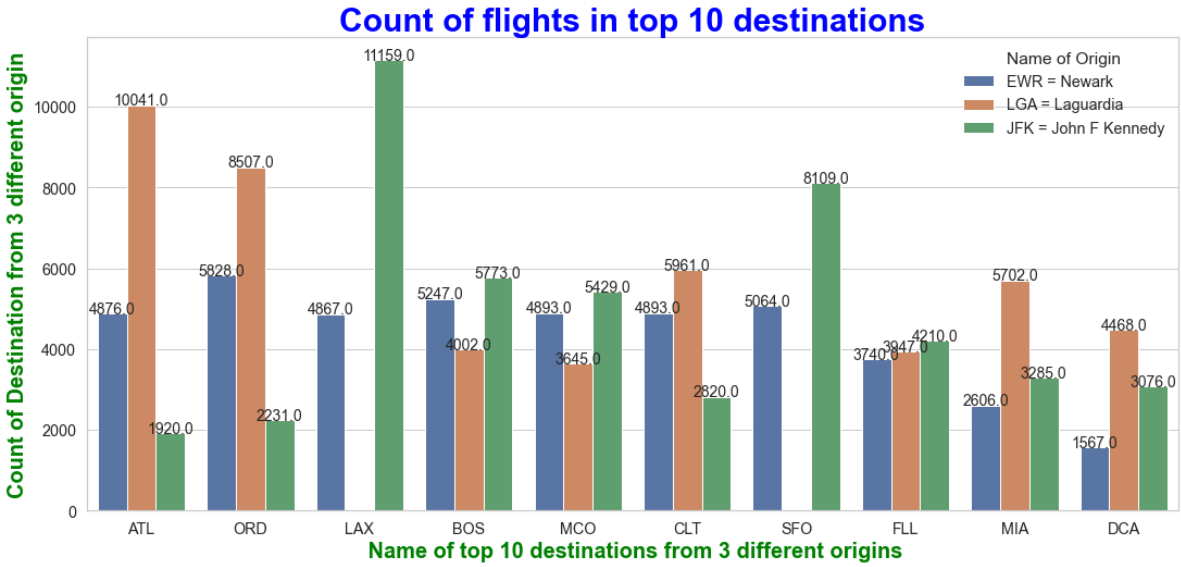
	dest	month
4	ATL	16837
68	ORD	16566
49	LAX	16026
11	BOS	15022
53	MCO	13967
23	CLT	13674
89	SFO	13173
35	FLL	11897
57	MIA	11593
28	DCA	9111

6.4 Top 10 destination using hue parameters as origin .

In [35]:

```
# Set Figure size
plt.figure(figsize= (18,8))

# Head of top 10 destinations in descending order
top_10_destination = flights['dest'].value_counts().head(10).sort_values(ascending = Fa

# Do count plot and use 'hue' parameter to get top 10 destination from three origin 'EW
ax = sns.countplot('dest', hue = 'origin',order = top_10_destination.index,data = fligh

# Styling of Count Plot
ax.set_title('Count of flights in top 10 destinations',fontweight = 'bold',fontsize = 3
ax.set_ylabel('Count of Destination from 3 different origin',fontweight = 'bold',fontsi
ax.set_xlabel(" Name of top 10 destinations from 3 different origins",fontweight = 'bol
plt.legend(labels=["EWR = Newark","LGA = Laguardia",'JFK = John F Kennedy'], title = "N



for rect in ax.patches:
    ax.text (rect.get_x() + rect.get_width()/2,rect.get_height(),rect.get_height(),hori

plt.show()
```

```
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
```



6.5 What is maximum numbers of flights headed to unique destination from origin ?

In [36]:

```
dest_flight  = flights.groupby('origin')['dest'].value_counts()
dest_flight.head(10)
```

Out[36]:

```
origin  dest
EWR     ORD     5828
        BOS     5247
        SFO     5064
        CLT     4893
        MCO     4893
        ATL     4876
        LAX     4867
        IAH     3923
        FLL     3740
        DFW     3014
Name: dest, dtype: int64
```

6.6 As Chicago has maximum number of flights so what is total number of unique airlines headed towards chicago ?

In [37]:

```
ord_dest = flights[flights['dest']=='ORD']

carrier_count = ord_dest['carrier'].unique()

print('Carrier fly to Chicago : ',carrier_count)
print('Count of flight: ',len(carrier_count))
```

```
Carrier fly to Chicago :  ['UA' 'AA' 'MQ' 'B6' '9E' 'OO' 'EV']
Count of flight:  7
```

6.7 Which aircraft is opearting for chicago and what are their tailnumber ?

In [38]:

```
ord_dest_count = ord_dest['tailnum'].unique()
print('Tailnumber of aircraft which is headed towards Chicago are:',ord_dest_count)
print('Count of tailnumber of aircraft which is headed towards Chicago are :', len(ord_
```

```
Tailnumber of aircraft which is headed towards Chicago are: ['N39463' 'N3ALA
A' 'N9EAMQ' ... 'N746JB' 'N527JB' 'N568JB']
Count of tailnumber of aircraft which is headed towards Chicago are : 1211
```

7 Flight Delay analysis

7.1 What is average 'monthly departure' delay by diffierent carriers ?

In [39]:

```
monthly_dep_delay = flights.groupby(['carrier','months_name'],axis = 0,as_index = True)
# monthly_dep_delay = flights.groupby(['months_name','carrier'],axis = 0,as_index = Tru
monthly_dep_delay.head(15)
```

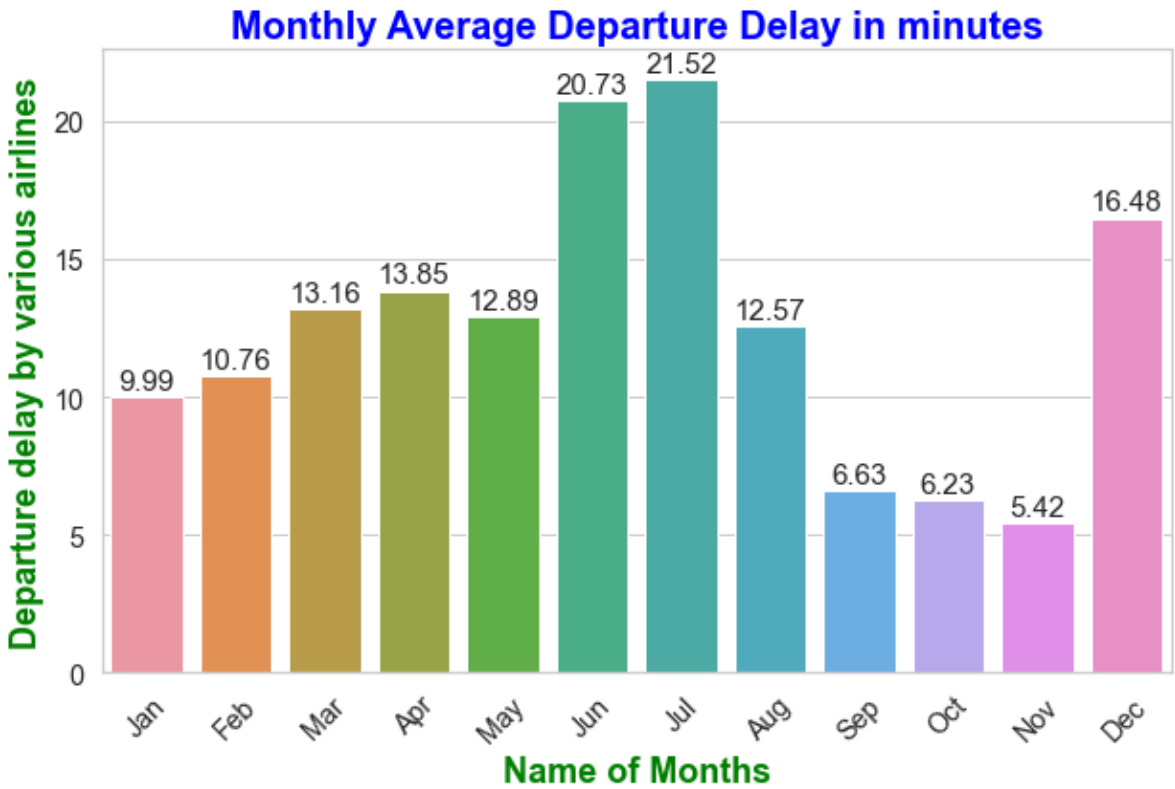
Out[39]:

carrier	months_name	dep_delay
9E	Apr	13
	Aug	17
	Dec	19
	Feb	16
	Jan	17
	Jul	30
	Jun	28
	Mar	13
	May	23
	Nov	7
	Oct	9
	Sep	8
AA	Apr	12
	Aug	7
	Dec	12

7.2 Pictoral representation of average monthly departure delay.

In [40]:

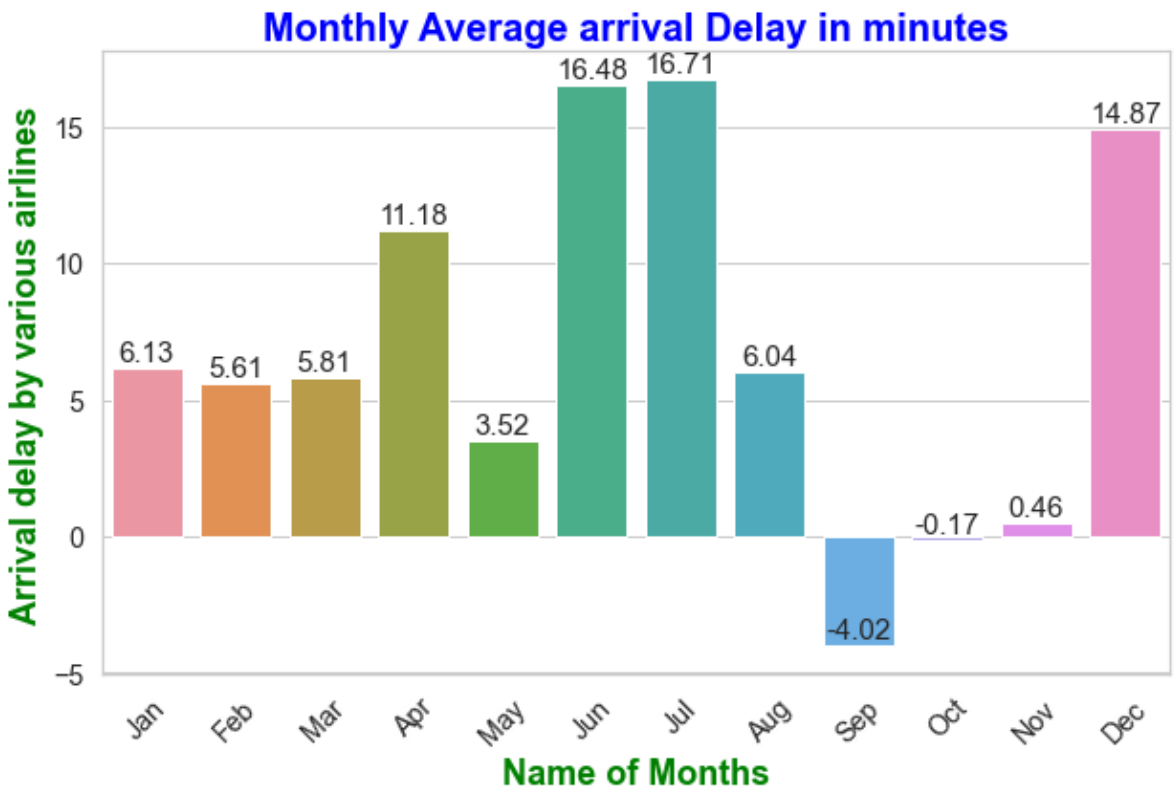
```
# set the figure size
plt.figure(figsize= (10,6))

# Drae barplot using "month_name" on x-axis and "average departure delay on y-axis"
ax = sns.barplot(x='months_name', y='dep_delay',data=flights,ci= False,order = ['Jan',
                                                                                 'Sep','(

# Set annotation in bar plot to see average departure delay in minute
for bar in ax.patches:
    ax.annotate(format(bar.get_height(),'.2f'),
                   (bar.get_x()+bar.get_width()/2,
                   bar.get_height()), ha = 'center', va = 'center',
                  size = 15,xytext = (0,8),
                  textcoords = 'offset points')

# Add tile ,xticks and yticks in chart
plt.title("Monthly Average Departure Delay in minutes",fontsize = 20,color = 'blue',for
plt.xticks(rotation = 45)
plt.xlabel('Name of Months', fontsize = 18,color= 'green', fontweight = 'bold')
plt.ylabel('Departure delay by various airlines',fontsize = 18,color= 'green',fontweigh

# Show the graph
plt.show()
```



Monthly Average arrival delay

In [41]:

```
# set the figure size
plt.figure(figsize= (10,6))

# Drae barplot using "month_name" on x-axis and "average departure delay on y-axis"
ax = sns.barplot(x='months_name', y='arr_delay',data=flights,ci= False,order = ['Jan',
                                                                                 'Sep','(

# Set annotation in bar plot to see average departure delay in minute
for bar in ax.patches:
    ax.annotate(format(bar.get_height(),'.2f'),
                   (bar.get_x()+bar.get_width()/2,
                   bar.get_height()), ha = 'center', va = 'center',
                  size = 15,xytext = (0,8),
                  textcoords = 'offset points')

# Add tile ,xticks and yticks in chart
plt.title("Monthly Average arrival Delay in minutes",fontsize = 20,color = 'blue',fontw
plt.xticks(rotation = 45)
plt.xlabel('Name of Months', fontsize = 18,color= 'green', fontweight = 'bold')
plt.ylabel('Arrival delay by various airlines',fontsize = 18,color= 'green',fontweight

# Show the graph
plt.show()
```



7.3 What are arrival and departure delay by carrier in year 2013 ?

In [42]:

```
# Average arrival delayed short by carrier in Year 2013.
most_delayed_carrier = flights.groupby('carrier').agg({'arr_delay':[np.size,np.mean]})

most_delayed_carrier.sort_values([('arr_delay','mean')],ascending = False).head(16)
```

Out[42]:

	arr_delay	
	size	mean
carrier		
F9	681	22
FL	3175	20
EV	51108	16
YV	544	16
OO	29	12
MQ	25037	11
WN	12044	10
B6	54049	9
9E	17294	7
UA	57782	4
US	19831	2
VX	5116	2
DL	47658	2
AA	31947	0
HA	342	-7
AS	709	-10

In [43]:

```
# Average arrival delayed short by carrier in Year 2013.
most_dep_delayed_carrier = flights.groupby('carrier').agg({'dep_delay':[np.size,np.mear

most_dep_delayed_carrier.sort_values([('dep_delay','mean')],ascending = False).head(16)
```

Out[43]:

	dep_delay	
	size	mean
carrier		
F9	681	20
EV	51108	20
YV	544	19
FL	3175	19
WN	12044	18
9E	17294	16
B6	54049	13
VX	5116	13
OO	29	13
UA	57782	12
MQ	25037	10
DL	47658	9
AA	31947	9
AS	709	6
HA	342	5
US	19831	4

In [44]:

```
len(flights.carrier.unique())
```

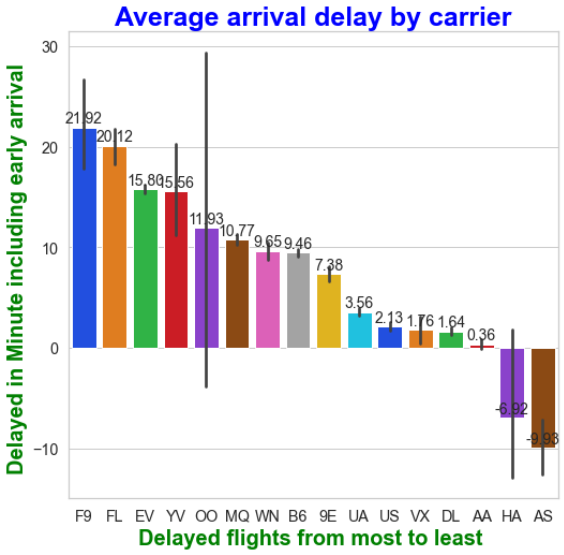
Out[44]:

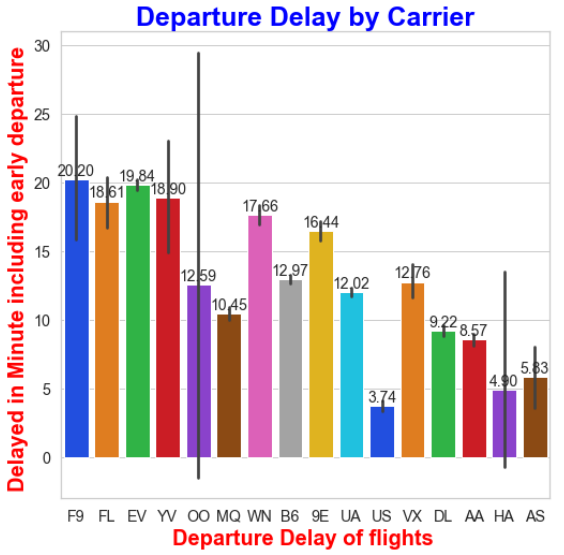
16

7.4 Pictorial average arrival delayed and departure delayed by carrier in year 2013 .

In [45]:

```python

# Set subplot with 1 row and 2 columns and figure size 20,8
f,ax = plt.subplots(1,2,figsize=(18,8))
#-----------------------------------------------------------------------------------------
# Draw barplot where on x- axis is 'carrier' and y-axis is 'arr_delay'. Use ax[0].
sns.barplot('carrier','arr_delay',data =flights ,ax = ax[0],
           order =['F9', 'FL', 'EV', 'YV','OO','MQ','WN','B6','9E','UA', 'US','VX','DL'

# Set xlabel and ylabel
ax[0].set_title('Average arrival delay by carrier', fontsize = 25, fontweight ='bold',
ax[0].set_xlabel('Delayed flights from most to least ',fontsize = 20,fontweight = 'bold
ax[0].set_ylabel('Delayed in Minute including early arrival ',fontsize = 20,fontweight

# Add annotation in bar plot
for bar in ax[0].patches:
    ax[0].annotate(format(bar.get_height(),'.2f'),
                   (bar.get_x()+bar.get_width()/2,
                   bar.get_height()), ha = 'center', va = 'center',
                  size = 14,xytext = (0,8),
                  textcoords = 'offset points')


#-----------------------------------------------------------------------------------------
# Draw box plot using ax[1] on x- axis is 'carrier' and y-axis is 'arr_delay'
sns.barplot('carrier','dep_delay',data = flights,ax = ax[1],
           order =['F9', 'FL', 'EV', 'YV','OO','MQ','WN','B6','9E','UA', 'US','VX','DL'

ax[1].set_title('Departure Delay by Carrier',fontsize = 25,fontweight ='bold', color =
ax[1].set_xlabel('Departure Delay of flights ',fontsize = 20,fontweight = 'bold',color
ax[1].set_ylabel('Delayed in Minute including early departure ',fontsize = 20,fontweigh

# Add annotation in bar plot
for bar in ax[1].patches:
    ax[1].annotate(format(bar.get_height(),'.2f'),
                   (bar.get_x()+bar.get_width()/2,
                   bar.get_height()), ha = 'center', va = 'center',
                  size = 14,xytext = (0,8),
                  textcoords = 'offset points')
#-----------------------------------------------------------------------------------------
plt.show()
print('*****************************************************************************
print(['WN: Southwest Airlines',  'AA: American Airlines', 'MQ: American Eagle Airlines
       'OO: Skywest Airlines','DL: Delta Airlines','US: US Airways','YV: Mesa Airlines'
       'EV: Atlantic Southeast Airlines','FL: AirTran Airways','F9: Frontier Airlines',
       'B6: JetBlue Airways','9E: Pinnacle Airlines','AS: Alaska Airlines'])
```






```
**********************************************************************
*********************************************
['WN: Southwest Airlines', 'AA: American Airlines', 'MQ: American Eagle Airl
ines', 'UA: United Airlines', 'OO: Skywest Airlines', 'DL: Delta Airlines',
'US: US Airways', 'YV: Mesa Airlines', 'HA: Hawaiian Airlines', 'EV: Atlanti
c Southeast Airlines', 'FL: AirTran Airways', 'F9: Frontier Airlines', 'VX:
Virgin America', 'B6: JetBlue Airways', '9E: Pinnacle Airlines', 'AS: Alaska
Airlines']
```

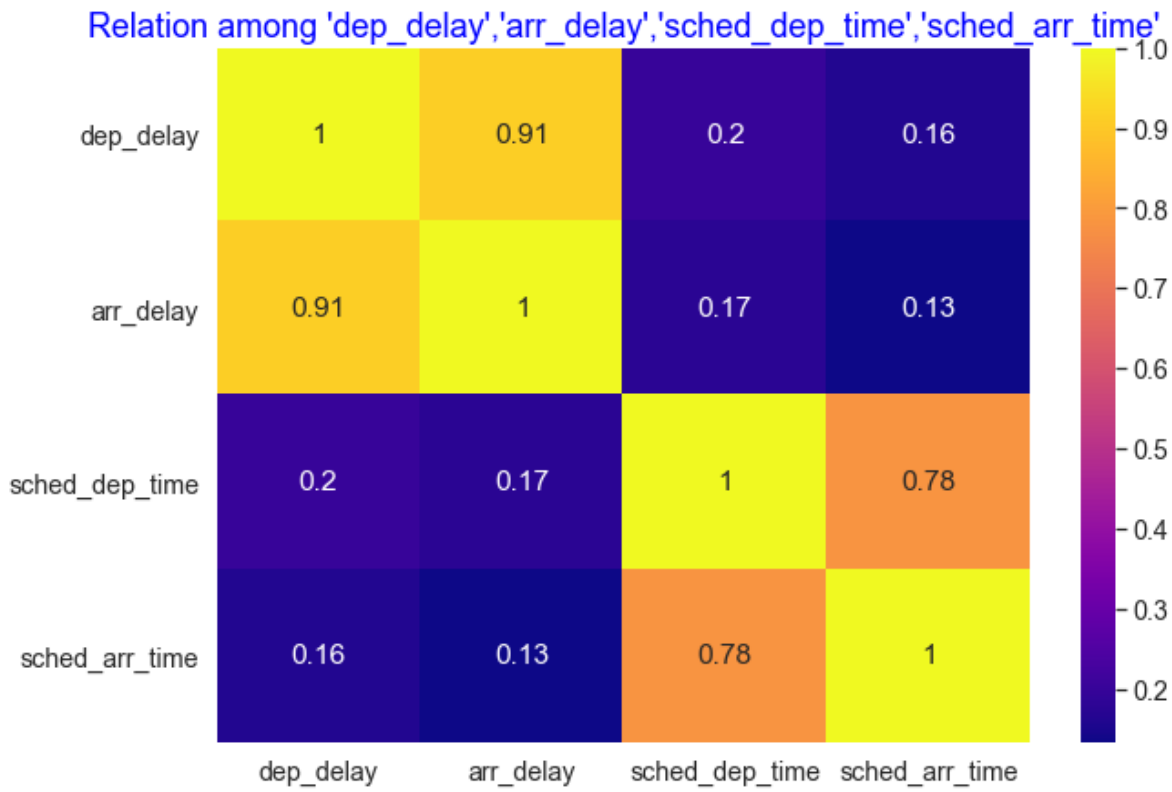
7.5 What is relation between departure delay and arrival delay of airlines ?

In [46]:

```
plt.figure(figsize = (10,7))
flights_data = flights.copy()
flight_delay_data = flights_data[['dep_delay','arr_delay','sched_dep_time','sched_arr_t
ax = sns.heatmap(flight_delay_data.corr(),cmap='plasma',annot=True)

ax.set_title("Relation among 'dep_delay','arr_delay','sched_dep_time','sched_arr_time''

```

Out[46]:

```
Text(0.5, 1.0, "Relation among 'dep_delay','arr_delay','sched_dep_time','sch
ed_arr_time'")
```



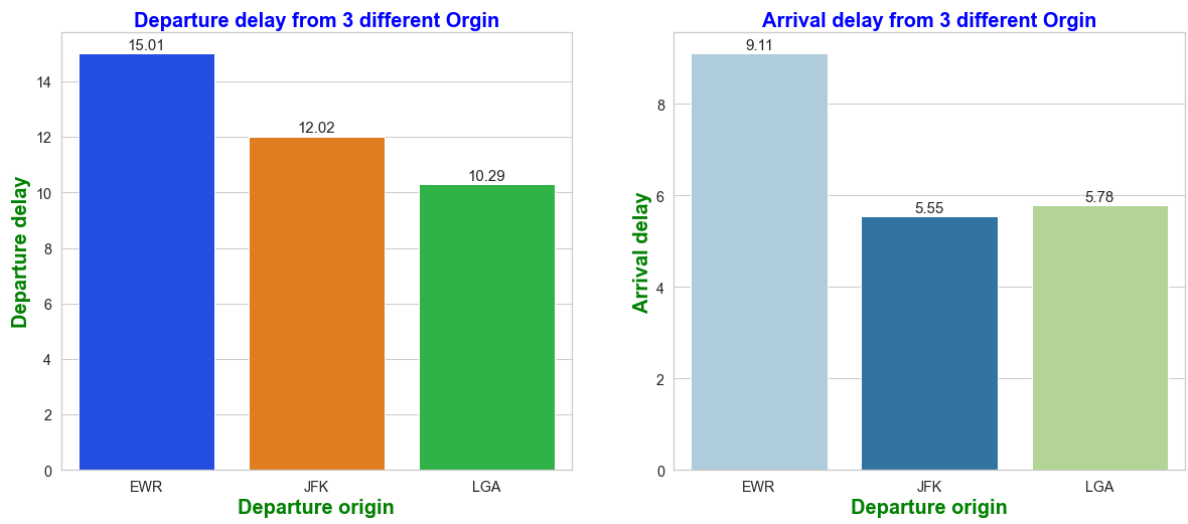
Observation

1. Arrival delay and departure delay are highly corelated that means if flight is delayed by departure than it will be delayed by arrrival as well.
2. Correlation between scheduled departure time and departure delay is 0.2 means both are not much depend on each other.

7.6 What is average departure delay by carrier with origin ?

In [47]:

```
f,ax = plt.subplots(1,2,figsize=(20,8))

sns.barplot('origin','dep_delay', data=flights,ax=ax[0], order=['EWR', 'JFK', 'LGA'],pa

for bar in ax[0].patches:
    ax[0].annotate(format(bar.get_height(),'.2f'),
                   (bar.get_x()+bar.get_width()/2,
                    bar.get_height()), ha = 'center', va = 'center',
                  size = 15,xytext = (0,8),
                  textcoords = 'offset points')

ax[0].set_title('Departure delay from 3 different Orgin', color = 'blue',fontsize = 20,
ax[0].set_xlabel('Departure origin',color = 'green',fontsize = 20, fontweight = 'bold')
ax[0].set_ylabel('Departure delay',color = 'green',fontsize = 20, fontweight = 'bold')

#-----------------------------------------------------------------------------------------
sns.barplot('origin','arr_delay', data=flights,ax=ax[1], order=['EWR', 'JFK', 'LGA'], p

for bar in ax[1].patches:
    ax[1].annotate(format(bar.get_height(),'.2f'),
                   (bar.get_x()+bar.get_width()/2,
                    bar.get_height()), ha = 'center', va = 'center',
                  size = 15,xytext = (0,8),
                  textcoords = 'offset points')

ax[1].set_title('Arrival delay from 3 different Orgin', color = 'blue',fontsize = 20, f
ax[1].set_xlabel('Departure origin',color = 'green',fontsize = 20, fontweight = 'bold')
ax[1].set_ylabel('Arrival delay',color = 'green',fontsize = 20, fontweight = 'bold')


plt.show()
```

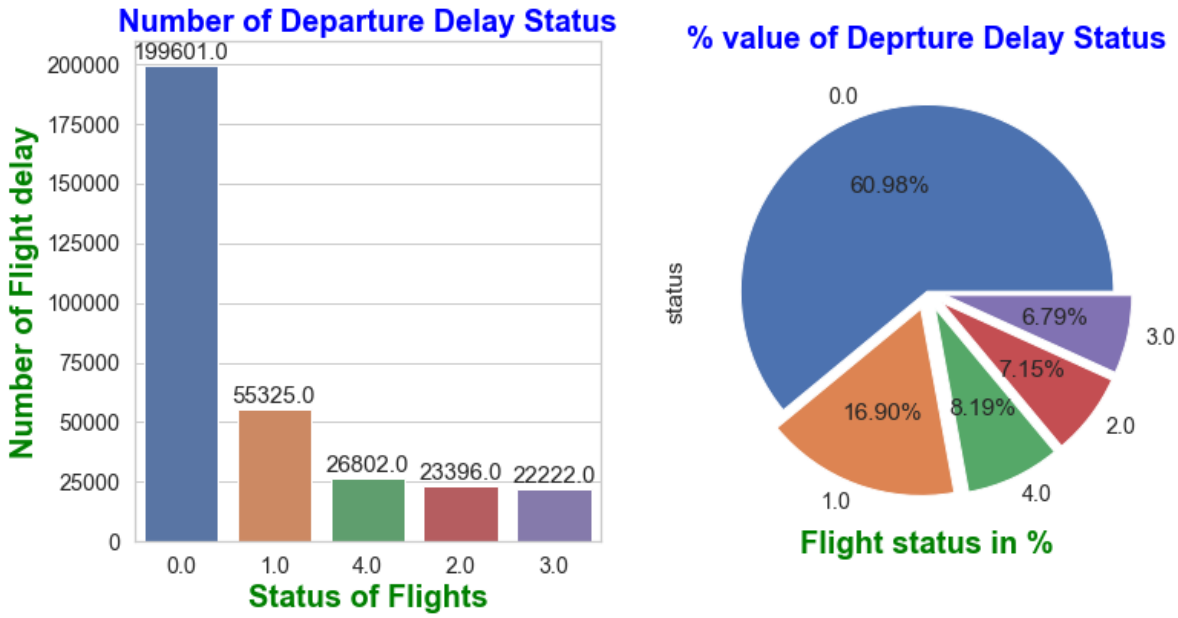


8. Ontime Departure and Arrival Analysis of Flights

8.1 Departure delay by 'Status' which is created by condition and inserted in new column.

In [48]:

```
# Insert column 'Status' on the basis of delay.
flights_data.loc[flights_data.dep_delay <=0,'status'] = 0
flights_data.loc[flights_data.dep_delay >=1,'status'] = 1
flights_data.loc[flights_data.dep_delay >=15,'status'] = 2
flights_data.loc[flights_data.dep_delay >=30,'status'] = 3
flights_data.loc[flights_data.dep_delay >=60,'status'] = 4


# Use subplot to know status of delay

f,ax=plt.subplots(1,2,figsize=(12,6))


sns.countplot('status',order = flights_data['status'].value_counts().index, data=flight
ax[0].set_title('Number of Departure Delay Status', color = 'blue', fontsize = 20, font
ax[0].set_xlabel('Status of Flights',fontsize = 20, color = 'green',fontweight = 'bold'
ax[0].set_ylabel('Number of Flight delay',fontsize = 20, color = 'green',fontweight = '

# Set annotation
for bar in ax[0].patches:
    ax[0].annotate(format(bar.get_height(),'.1f'),
                   (bar.get_x()+bar.get_width()/2,
                   bar.get_height()), ha = 'center', va = 'center',
                  size = 15,xytext = (0,8),
                  textcoords = 'offset points')
#-----------------------------------------------------------------------------------

# Matplot pie chart using explode
flights_data['status'].value_counts().plot.pie(explode=[0,0.1,0.1,0.1,0.1],autopct='%1.
ax[1].set_title('% value of Deprture Delay Status',color = 'blue', fontsize = 20, fontw
ax[1].set_xlabel('Flight status in %',fontsize = 20, color = 'green',fontweight = 'bold

plt.show()
print('*****************************************************************************
print("status for <0 = 0,  status for >1 = 1, Status for >15 = 2, Status for >30 = 3, S
```



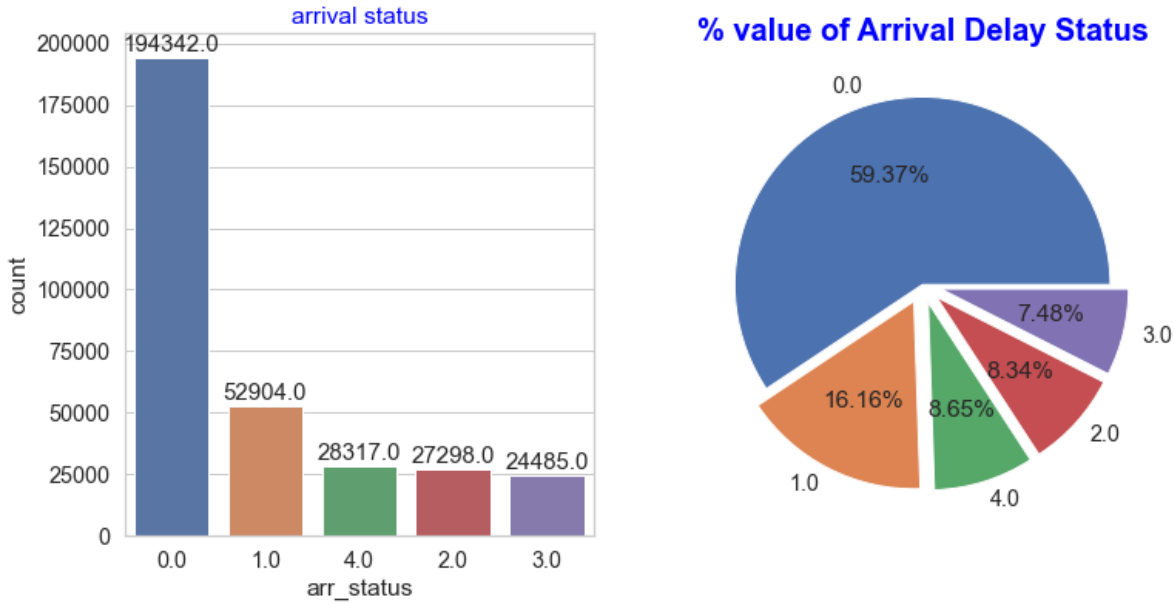

```
*****************************************************************************
*******************************************
```

```
status for <0 = 0,  status for >1 = 1, Status for >15 = 2, Status for >30
= 3, Status for >60 = 4
```

8.2 Arrival delay status.

In [49]:

```
# Insert column 'Status' on the basis of delay.
flights_data.loc[flights_data.arr_delay <=0,'arr_status'] = 0
flights_data.loc[flights_data.arr_delay >=1,'arr_status'] = 1
flights_data.loc[flights_data.arr_delay >=15,'arr_status'] = 2
flights_data.loc[flights_data.arr_delay >=30,'arr_status'] = 3
flights_data.loc[flights_data.arr_delay >=60,'arr_status'] = 4

# Use subplot to know status of delay

f,ax=plt.subplots(1,2,figsize=(12,6))


sns.countplot('arr_status',order = flights_data['arr_status'].value_counts().index, dat
ax[0].set_title('Number of Arrival Delay Status', color = 'blue', fontsize = 20, fontwe
ax[0].set_title('arrival status')
for bar in ax[0].patches:
    ax[0].annotate(format(bar.get_height(),'.1f'),
                   (bar.get_x()+bar.get_width()/2,
                   bar.get_height()), ha = 'center', va = 'center',
                  size = 15,xytext = (0,8),
                  textcoords = 'offset points')

# Matplot pie chart using explode
flights_data['arr_status'].value_counts().plot.pie(explode=[0,0.1,0.1,0.1,0.1],autopct=
ax[1].set_title('% value of Arrival Delay Status',color = 'blue', fontsize = 20, fontwe
ax[1].set_ylabel(''),

plt.show()
print('*********************************************************************************
print("status for <0 = 0,  status for >1 = 1, Status for >15 = 2, Status for >30 = 3, S
```




```
*********************************************************************************
*****************************************
status for <0 = 0,  status for >1 = 1, Status for >15 = 2, Status for >30 =
3, Status for >60 = 4
```

Getting Delayed flights

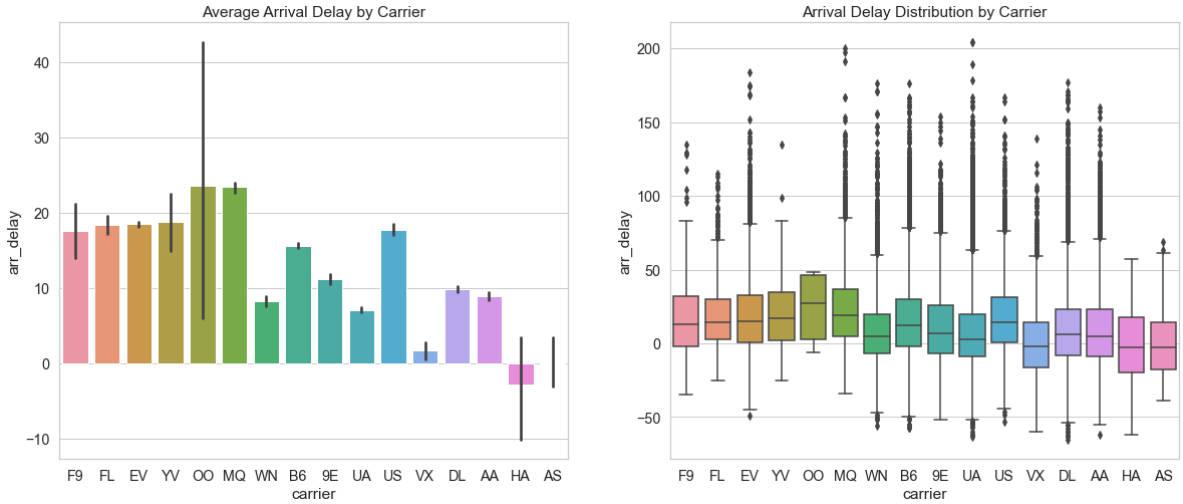
In [50]:

```
delayed_flights = flights_data[(flights_data.status >=1) & (flights_data.status <=3) ]
```

8.3 Average arrival delay by carrier.

In [51]:

```
#Average arrival delay by carrier
f,ax=plt.subplots(1,2,figsize=(20,8))
sns.barplot('carrier','arr_delay', data=delayed_flights,ax=ax[0], order=['F9', 'FL', 'E
                                                                          '9E','U

ax[0].set_title('Average Arrival Delay by Carrier')
sns.boxplot('carrier','arr_delay', data=delayed_flights,ax=ax[1], order=['F9', 'FL', 'E
                                                                          '9E','U
ax[1].set_title('Arrival Delay Distribution by Carrier')

plt.show()

print(['WN: Southwest Airlines', 'AA: American Airlines', 'MQ: American Eagle Airlines
       'OO: Skywest Airlines','DL: Delta Airlines','US: US Airways','YV: Mesa Airlines'
       'EV: Atlantic Southeast Airlines','FL: AirTran Airways','F9: Frontier Airlines',
       'B6: JetBlue Airways','9E: Pinnacle Airlines','AS: Alaska Airlines'])

```




```
['WN: Southwest Airlines', 'AA: American Airlines', 'MQ: American Eagle Airl
ines', 'UA: United Airlines', 'OO: Skywest Airlines', 'DL: Delta Airlines',
'US: US Airways', 'YV: Mesa Airlines', 'HA: Hawaiian Airlines', 'EV: Atlanti
c Southeast Airlines', 'FL: AirTran Airways', 'F9: Frontier Airlines', 'VX:
Virgin America', 'B6: JetBlue Airways', '9E: Pinnacle Airlines', 'AS: Alaska
Airlines']
```

Top 5 US airline

1. American Airlines (AA) | 2. Southwest Airlines (WN) | 3. Delta Air Lines (DL) | 4. United Airlines (UA) | 5. Alaska Airlines (AS)

Note : generate an average delay of 37.2 minutes. Alaska Airlines, with an 34 minutes per flight, the second lowest of all the carriers.

Highest Delay

- Carriers with higher average delay generation are

1. Skywest Airlines(OO) with 60 minutes per flight,
2. Mesa Airlines (YV) with 50 minutes per flight,
3. Pinnacle Airlines (9E) with 49 minutes per flight.

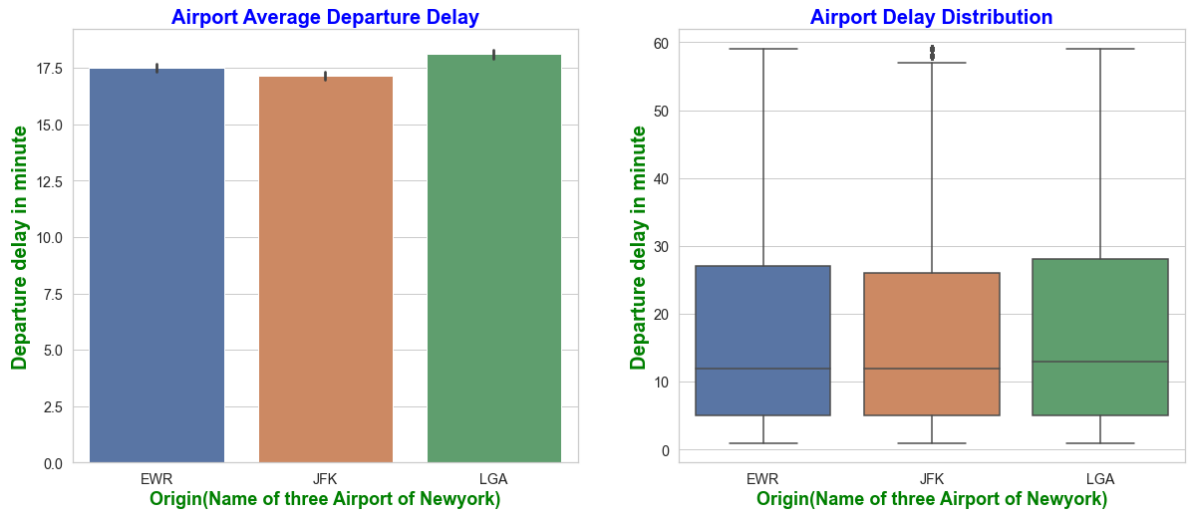
Note: The error bar provide the insight that airlines with low number of flights having higher standard deviation distribution from the mean (OO, HA, YV, F9, AS); so it seems like size matters.

- The **boxplot shows**, airlines with higher number of flights results having a higher chance of extreme waiting situation. American Eagle Airlines (MQ), American Airlines(AA), Delta Airline(DL) registered the maximum Carrier Delay for 2013 with an exception of Hawaiian Airlines (HA).

8.4 Average departure delay by carrier

In [52]:

```
#Average departure delay by carrier

# Set Figure size
f,ax=plt.subplots(1,2,figsize=(20,8))

# subplot 1 row and 0 column drwa bar plot to know average departure delay from origin
sns.barplot('origin','dep_delay', data=delayed_flights,ax=ax[0], order=['EWR', 'JFK', '

# Set Title and xlabel and ylabel in ax[0]
ax[0].set_title('Airport Average Departure Delay', fontsize = 20, color = 'blue',fontwe
ax[0].set_xlabel('Origin(Name of three Airport of Newyork)',fontsize = 18, color = 'gre
ax[0].set_ylabel('Departure delay in minute',fontsize = 20, color = 'green',fontweight

# In subplot row 1 and 1 column draw Box plot
sns.boxplot('origin','dep_delay', data=delayed_flights,ax=ax[1], order=['EWR', 'JFK', '

ax[1].set_title('Airport Delay Distribution',fontsize = 20, color = 'blue',fontweight =
ax[1].set_xlabel('Origin(Name of three Airport of Newyork)',fontsize = 18, color = 'gre
ax[1].set_ylabel('Departure delay in minute',fontsize = 20, color = 'green',fontweight

plt.show()

```



There seems to be a correlation between the number of flights operated and departure delay. A descending pattern can be seen from barplot for average departure delay per flight. Considering, the assumption that JFK being a busiest airport among 3 due to international flights; so the maximum departure delay for 2013 is registered by JFK.

9. Perfomance / speed analysis of Aircraft

In [53]:

```
flights.columns
```

Out[53]:

```
Index(['year', 'month', 'day', 'dep_time', 'sched_dep_time', 'dep_delay',
       'arr_time', 'sched_arr_time', 'arr_delay', 'carrier', 'flight',
       'tailnum', 'origin', 'dest', 'air_time', 'distance', 'hour', 'minut
e',
       'time_hour', 'date', 'months_name', 'carrier_name'],
      dtype='object')
```

In [54]:

```
scheduled_departure = flights_data['sched_dep_time'].count()
actual_departure = flights_data['dep_time'].count()

```

9.1 Ratio of scheduled flight and cancelled flight.

In [55]:

```
# Count of Scheduled departure
scheduled_departure = flights_data.count()['sched_dep_time']

# Count of Actual departure
actual_departure = flights_data['dep_time'].count()

# Cancellation of departuer
cancel_departure = scheduled_departure - actual_departure

# Ratio of flight operation
flight_operation_ratio = actual_departure / scheduled_departure * 100

flight_cancellation_ratio= (100 - flight_operation_ratio)

print("Total_scheduled_departure: ", scheduled_departure)
print("Total_actually_Operated: ", actual_departure)
print("Total_flight_Cancelled: ", cancel_departure)
print("\n")
print("Ratio operated flights over scheduled flights: %s" % flight_operation_ratio,'%')
print("Ratio of cancelled flights: %s" % flight_cancellation_ratio,'%')
```

```
Total_scheduled_departure:  327346
Total_actually_Operated:  327346
Total_flight_Cancelled:  0


Ratio operated flights over scheduled flights: 100.0 %
Ratio of cancelled flights: 0.0 %
```

9.2 In which month have highest average departure delay

In [56]:

```
# The day and month having highest average delay departure by

delay_day=flights.groupby(['day','month'], as_index=False).agg({'dep_delay': 'mean'})
delay_day_max=delay_day['dep_delay'].max()
delay_day_info=delay_day[delay_day['dep_delay']==delay_day_max]
print("The day and month having highest delay by average for departures" '\n' ,delay_da
```

```
The day and month having highest delay by average for departures
     day  month  dep_delay
86    8      3         84
```

9.3 In which month have highest number of flights delay

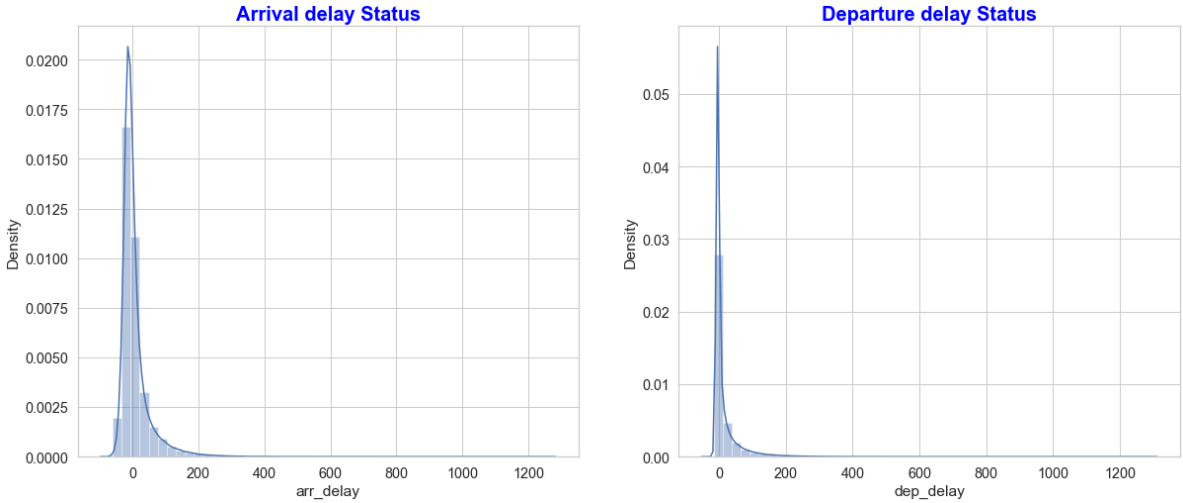
In [57]:

```
# The day and month having highest number of flight delay

max_flightdelay_day=flights[flights['dep_delay'] > 0].groupby(['day','month'], as_index
max_flightdelay_info = max_flightdelay_day[max_flightdelay_day['flight'].max() == max_f
print("Day and month which have highest number of flight delay" '\n' ,max_flightdelay_i
```

```
Day and month which have highest number of flight delay
      day  month  flight
275   23     12     673
```

In [58]:

```
# histogram for arrival and departure delay

f,ax=plt.subplots(1,2,figsize=(20,8))
sns.distplot(flights['arr_delay'], ax=ax[0])

ax[0].set_title('Arrival delay Status',fontsize = 20, color = 'blue', fontweight = 'bol

sns.distplot(flights['dep_delay'], ax=ax[1])
ax[1].set_title('Departure delay Status',fontsize = 20, color = 'blue', fontweight = 'b
plt.show()

print("Skewness_arr: %f" % flights['arr_delay'].skew())
print("Kurtosis_arr: %f" % flights['arr_delay'].kurt())
print("Skewness_dep: %f" % flights['dep_delay'].skew())
print("Kurtosis_dep: %f" % flights['dep_delay'].kurt())
```




```
Skewness_arr: 3.716817
Kurtosis_arr: 29.233044
Skewness_dep: 4.818018
Kurtosis_dep: 44.355043
```

Observation:

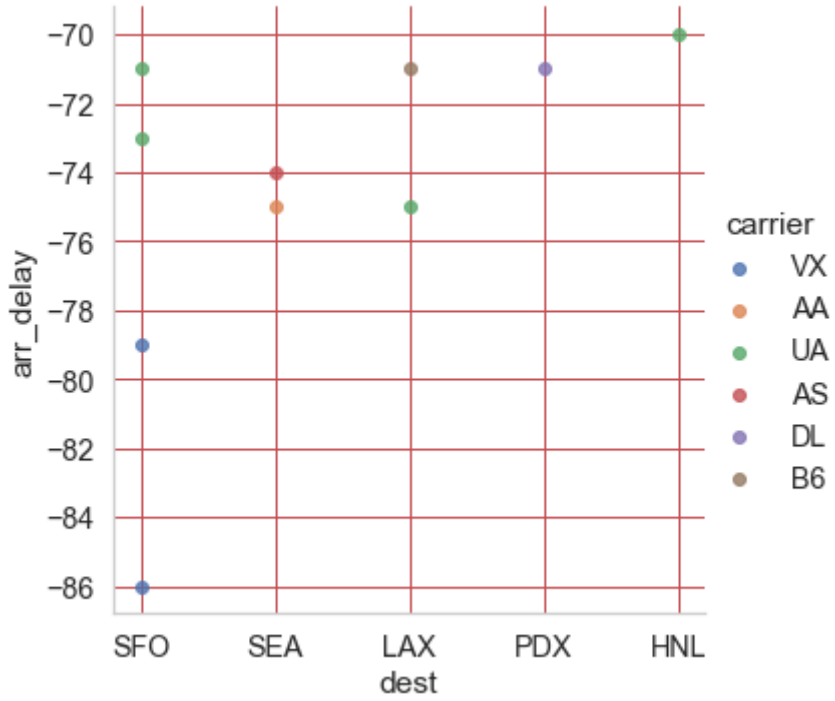
1. It can be seen on the histogram and by the skewness and kurtosis indexes, that the **skewness is > 1** which reflect the data distribution is **highly positively skewed** and **Kurtosis > 3** shows the leptokurtic distribution, having longer and fatter tail with a central peak higher and sharper.
2. The **histogram** shows the delays are mostly located on the **left side of the graph**, with a long tail to the right. The majority of delays are short, and the longer delays, while unusual, are more heavy loaded in time.

9.4 Top 5 destination where flight arrives early than expected arrival time.

In [59]:

```
# Top5 destination where flight arrives early than expected arrival time

plt.figure(figsize = (20,10))
Ontime_Early_flights =flights_data[(flights_data.status <= 0)]

top10flight=(Ontime_Early_flights.sort_values(['arr_delay'], ascending=True)).head(10)
top10flight_details=pd.DataFrame(top10flight, columns=['day','month','origin','dest','a
print("Top 5 destinations where flights arrives early than expected arrival time","\n")


gx= sns.lmplot(x = 'dest', y='arr_delay', data = top10flight_details, fit_reg=False, hu



plt.grid(True, color='r')

plt.show()
```

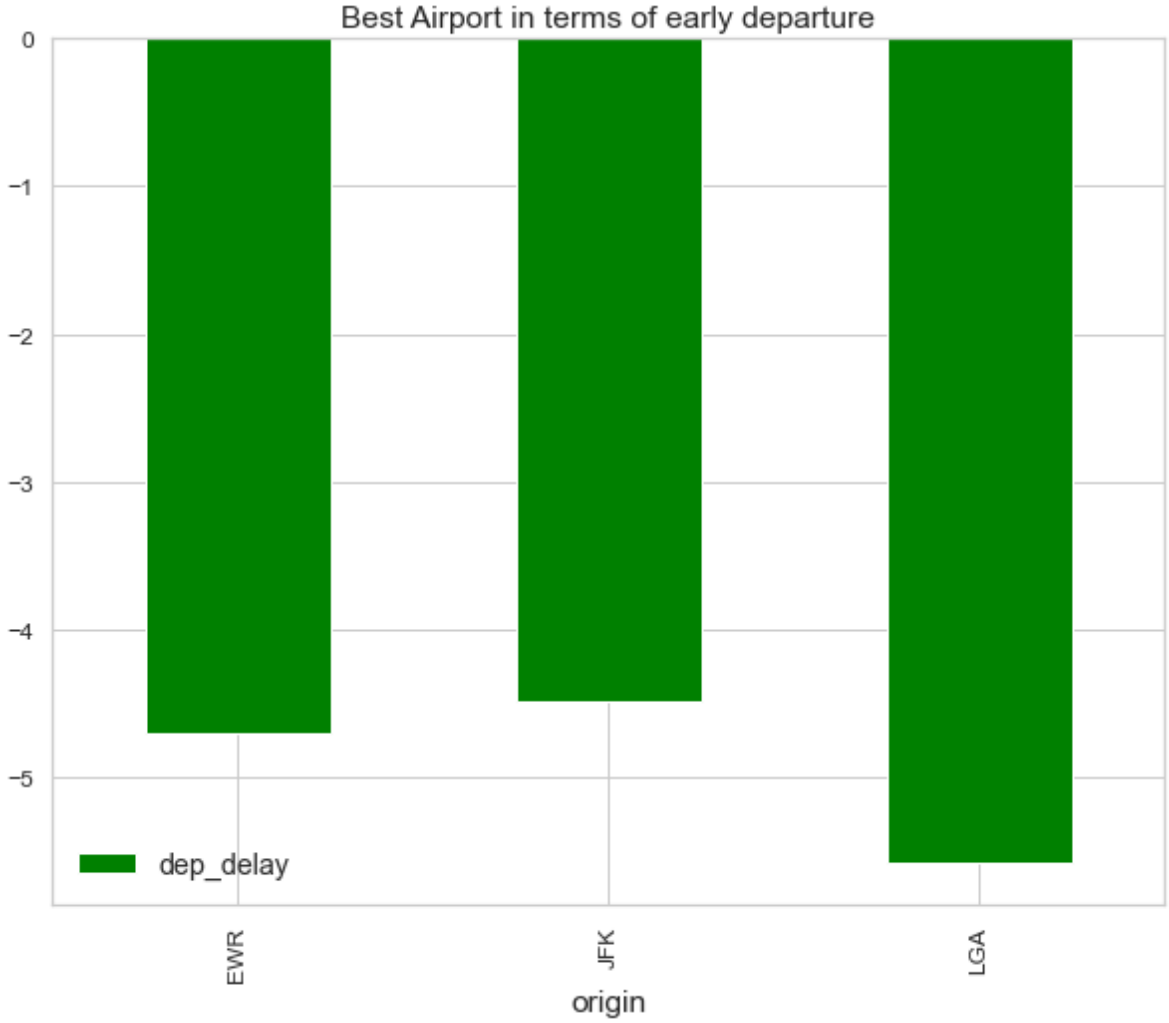
```
Top 5 destinations where flights arrives early than expected arrival time


<Figure size 1440x720 with 0 Axes>
```



9.5 The best airport in terms of early departure from NYC.

In [60]:

```
# Best airport in terms of early departure from NYC

airport_info = pd.DataFrame(flights_data,columns=['day','month','dep_delay','arr_delay'
airport_origin = airport_info[airport_info['dep_delay']<0]
best_airport = airport_origin.sort_values(['dep_delay']).groupby(['origin']).agg({'dep_
best_airport.plot(kind='bar',color='green', title ="Best Airport in terms of early depa
plt.show()
plt.close()
```



In [61]:

```
flights.columns
```

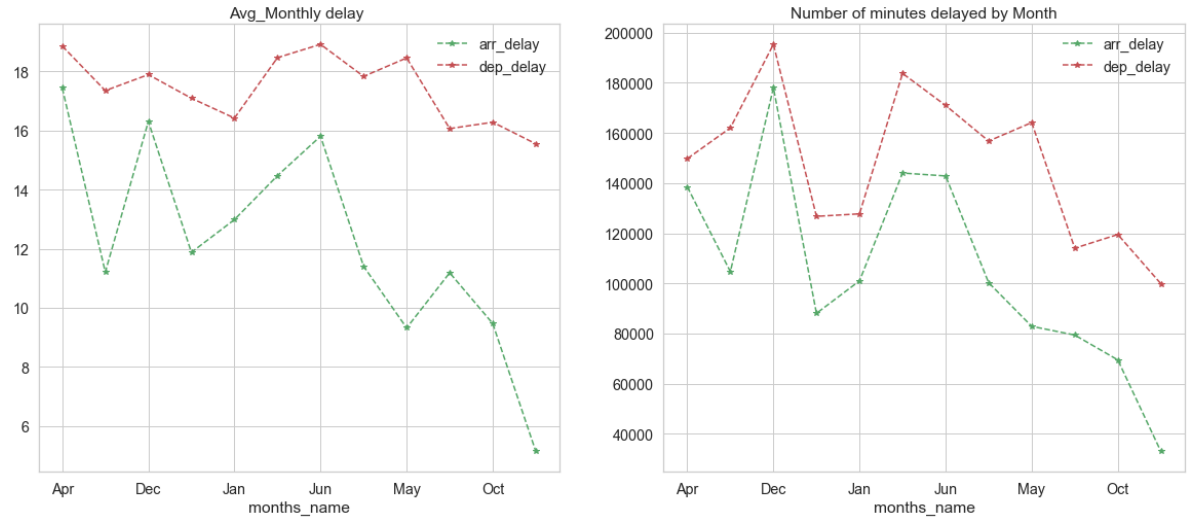
Out[61]:

```
Index(['year', 'month', 'day', 'dep_time', 'sched_dep_time', 'dep_delay',
       'arr_time', 'sched_arr_time', 'arr_delay', 'carrier', 'flight',
       'tailnum', 'origin', 'dest', 'air_time', 'distance', 'hour', 'minut
e',
       'time_hour', 'date', 'months_name', 'carrier_name'],
      dtype='object')
```

9.6 Graphical analysis of average monthly delay of aircraft and number of minute delayed by months.

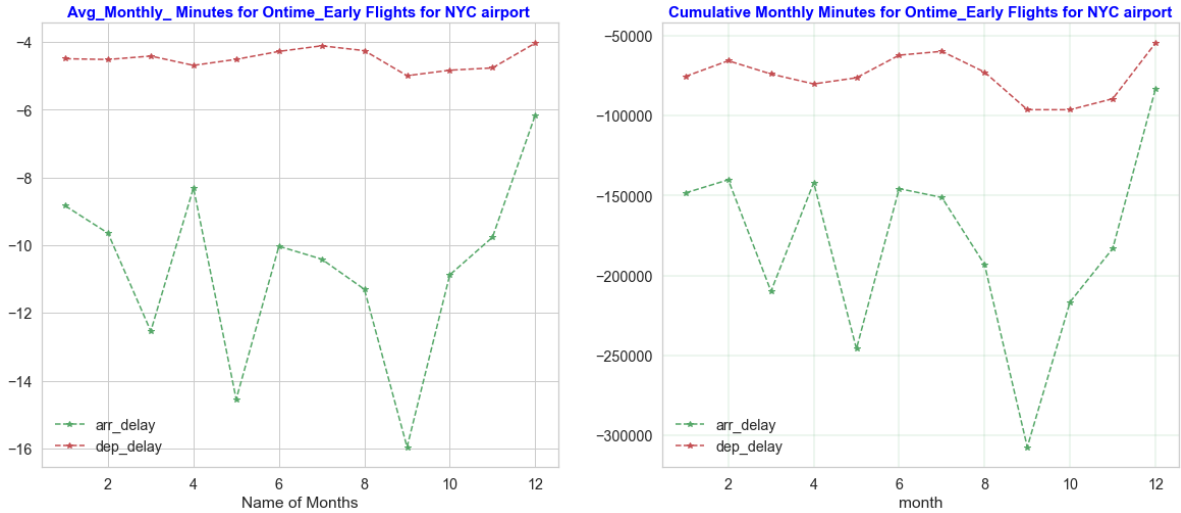
In [62]:

```
f,ax=plt.subplots(1,2,figsize=(20,8))
delayed_flights[['months_name','arr_delay','dep_delay']].groupby(['months_name']).mean(
ax[0].set_title('Avg_Monthly delay ')

delayed_flights[['months_name','arr_delay','dep_delay']].groupby(['months_name']).sum()
ax[1].set_title('Number of minutes delayed by Month')
plt.show()
```



In [63]:

```
#Monthly_Delays of Ontime_Early_ Flights (only for status 0)

f,ax=plt.subplots(1,2,figsize=(20,8))
Ontime_Early_flights[['month','arr_delay','dep_delay']].groupby(['month']).mean().plot(
ax[0].set_title('Avg_Monthly_ Minutes for Ontime_Early Flights for NYC airport ',color=
ax[0].set_xlabel('Name of Months')


Ontime_Early_flights[['month','arr_delay','dep_delay']].groupby(['month']).sum().plot(a
ax[1].set_title('Cumulative Monthly Minutes for Ontime_Early Flights for NYC airport',c
plt.grid(True, color='g',linewidth='0.2')
plt.show()
```



9.7 Other explanation like speed ananlysis / heatmap for correlation

In [64]:

```
# Flight Speed Analysis

flight_speed =flights_data['distance'] / (flights_data['air_time']/60)
flights_data['flight_speed'] =flight_speed
flights_data.sort_values(by='flight_speed', ascending=False).head(5)
```

Out[64]:

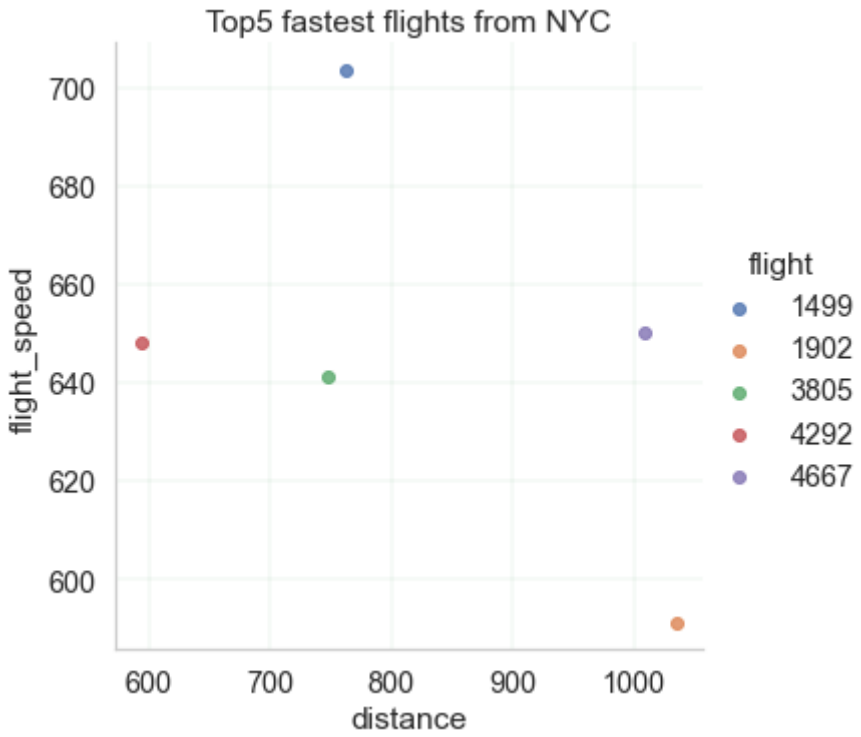
	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr
216447	2013	5	25	1709	1700	9	1923	1937	
251999	2013	7	2	1558	1513	45	1745	1719	
205388	2013	5	13	2040	2025	15	2225	2226	
157516	2013	3	23	1914	1910	4	2045	2043	
10223	2013	1	12	1559	1600	-1	1849	1917	

In [65]:

```
# Top5 fastest flights details from NYC

fastest_5=flights_data.loc[:, ['flight','carrier_name', 'tailnum','distance','air_time'
sns.lmplot(x = 'distance', y='flight_speed', data = fastest_5, fit_reg=False, hue='flig
plt.grid(True, color='g', linewidth=0.1)
plt.title("Top5 fastest flights from NYC")
fastest_5
```

Out[65]:

	flight	carrier_name	tailnum	distance	air_time	flight_speed
216447	1499	DELTA AIR LINES	N666DN	762	65	703
251999	4667	EXPRESSJET AIRLINES(EV)	N17196	1008	93	650
205388	4292	EXPRESSJET AIRLINES(EV)	N14568	594	55	648
157516	3805	EXPRESSJET AIRLINES(EV)	N12567	748	70	641
10223	1902	DELTA AIR LINES	N956DL	1035	105	591



In [66]:

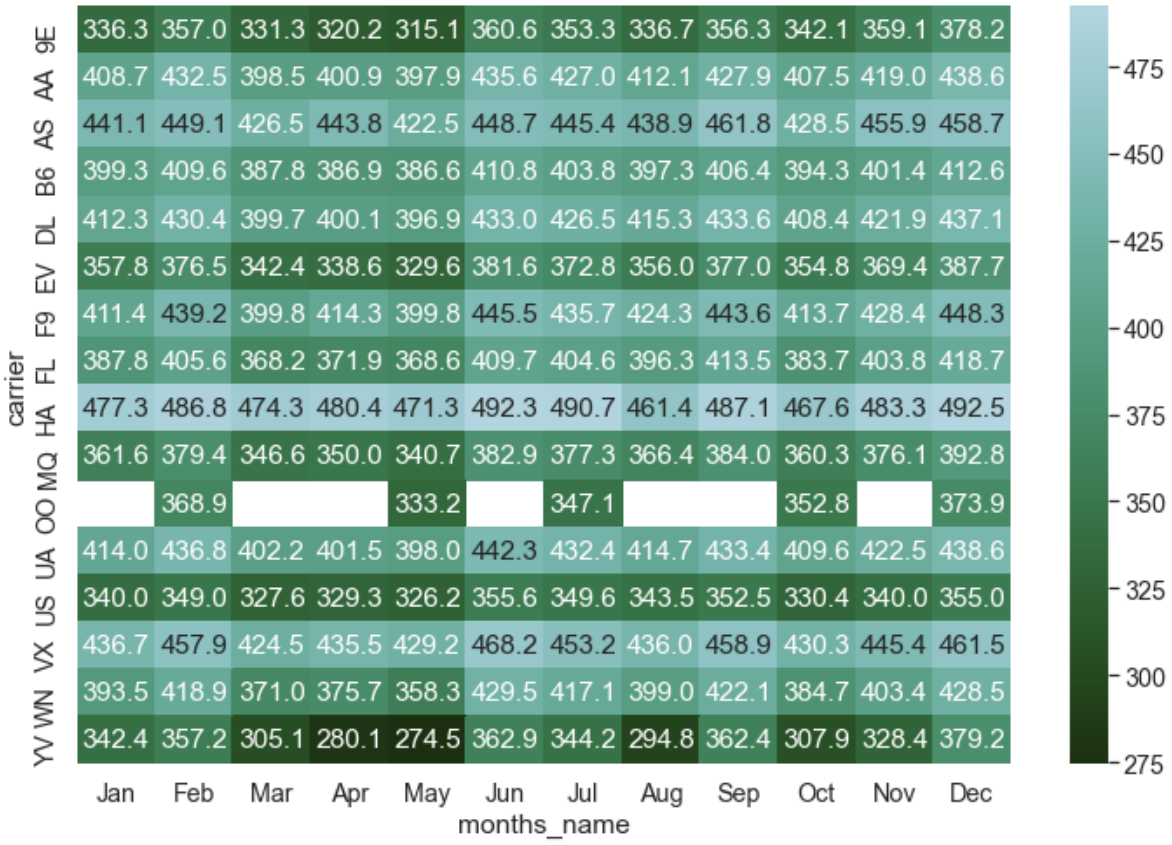
```
carrier_hmap=flights_data.drop(['dep_time','sched_dep_time','arr_time','sched_arr_time'
                                'time_hour','month','year','origin','dest','tailnum','sta
                                   'distance','flight','air_time','carrier_name'], axis=1
carrier_hmap.head()
```

Out[66]:

	day	dep_delay	arr_delay	carrier	months_name	arr_status	flight_speed
0	1	2	11	UA	Jan	1	370
1	1	4	20	UA	Jan	2	374
2	1	2	33	AA	Jan	3	408
3	1	-1	-18	B6	Jan	0	517
4	1	-6	-25	DL	Jan	0	394

In [68]:

```
carrier_hmap_OO = pd.pivot_table(carrier_hmap,values='flight_speed', aggfunc='mean', in
carrier_hmap_OO.head()

plt.figure(figsize=(12,8))
xticks = ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
cmap = sns.cubehelix_palette(start=1.6, light=0.8, as_cmap=True, reverse=True)
sns.heatmap(carrier_hmap_OO,annot=True,fmt='.1f',xticklabels=xticks,cmap=cmap)

plt.show()
```



Observation

1. The peak season for air travel in USA is considered to be June to August
2. Lean season is mid of January to February.
3. The airlines operate highest number of flights and carry maximum PAX load during the summer season and vis-a-versa during lean season.
4. The data proves that the statement is true and most of the airlines having maximum departure between May to August and minimum between January to February.
5. From, the heatmap, it is visible that during May to August most of the airlines tend to fly faster than normal flight speed, to cover maximum departure. Whereas, it is vis-a-versa during lean season.')

10. Conclusion

1. The dataset doesn't offer reasons for delays and missing important data such as latitude and longitude, flight diversion, chocks on and off timing, and fuel consumption.
2. So, it is clear that the dataset doesn't provide clear understanding of delay issues, which may be supportive to look into delays that can be controlled or reduced.
3. For example: If airlines permits pilots to fly aircraft at higher speed and fuel consumption on planes that departed late, the delay spread can be minimize along the flight network. This would decrease the possible delay itself and significatively reduced the number of aircraft delays.
4. A solution applicable to one type of delay may affect the others, resulting in a ripple effect that will allow more efficient operations; benefiting passengers, airports, carriers and even the world as a whole.

THE END