Part I - (US Flights (2006-2007) Delay Exploration)

by (Yaninthé Tiomene)

Introduction

> Have you ever traveled with an airplane? You may have experienced one of the inevitable pains of flying: the delays/Cancellation. Sometimes your plane arrives late, other times there may be a queue for takeoff, occasionally the weather (carrier, or securities reasons) forces long hours delays (or even cancellation); regardless of the reason for the delay, they pose a huge inconvenience for travelers. Someone may wonder if He had Data to investigate and find out reasons and how to predict these delays. This is the subject that I hope to explore and potentially find out valuable insights from the availabe data.

The dataset that will be used is from Dataverse and it reports flights in the United States, including carriers, arrival and departure delays from 1987 to 2008. But due to my CPU and GPU capacity, I will only consider data for 2006, and 2007. We could have considered flights data of 2008, but it only has records up to April of 2008, as the hole dataset was published in October of 2008. If we want to compare flights per month or per year the result may be biased.

Preliminary Wrangling

In [1]:
```
# import all packages and set plots to be embedded inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

%matplotlib inline

import time
import os
from pyDataverse.api import NativeApi, DataAccessApi
from pyDataverse.models import Dataverse

import warnings
warnings.simplefilter("ignore")
```

> Connecting to Harvard Dataverse API

In [2]:
```
#Connecting to flights data from dataverse API

base_url = 'https://dataverse.harvard.edu/'
api_token = 'ACCESS TOKEN'
api = NativeApi(base_url, api_token)
data_api = DataAccessApi(base_url)

DOI = "doi:10.7910/DVN/HG7NV7"
dataset = api.get_dataset(DOI)
```

In [3]:
```
files_list = dataset.json()['data']['latestVersion']['files']

files_list[-4:-3]
```

Out[3]:
```
[{'description': 'Describes the locations of US airports, with the fields: iata: the int
ernational airport abbreviation code \r name of the airport \r city and country in which
airport is located. \r lat and long: the latitude and longitude of the airport \r ',
  'label': 'airports.csv',
  'restricted': False,
  'version': 1,
  'datasetVersionId': 62280,
  'categories': ['1. Documentation'],
  'dataFile': {'id': 1374930,
   'persistentId': 'doi:10.7910/DVN/HG7NV7/XTPZZY',
   'pidURL': 'https://doi.org/10.7910/DVN/HG7NV7/XTPZZY',
   'filename': 'airports.csv',
   'contentType': 'text/plain; charset=US-ASCII',
   'filesize': 244438,
   'description': 'Describes the locations of US airports, with the fields: iata: the in
ternational airport abbreviation code \r name of the airport \r city and country in whic
h airport is located. \r lat and long: the latitude and longitude of the airport \r ',
   'storageIdentifier': 's3://dvn-cloud:16978',
   'rootDataFileId': -1,
   'md5': '7fbeda0239bed45f04a67763ac189a04',
   'checksum': {'type': 'MD5', 'value': '7fbeda0239bed45f04a67763ac189a04'},
   'creationDate': '2008-10-06'}}]
```

In [4]:
```
#Function that will help to download data from dataverse API and return it as a datafram

def get_data(file_range_start=0, file_range_end=-4):
    df = pd.DataFrame()

    count = 1
    start = time.time()
    print('count | file_id        | filename     |')

    for file in files_list[:-4]:
        filename = file["dataFile"]["filename"]
        file_id = file["dataFile"]["id"]
        print("{}     | {}        | {} |".format(count, file_id, filename))

        response = data_api.get_datafile(file_id)
        with open(filename, "wb") as f:
            f.write(response.content)
            try:
                df = pd.concat([df,pd.read_csv(filename)], ignore_index=True)
            except:
                df = pd.concat([df, pd.read_csv(filename, encoding='latin-1')], ignore_i

        count+=1

    end = time.time()
    print(end - start)

    return df
```

In [5]:
```
# # Downloading and saving flights details for 4 years (2006 - 2008)

#df_flight = get_data(file_range_start=0, file_range_end=-4)


flight = pd.DataFrame()

count = 1
start = time.time()
```

```
basepath='flight_files/'
for file in os.listdir(basepath):
    if os.path.isfile(os.path.join(basepath, file)):
        with open(os.path.join(basepath, file), "r") as f:
            print("{}     | {} |".format(count, file))

            try:
                df = pd.read_csv(os.path.join(basepath, file), low_memory=False)
                #df = df[~(df.DepTime.isnull())]
                #df = df[~(df.ArrTime.isnull())]
                flight = pd.concat([flight, df], ignore_index=True)
            except:
                df = pd.read_csv(os.path.join(basepath, file), encoding='latin-1', low_m
                #df = df[~(df.DepTime.isnull())]
                #df = df[~(df.ArrTime.isnull())]
                flight = pd.concat([flight, df], ignore_index=True)


    count+=1

end = time.time()
print(end - start)
flight.head()
```

```
1     | 2007.csv.bz2 |
2     | 2006.csv.bz2 |
127.14789819717407
```

Out[5]:

	Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarrier	F
0	2007	1	1	1	1232.0	1225	1341.0	1340	WN	
1	2007	1	1	1	1918.0	1905	2043.0	2035	WN	
2	2007	1	1	1	2206.0	2130	2334.0	2300	WN	
3	2007	1	1	1	1230.0	1200	1356.0	1330	WN	
4	2007	1	1	1	831.0	830	957.0	1000	WN	

5 rows × 29 columns

In [6]:
```
#saving dataframes fligths into a csv file
flight.to_csv('flights.csv', index=False)
```

In [7]:
```
# # Downloading variable descriptions
df_desc=pd.read_csv('variable-descriptions.csv')
df_desc
```

Out[7]:

	Name	Description
1	Year	1987-2008
2	Month	12-Jan
3	DayofMonth	31-Jan
4	DayOfWeek	1 (Monday) - 7 (Sunday)
5	DepTime	actual departure time (local, hhmm)
6	CRSDepTime	scheduled departure time (local, hhmm)
7	ArrTime	actual arrival time (local, hhmm)
8	CRSArrTime	scheduled arrival time (local, hhmm)
9	UniqueCarrier	unique carrier code

10	FlightNum	flight number
11	TailNum	plane tail number
12	ActualElapsedTime	in minutes
13	CRSElapsedTime	in minutes
14	AirTime	in minutes
15	ArrDelay	arrival delay, in minutes
16	DepDelay	departure delay, in minutes
17	Origin	origin IATA airport code
18	Dest	destination IATA airport code
19	Distance	in miles
20	TaxiIn	taxi in time, in minutes
21	TaxiOut	taxi out time in minutes
22	Cancelled	was the flight cancelled?
23	CancellationCode	reason for cancellation (A = carrier, B = weat...
24	Diverted	1 = yes, 0 = no
25	CarrierDelay	in minutes
26	WeatherDelay	in minutes
27	NASDelay	in minutes
28	SecurityDelay	in minutes
29	LateAircraftDelay	in minutes

In [8]:
```
# high-level overview of data shape and composition
print(flight.shape)
print(flight.dtypes)
print(flight.head(10))
```

```
(14595137, 29)
Year                   int64
Month                  int64
DayofMonth             int64
DayOfWeek              int64
DepTime              float64
CRSDepTime             int64
ArrTime              float64
CRSArrTime             int64
UniqueCarrier         object
FlightNum              int64
TailNum               object
ActualElapsedTime    float64
CRSElapsedTime       float64
AirTime              float64
ArrDelay             float64
DepDelay             float64
Origin                object
Dest                  object
Distance               int64
TaxiIn                 int64
TaxiOut                int64
Cancelled              int64
CancellationCode      object
Diverted               int64
```

```
CarrierDelay           int64
WeatherDelay           int64
NASDelay               int64
SecurityDelay          int64
LateAircraftDelay      int64
dtype: object
   Year  Month  DayofMonth  DayOfWeek  DepTime  CRSDepTime  ArrTime  \
0  2007      1           1          1   1232.0        1225   1341.0
1  2007      1           1          1   1918.0        1905   2043.0
2  2007      1           1          1   2206.0        2130   2334.0
3  2007      1           1          1   1230.0        1200   1356.0
4  2007      1           1          1    831.0         830    957.0
5  2007      1           1          1   1430.0        1420   1553.0
6  2007      1           1          1   1936.0        1840   2217.0
7  2007      1           1          1    944.0         935   1223.0
8  2007      1           1          1   1537.0        1450   1819.0
9  2007      1           1          1   1318.0        1315   1603.0

   CRSArrTime UniqueCarrier  FlightNum  ... TaxiIn  TaxiOut  Cancelled  \
0        1340            WN       2891  ...      4       11          0
1        2035            WN        462  ...      5        6          0
2        2300            WN       1229  ...      6        9          0
3        1330            WN       1355  ...      3        8          0
4        1000            WN       2278  ...      3        9          0
5        1550            WN       2386  ...      2        7          0
6        2130            WN        409  ...      5        7          0
7        1225            WN       1131  ...      4        9          0
8        1735            WN       1212  ...      5        7          0
9        1610            WN       2456  ...      5        8          0

   CancellationCode  Diverted  CarrierDelay WeatherDelay NASDelay  \
0               NaN         0             0            0        0
1               NaN         0             0            0        0
2               NaN         0             3            0        0
3               NaN         0            23            0        0
4               NaN         0             0            0        0
5               NaN         0             0            0        0
6               NaN         0            46            0        0
7               NaN         0             0            0        0
8               NaN         0            20            0        0
9               NaN         0             0            0        0

   SecurityDelay  LateAircraftDelay
0              0                  0
1              0                  0
2              0                 31
3              0                  3
4              0                  0
5              0                  0
6              0                  1
7              0                  0
8              0                 24
9              0                  0

[10 rows x 29 columns]
```

> **Observations**: As we can see `DepTime`, `CRSDepTime`, `ArrTime` and `CRSArrTime` variables are numerical data instead of being local hhmm as stated in the variable despcription dataframe. We should reformat them and created 2 new variables `DepDayTime` and `ArrDayTime` which extract the day time from Variables `DepTime` and `ArrTime` respectively.

In [9]:
```
#let's make a copy of the dataframe

df_flight = flight.copy()
```

In [10]:
```
#let's format DayOfMonth's name to fit the other variable Format name
df_flight.rename(columns={'DayofMonth':'DayOfMonth'}, inplace=True)

df_flight.columns
```

Out[10]:
```
Index(['Year', 'Month', 'DayOfMonth', 'DayOfWeek', 'DepTime', 'CRSDepTime',
       'ArrTime', 'CRSArrTime', 'UniqueCarrier', 'FlightNum', 'TailNum',
       'ActualElapsedTime', 'CRSElapsedTime', 'AirTime', 'ArrDelay',
       'DepDelay', 'Origin', 'Dest', 'Distance', 'TaxiIn', 'TaxiOut',
       'Cancelled', 'CancellationCode', 'Diverted', 'CarrierDelay',
       'WeatherDelay', 'NASDelay', 'SecurityDelay', 'LateAircraftDelay'],
      dtype='object')
```

In [11]:
```
# Formating times variables
df_flight['DepTime'] = df_flight.DepTime.apply(lambda x: str(int(x)).zfill(4) if pd.notn
```

In [12]:
```
df_flight['CRSDepTime'] = df_flight.CRSDepTime.apply(lambda x: str(int(x)).zfill(4) if p
```

In [13]:
```
df_flight['ArrTime'] = df_flight.ArrTime.apply(lambda x: str(int(x)).zfill(4) if pd.notn
```

In [14]:
```
df_flight['CRSArrTime'] = df_flight.CRSArrTime.apply(lambda x: str(int(x)).zfill(4) if p

df_flight.head()
```

Out[14]:

	Year	Month	DayOfMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarrier	F
0	2007	1	1	1	1232	1225	1341	1340	WN	
1	2007	1	1	1	1918	1905	2043	2035	WN	
2	2007	1	1	1	2206	2130	2334	2300	WN	
3	2007	1	1	1	1230	1200	1356	1330	WN	
4	2007	1	1	1	0831	0830	0957	1000	WN	

5 rows × 29 columns

In [15]:
```
df_flight[['DepTime', 'ArrTime']].isnull().sum()
```

Out[15]:
```
DepTime    282682
ArrTime    316047
dtype: int64
```

> **Observations**:The number of missing values in `DepTime` and `ArrTime` are close so they might be in the same rows. I suspect that these missing values are from `Cancelled` and/or `Diverted` flights. let's check value_count for these too.

In [16]:
```
df_flight['Cancelled'].value_counts()
```

Out[16]:
```
0    14312455
1      282682
Name: Cancelled, dtype: int64
```

In [17]:
```
df_flight[df_flight.Cancelled == 1][['DepTime', 'ArrTime']].isnull().sum()
```

Out[17]:
```
DepTime    282682
ArrTime    282682
```

```
dtype: int64
```

In [18]:
```
df_flight['Diverted'].value_counts()
```

Out[18]:
```
0    14561772
1       33365
Name: Diverted, dtype: int64
```

In [19]:
```
df_flight[df_flight.Diverted == 1][['DepTime', 'ArrTime']].isnull().sum()
```

Out[19]:
```
DepTime        0
ArrTime    33365
dtype: int64
```

> **Observation** : As suspected, Cancelled flights are flight with neither Departure nor Arrival time, And diverted flights are flights with no Arrival Time.

In [20]:
```
df_flight[df_flight.DepTime.str[:2].astype(float) > 24].head()
```

Out[20]:

	Year	Month	DayOfMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCa
7753059	2006	1	13	5	2755	2023	2923	2219	
7753876	2006	1	2	1	2509	1844	2657	2047	
7754049	2006	1	2	1	2508	2250	2609	0007	
7755020	2006	1	2	1	2525	2250	2530	2307	
7755040	2006	1	22	7	2515	2250	2528	2307	

5 rows × 29 columns

In [21]:
```
df_flight[df_flight.DepTime.str[:2].astype(float) > 24].CRSDepTime.min()
```

Out[21]:
```
'0155'
```

In [22]:
```
df_flight[df_flight.ArrTime.str[:2].astype(float) > 24].CRSArrTime.min()
```

Out[22]:
```
'0001'
```

There are some flights that were schedule in the evening and departed or arrive late than midnight. Actual departure time and arrival time for these flights have unqualified values like 2505, 2610.

In [23]:
```
# creating AM/PM feature into df_flight
def setDayTime(x):
    if x == np.nan:
        return x
    elif((x>=12.0) & (x<= 23.0)):
        return 'PM'
    elif((x>=0.0) & (x<= 11.0) | (x==24.0)):
        return 'AM'
    else:
        return x
```

In [24]:
```
df_flight['DepDayTime'] = df_flight.CRSDepTime.str[:2].astype(float)
df_flight['DepDayTime'] = df_flight.DepDayTime.apply(setDayTime)
```

In [25]:
```
df_flight['ArrDayTime'] = df_flight.CRSArrTime.str[:2].astype(float)
df_flight['ArrDayTime'] = df_flight.ArrDayTime.apply(setDayTime)
```

In [26]:
```
df_flight.DepDayTime.value_counts()
```

Out[26]:
```
PM    8629692
AM    5965445
Name: DepDayTime, dtype: int64
```

In [27]:
```
df_flight.ArrDayTime.value_counts()
```

Out[27]:
```
PM    10298635
AM     4296502
Name: ArrDayTime, dtype: int64
```

In [28]:
```
#Let's change the Type of Month to an ordered object type
def setMonth(x):
    months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov
    if x==np.nan:
        return x
    else:
        return months[x-1]

df_flight['Month'] = df_flight.Month.apply(setMonth)
```

In [29]:
```
#Let's change the Type of DayOfWeek to an ordered object type
def setDay(x):
    days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
    if x==np.nan:
        return x
    else:
        return days[x-1]

df_flight['DayOfWeek'] = df_flight.DayOfWeek.apply(setDay)
```

In [30]:
```
# convert Year, Month, DayOfMonth, and DayOfWeek into ordered categorical types
ordinal_var_dict = {'Year': [2006, 2007],
                    'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'S
                    'DayOfMonth': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
                    'DayOfWeek': ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']}

for var in ordinal_var_dict:
    ordered_var = pd.api.types.CategoricalDtype(ordered = True,
                                                categories = ordinal_var_dict[var])
    df_flight[var] = df_flight[var].astype(ordered_var)
```

In [31]:
```
# Let's categorize departure delay and arrival delay.
df_flight['DepDelay_cat'] = pd.cut(df_flight['DepDelay'], bins=[-5000, -30, 0, 30, 300,
                                   labels=['Early >30mins', 'Early <=30mins', 'Late <=30


df_flight['ArrDelay_cat'] = pd.cut(df_flight['ArrDelay'], bins=[-5000, -30, 0, 30, 300,
                                   labels=['Early >30mins', 'Early <=30mins', 'Late <=30
```

In [ ]:
```
```

In [32]:
```
df_flight.head()
```

Out[32]:

	Year	Month	DayOfMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarrier	F
0	2007	Jan	1	Mon	1232	1225	1341	1340	WN	
1	2007	Jan	1	Mon	1918	1905	2043	2035	WN	
2	2007	Jan	1	Mon	2206	2130	2334	2300	WN	

3	2007	Jan	1	Mon	1230	1200	1356	1330	WN
4	2007	Jan	1	Mon	0831	0830	0957	1000	WN

5 rows × 33 columns

In [33]:
```
# descriptive statistics for numeric variables
print(df_flight.describe())
```

```
          FlightNum  ActualElapsedTime  CRSElapsedTime       AirTime  \
count  1.459514e+07       1.427909e+07    1.459414e+07  1.427909e+07
mean   2.187446e+03       1.261937e+02    1.272180e+02  1.028349e+02
std    1.980504e+03       7.124866e+01    7.033691e+01  7.238262e+01
min    1.000000e+00       5.000000e+00   -1.240000e+03 -1.425000e+03
25%    5.870000e+02       7.500000e+01    7.700000e+01  5.400000e+01
50%    1.501000e+03       1.080000e+02    1.090000e+02  8.400000e+01
75%    3.499000e+03       1.560000e+02    1.570000e+02  1.310000e+02
max    9.619000e+03       1.879000e+03    1.430000e+03  1.958000e+03

           ArrDelay      DepDelay      Distance        TaxiIn       TaxiOut  \
count  1.427909e+07  1.431246e+07  1.459514e+07  1.459514e+07  1.459514e+07
mean   9.451859e+00  1.075882e+01  7.238142e+02  6.872862e+00  1.602864e+01
std    3.800010e+01  3.487780e+01  5.683351e+02  2.208140e+01  1.156531e+01
min   -5.920000e+02 -1.200000e+03  1.100000e+01  0.000000e+00  0.000000e+00
25%   -9.000000e+00 -4.000000e+00  3.170000e+02  4.000000e+00  1.000000e+01
50%   -1.000000e+00  0.000000e+00  5.690000e+02  5.000000e+00  1.300000e+01
75%    1.300000e+01  1.000000e+01  9.510000e+02  8.000000e+00  1.900000e+01
max    2.598000e+03  2.601000e+03  4.962000e+03  1.501000e+03  6.020000e+02

          Cancelled      Diverted  CarrierDelay  WeatherDelay      NASDelay  \
count  1.459514e+07  1.459514e+07  1.459514e+07  1.459514e+07  1.459514e+07
mean   1.936823e-02  2.286035e-03  3.635713e+00  7.258234e-01  3.686945e+00
std    1.378155e-01  4.775782e-02  1.987083e+01  9.085721e+00  1.591085e+01
min    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
25%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
50%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
75%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
max    1.000000e+00  1.000000e+00  2.580000e+03  1.429000e+03  1.392000e+03

       SecurityDelay  LateAircraftDelay
count   1.459514e+07       1.459514e+07
mean    2.728834e-02       4.813299e+00
std     1.183342e+00       2.059818e+01
min     0.000000e+00       0.000000e+00
25%     0.000000e+00       0.000000e+00
50%     0.000000e+00       0.000000e+00
75%     0.000000e+00       0.000000e+00
max     3.820000e+02       1.366000e+03
```

Observations: From this stat description of numerical features, we can see that:

1. `DepDelay` which is the difference between the official departure time and the actual departure time of the flight measured in minutes, has a median of *0 minute* and a mean of around *11 minutes*. This suggests that more than 50% of the flight quit in time and that the average departure delay is only around 11 minutes.

2. `ArrDelay` which is the difference between the official arrival time and the actual arrival time of the flight measured in minutes, has a median of *-1 minutes* and a mean of around *9 minutes*. This

also tells us that more than 50% of flights in this dataset arrive earlier than scheduled and that the average arrival delay is only 9 minutes.

What is the structure of your dataset?

> The dataset consist of 14595137 entries and 33 features where majorities are numericals.

What is/are the main feature(s) of interest in your dataset?

> I'm mostly interrested in figuring out what are the features that may contribute to the cancellation or delay of the flight (departure or arrival).

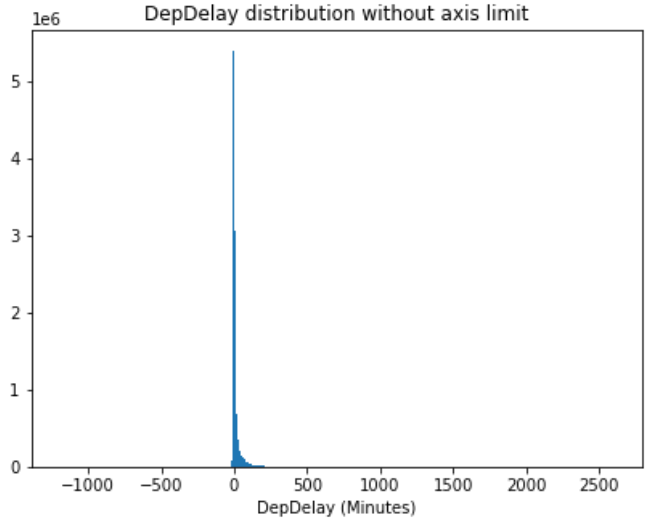
What features in the dataset do you think will help support your investigation into your feature(s) of interest?

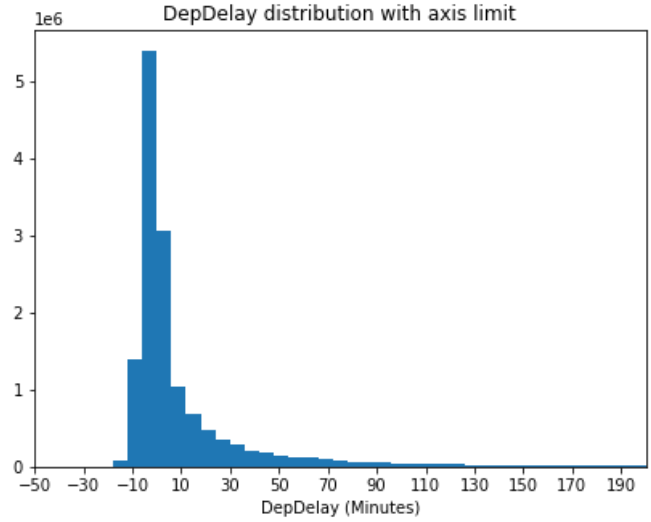
> I expect that `ArrDelay`, `DepDelay`, `Cancelled`, `Diverted`, `Year`, `Month`, `DayOfMonth`, `DayOfWeek`, `DepTime`, `ArrTime`, `UniqueCarrier`, `DepDayTime`, `ArrDayTime`, `Origin` & `Dest`, `Distance`, `DepDelay_cat` & `ArrDelay_cat` will support the investigation.

Univariate Exploration

> What is the distribution of `DepDelay` ?

In [34]:
```
# start with a standard-scaled plot
def distrib(variable='DepDelay'):
    binsize = 6
    bins = np.arange(df_flight[variable].min(), df_flight[variable].max()+binsize, binsi
    x_min, x_max = -50, 200

    plt.figure(figsize=[15, 5])
    plt.subplot(1,2,1)
    plt.hist(data = df_flight, x = variable, bins = bins)
    plt.xlabel(f'{variable} (Minutes)')
    plt.title(f'{variable} distribution without axis limit')


    plt.subplot(1,2,2)
    plt.hist(data = df_flight, x = variable, bins = bins)
    plt.xlabel(f'{variable} (Minutes)')
    plt.title(f'{variable} distribution with axis limit')
    plt.xlim(x_min,x_max)
    plt.xticks(np.arange(x_min, x_max, binsize+14))

    plt.show()

distrib(variable='DepDelay')
```

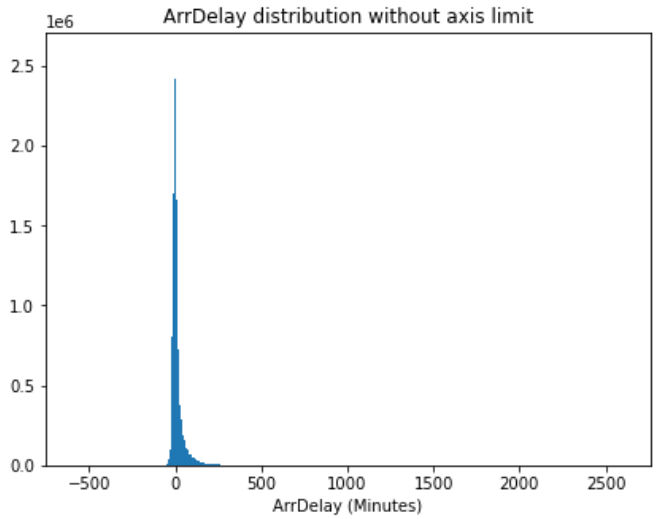


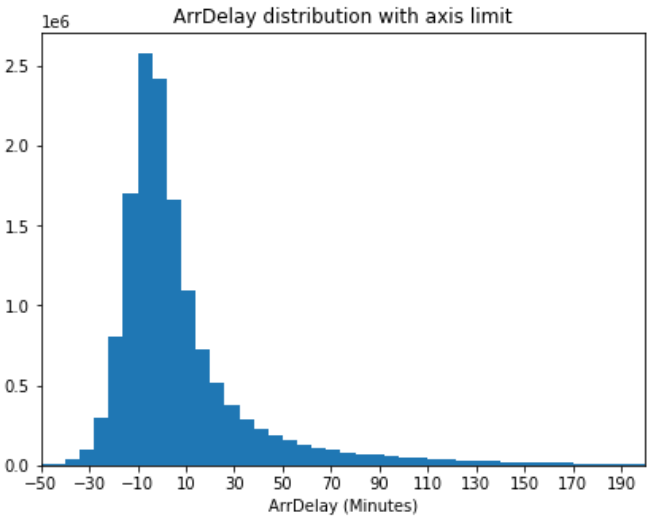


> **Observation**:It can be seen on the histogram, that departure delays are mostly located on the left side of the graph, with a long tail to the right. The majority of departure delays are short (earlier flights and departure flights delay between -20 and 70 minutes) , and the longest delays, while unusual, are more heavy loaded in time.

> what is the distribution of `ArrDelay` ?

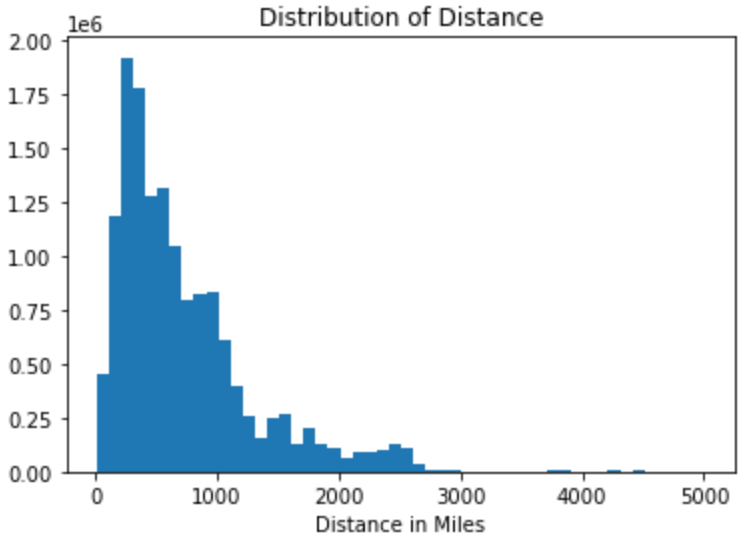
In [35]:
```
# start with a standard-scaled plot
distrib(variable='ArrDelay')
```





> **Observation**:It can be seen on the histogram, that Arrival delays are mostly located on the left side of the graph, nearest to a normal distribution, but skew to the right.
>
> Distribution of Distance

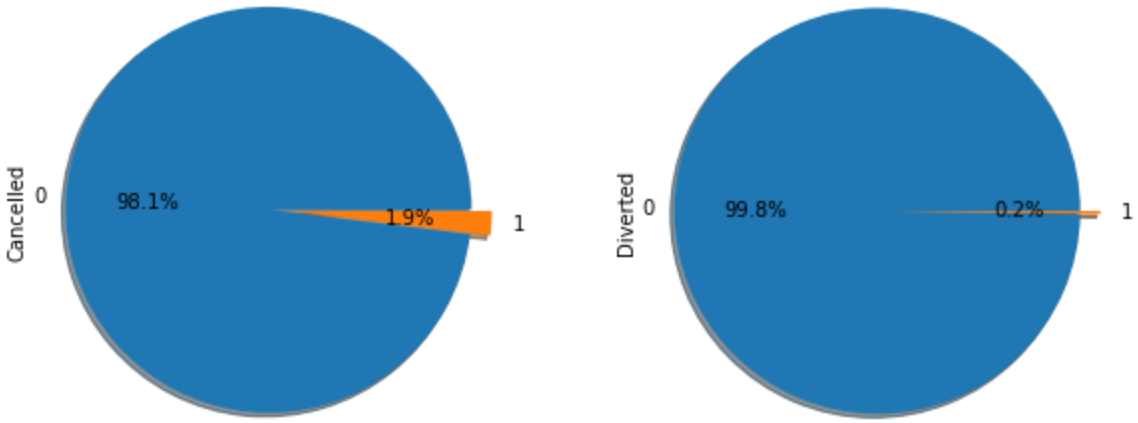
In [36]:
```
binsize = 100
bins = np.arange(df_flight['Distance'].min(), df_flight['Distance'].max()+binsize, binsi

plt.hist(data = df_flight, x = 'Distance', bins = bins)
plt.xlabel('Distance in Miles')
plt.title('Distribution of Distance')
plt.show()
```



Observations: `Distance` is unimodal, and skewed right.

> Distribution of `Cancelled` and `Diverted` flights?

In [37]:
```
# let's plot all 2 together to get an idea of each variable's distribution.

fig, ax = plt.subplots(ncols=2, figsize = [10,10])

df_flight.Cancelled.value_counts().plot.pie(explode=[0.1,0],autopct='%1.1f%%',ax=ax[0],s

df_flight.Diverted.value_counts().plot.pie(explode=[0.1,0],autopct='%1.1f%%',ax=ax[1],sh

plt.show()
```

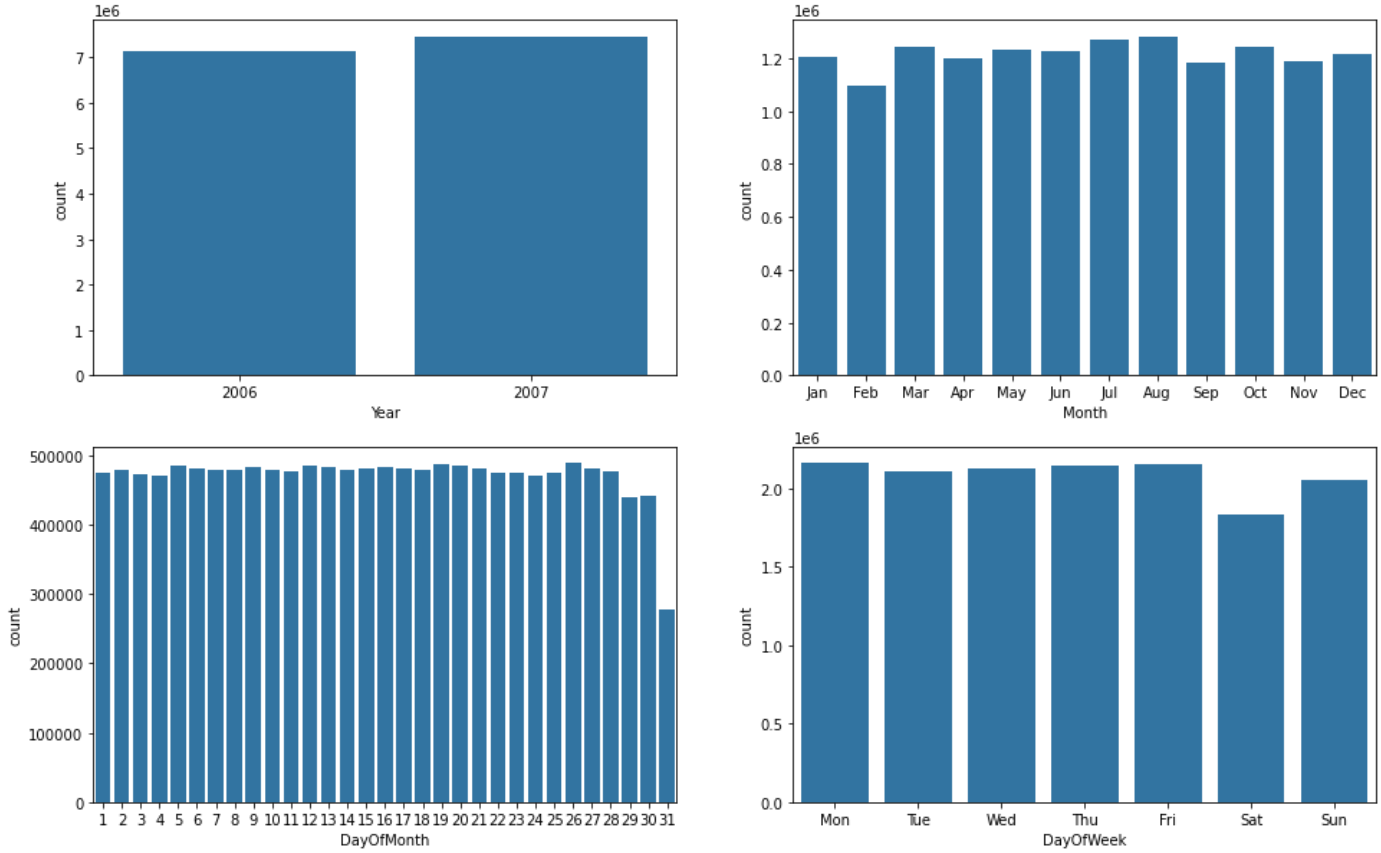


> **Observations**: From this plot, we can see that only 1.9% of flights were Cancelled, while 0.2% were Diverted out of 14595137 flights

> Distributions of fligths per `Year`, `Month` and `DayOfWeek`?

In [38]:
```
# let's plot all three together to get an idea of each ordinal variable's distribution.

fig, ax = plt.subplots(nrows=2, ncols=2, figsize = [16,10])
```

```
default_color = sb.color_palette()[0]
sb.countplot(data = df_flight, x = 'Year', color = default_color, ax = ax[0][0])
sb.countplot(data = df_flight, x = 'Month', color = default_color, ax = ax[0][1])
sb.countplot(data = df_flight, x = 'DayOfMonth', color = default_color, ax = ax[1][0])
sb.countplot(data = df_flight, x = 'DayOfWeek', color = default_color, ax = ax[1][1])

plt.show()
```



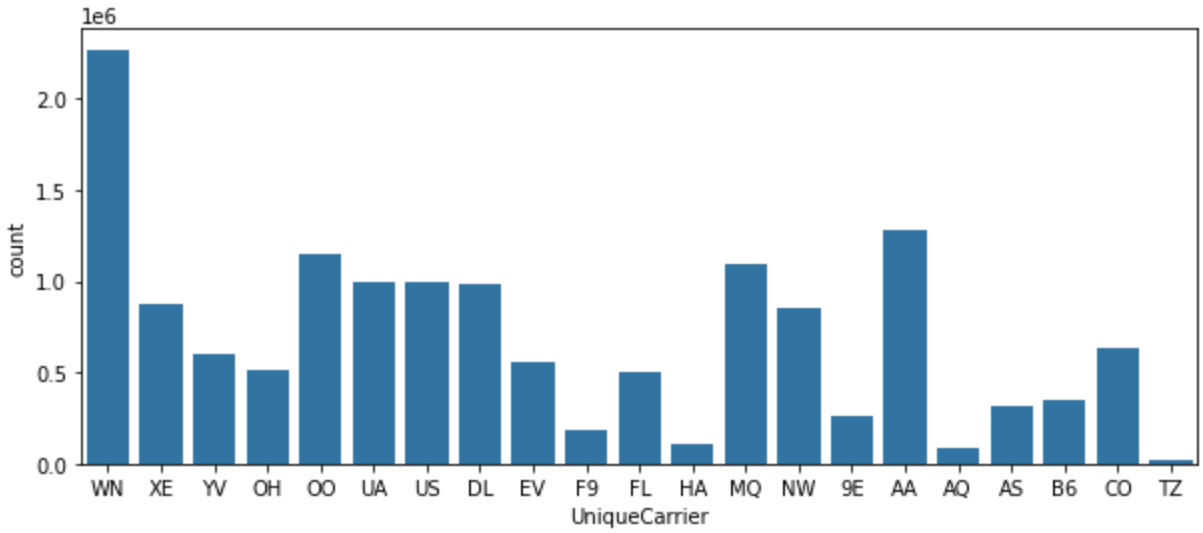
Observations:

1. Number of flights increases a litle bit from 2006 to 2007.
2. There are not big differences amongst number of flights per months, but we can see that Jully and August (summer) are a litle bit higher than other months and that in September the number of flight decreases. February is the smallest month with number of flights as expected due to less number of days compare to other months.
3. Flights per days of months are pretty the same except for the last three days of longest months (29, 30 and 31) where we have less flights. This is normal because month of February in 2006 and 2007 has only 28 days.
4. During the week days there are more flights than the Weekend. Saturday is the less busy flight day, it will be interresting to undesterdant why.

> Distribution of UniqueCarrier flights?

In [39]:
```
#Ploting distibution of UniqueCarrier

plt.figure(figsize=[10, 4])
default_color = sb.color_palette()[0]
sb.countplot(data = df_flight, x = 'UniqueCarrier', color = default_color)
```

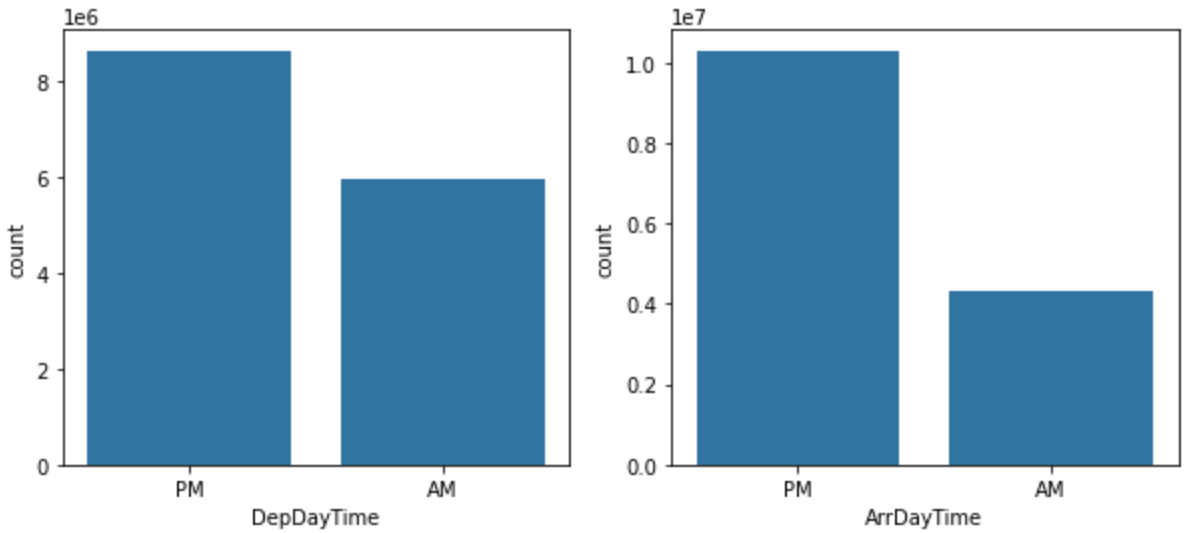
```
plt.show()
```



Observations: There are big difference amongst airlines, where **WN** is the carrier code with highest number of flight and **TZ** the carrier code with less number of flight.

> Distribution of flight scheduled departure and arrival per day time

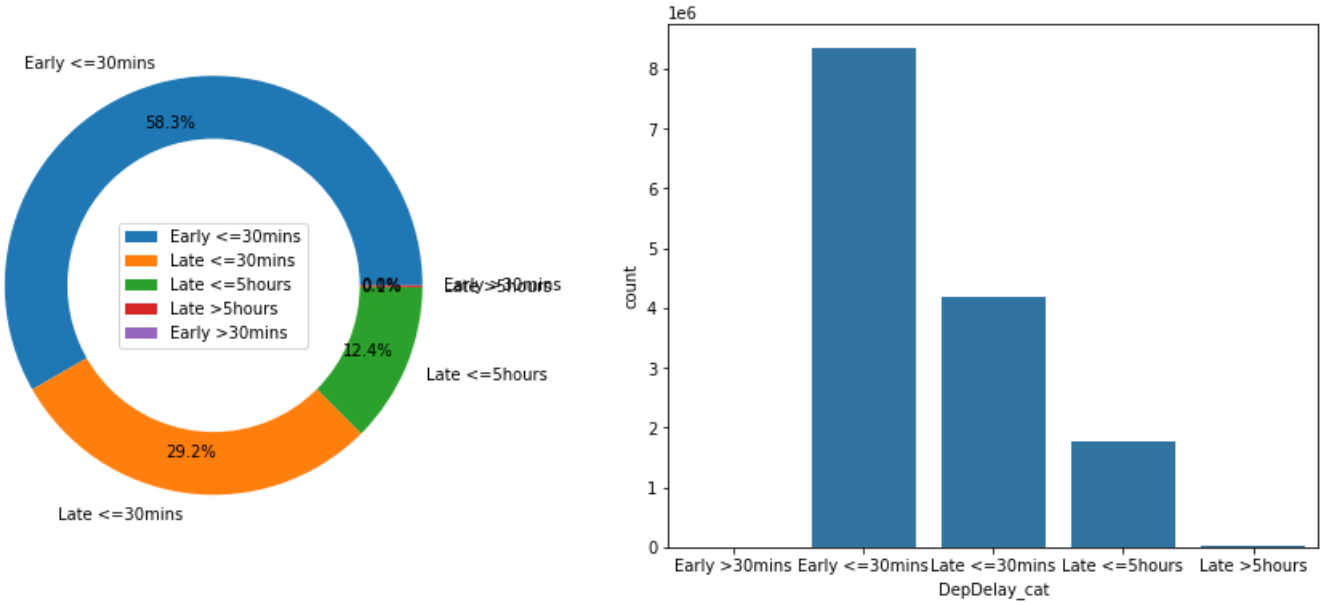
In [40]:
```
fig, ax = plt.subplots(ncols=2, figsize = [10,4])

default_color = sb.color_palette()[0]
sb.countplot(data = df_flight, x = 'DepDayTime', color = default_color, ax = ax[0])
sb.countplot(data = df_flight, x = 'ArrDayTime', color = default_color, ax = ax[1])

plt.show()
```



Observations: From these plots we can see that most of the flight are scheduled between midday and midnight.

In [41]:
```
delay_cat = df_flight.DepDelay_cat.value_counts().index

# Setting size in Chart based on
# given values
delay_count = df_flight.DepDelay_cat.value_counts()
```

```
plt.figure(figsize=[16,6])

plt.subplot(1,2,1)
# Pie Chart
plt.pie(delay_count, labels=delay_cat,
        autopct='%1.1f%%', pctdistance=.8)

# draw circle
centre_circle = plt.Circle((0, 0), 0.70, fc='white')
fig = plt.gcf()

# Adding Circle in Pie chart
fig.gca().add_artist(centre_circle)

plt.legend(labels=delay_cat, loc = 10)

plt.subplot(1,2,2)

sb.countplot(data = df_flight, x = 'DepDelay_cat', color = default_color)

plt.show()
```

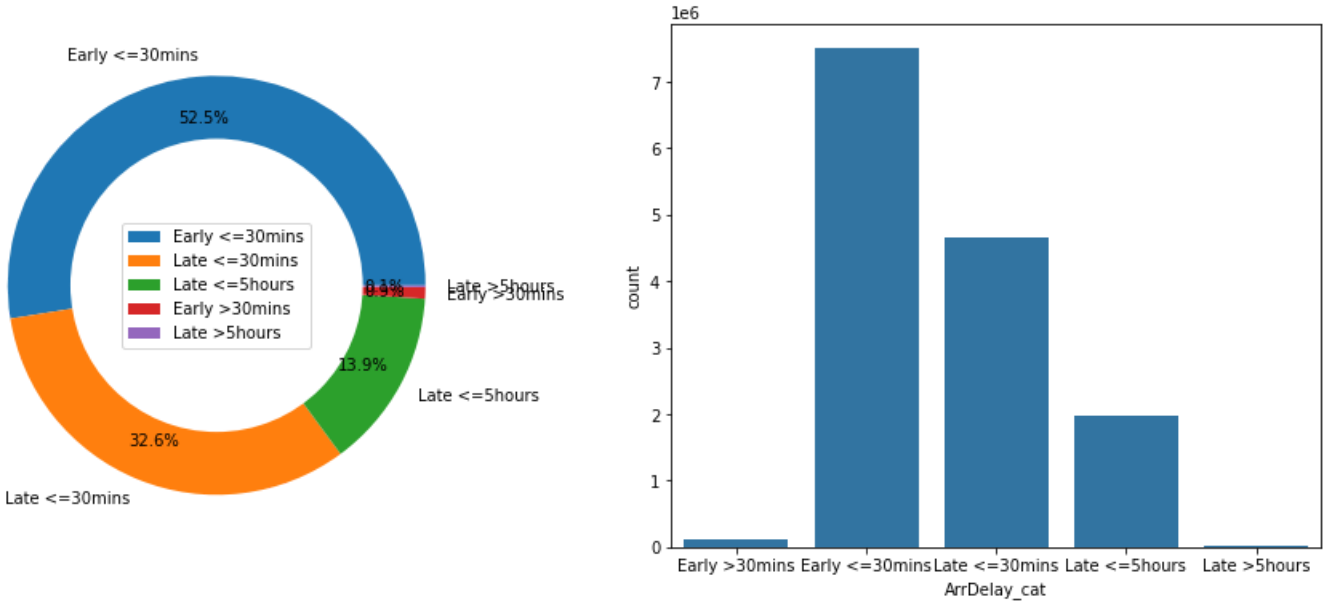


Observations: From these plots we can see that most of the flights (58.3%) departed less than 30 minutes earlier as shceduled and 29.2% departed less than 30 minutes later as scheduled.

In [42]:

```
delay_cat = df_flight.ArrDelay_cat.value_counts().index

# Setting size in Chart based on
# given values
delay_count = df_flight.ArrDelay_cat.value_counts()


plt.figure(figsize=[16,6])

plt.subplot(1,2,1)
# Pie Chart
plt.pie(delay_count, labels=delay_cat,
        autopct='%1.1f%%', pctdistance=.8)

# draw circle
centre_circle = plt.Circle((0, 0), 0.70, fc='white')
fig = plt.gcf()

# Adding Circle in Pie chart
```

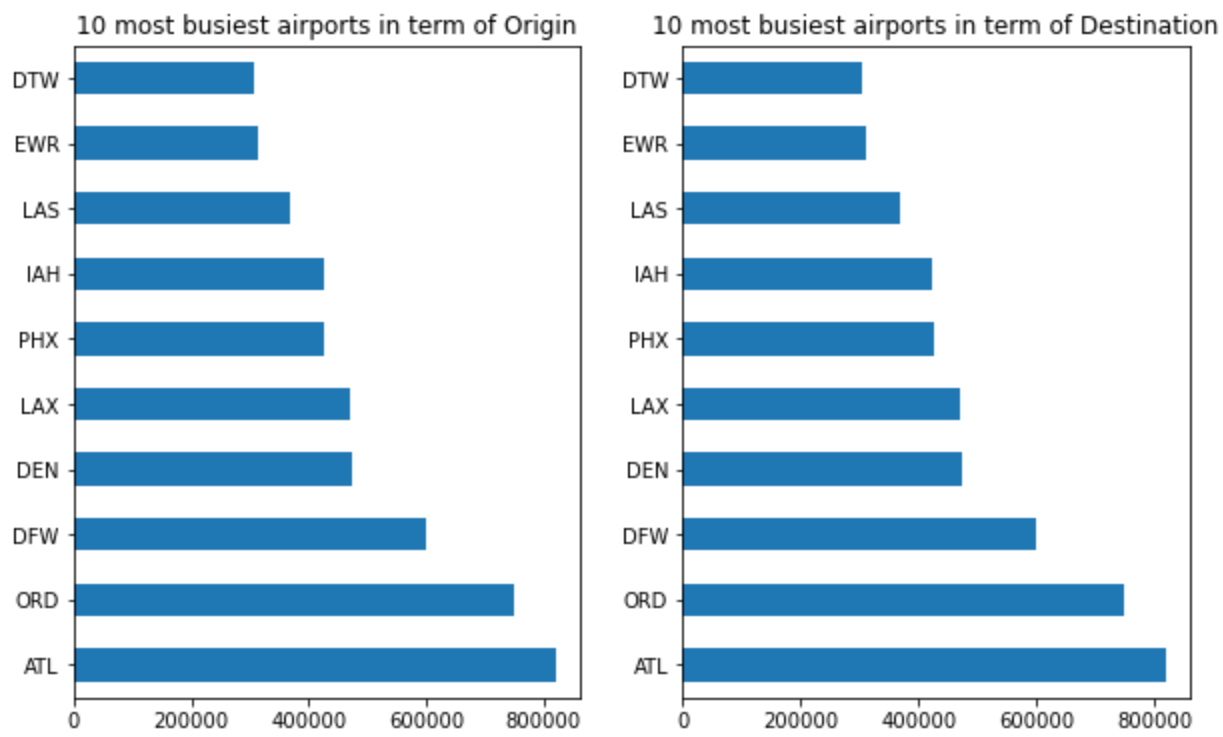
```
fig.gca().add_artist(centre_circle)

plt.legend(labels=delay_cat, loc = 10)

plt.subplot(1,2,2)

sb.countplot(data = df_flight, x = 'ArrDelay_cat', color = default_color)

plt.show()
```



Observations: As seen on the plot for departure, from these plots, most of the flights (52.5%) arrived less than 30 minutes earlier as scheduled and 32.6% arrived less than 30 minutes later as scheduled.

Let's plot the 10 most busiest airports in term of origin and destination

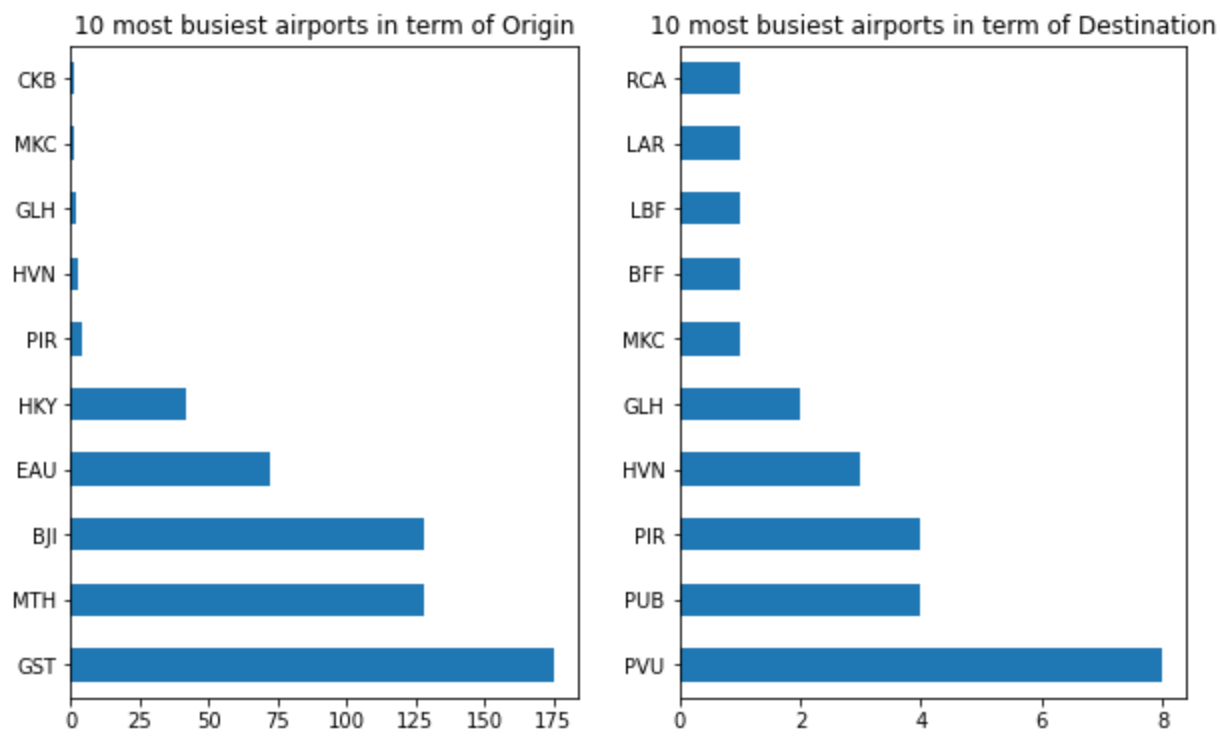
In [43]:
```
plt.figure(figsize=[10,6])

plt.subplot(1, 2, 1)
df_flight.Origin.value_counts(ascending=False).head(10).plot(kind='barh')
plt.title('10 most busiest airports in term of Origin')

plt.subplot(1, 2, 2)
df_flight.Dest.value_counts(ascending=False).head(10).plot(kind='barh')
plt.title('10 most busiest airports in term of Destination')

plt.show()
```



Observations: We can see that, we have same most business airport from origin and destination, with ATL being the busiest.

Let's check if it is still the same with less busiest one

In [44]:
```
plt.figure(figsize=[10,6])

plt.subplot(1, 2, 1)
df_flight.Origin.value_counts().tail(10).plot(kind='barh')
plt.title('10 most busiest airports in term of Origin')

plt.subplot(1, 2, 2)
df_flight.Dest.value_counts().tail(10).plot(kind='barh')
plt.title('10 most busiest airports in term of Destination')

plt.show()
```



Observations: The less busiest airport are differents for origin and destination. While CKB is the less busiest origin airport with only 1 flight, RCA, LAR, LBF, BFF and MKC are the less busiest destination airports with only 1 flight. It also appear that MKC and GLH are the only airport that appear amongst the less busiest airport in origin and destination.

Discuss the distribution(s) of your variable(s) of interest. Were there any unusual points? Did you need to perform any transformations?

> From the plots, We discover that departure and arrival delays are mostly located on the left side of the graph, with a long tail to the right. The majority of departure and arrival delays are short (earlier departures (and arrivals), 20 minutes before the scheduled time and late departure (and arrivals) less than 70 minutes), and the longest delays, while unusual, are more heavy loaded in time.

Of the features you investigated, were there any unusual distributions? Did you perform any operations on the data to tidy, adjust, or change the form of the data? If so, why did you do this?

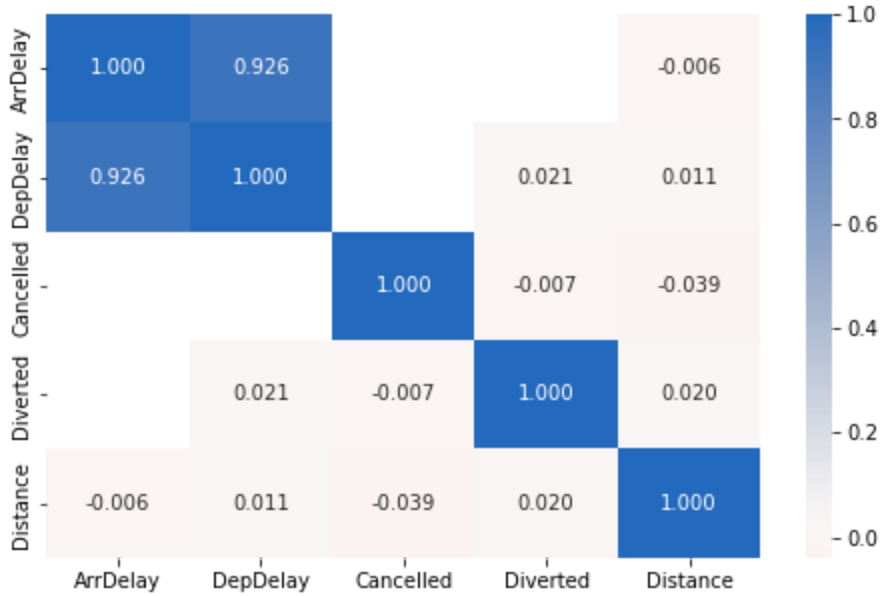
> While investigating the number of flight per `DayOfWeek`, we came out with something curious. While the number of flights per other days were at least equal to 2 millions, the number of flight of *saturday* were less than 2 millions and I couldn't see any justification about it. Still to investigate deeper.

Bivariate Exploration

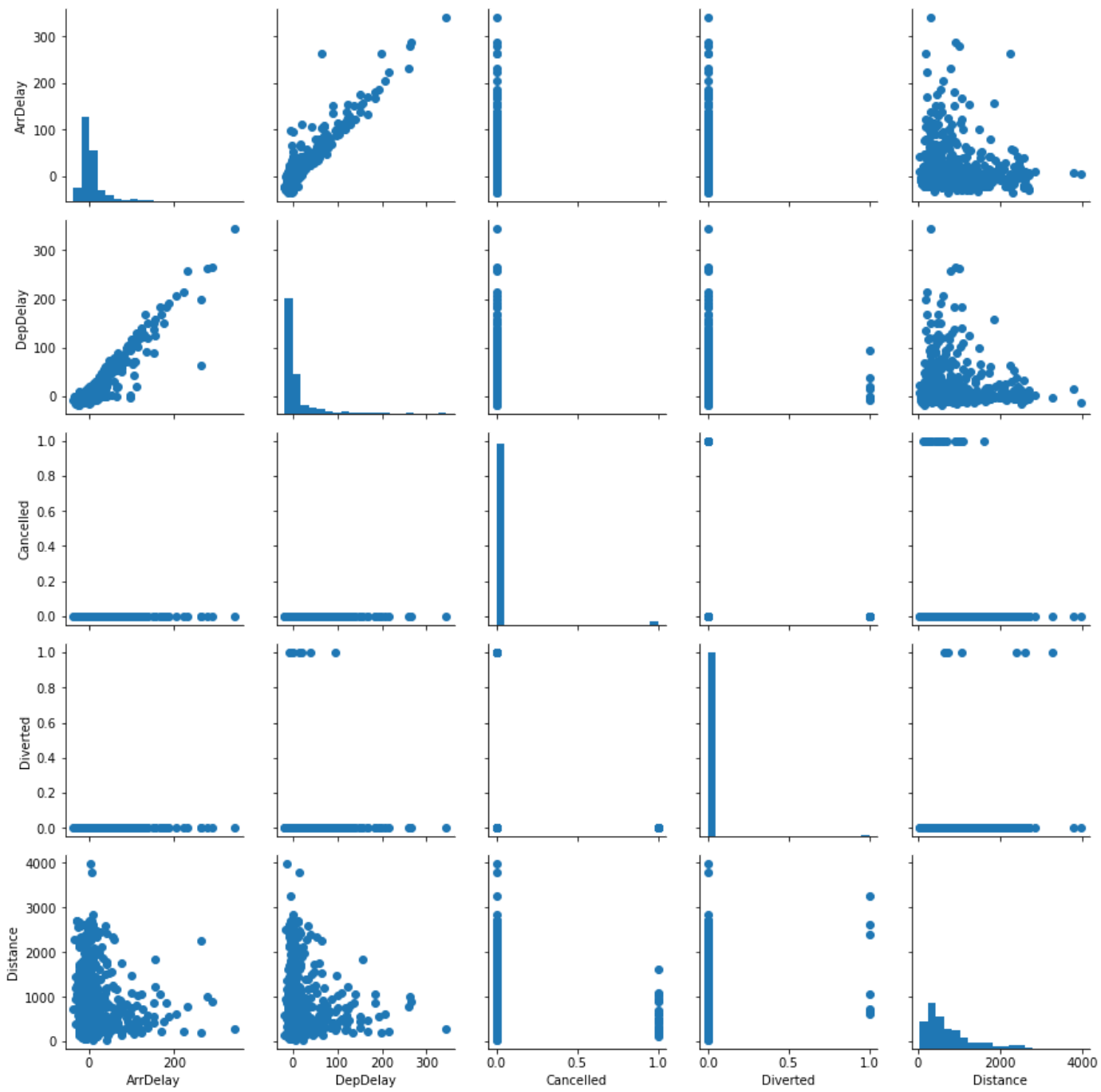
> Let's start off by looking correlations between numerical fetures in the dataset.

In [45]:
```
numeric_vars = ['ArrDelay', 'DepDelay', 'Cancelled', 'Diverted', 'Distance']
categoric_vars = ['Year', 'Month', 'DayOfMonth', 'DayOfWeek', 'UniqueCarrier', 'Origin',
                  'Dest', 'DepDayTime', 'ArrDayTime', 'DepDelay_cat', 'ArrDelay_cat']
```

In [46]:
```
# correlation plot
plt.figure(figsize = [8, 5])
sb.heatmap(df_flight[numeric_vars].corr(), annot = True, fmt = '.3f',
           cmap = 'vlag_r', center = 0)
plt.show()
```



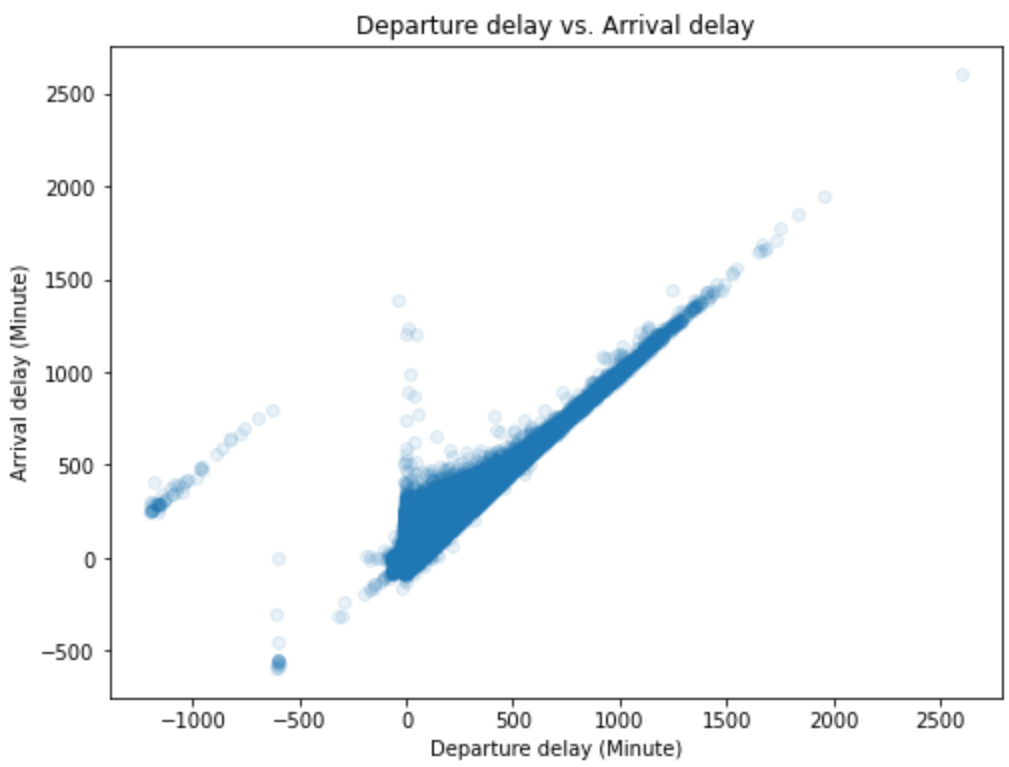
In [47]:
```
# plot matrix: sample 1000 flights so that plots are clearer and
# they render faster
samples = np.random.choice(df_flight.shape[0], 1000, replace = False)
flight_samp = df_flight.loc[samples,:]

g = sb.PairGrid(data = flight_samp, vars = numeric_vars)
g = g.map_diag(plt.hist, bins = 20);
g.map_offdiag(plt.scatter)

plt.show()
```



Observations: As expected ArrDelay and DepDelay are highly correlated with one another. Distance does not have any correlation with other variable. Cancelled and Diverted are discrete variable and as expected, they are not correlated neither with ArrDelay nor DepDelay. We will also check the relationship between delay and object variables.

Correlation between `ArrDelay` and `DepDelay`

In [48]:
```
plt.figure(figsize = [8, 6])
plt.scatter(data = df_flight, x = 'DepDelay', y = 'ArrDelay', alpha = 1/10)

plt.title('Departure delay vs. Arrival delay')

plt.xlabel('Departure delay (Minute)')

plt.ylabel('Arrival delay (Minute)')

plt.show()
```

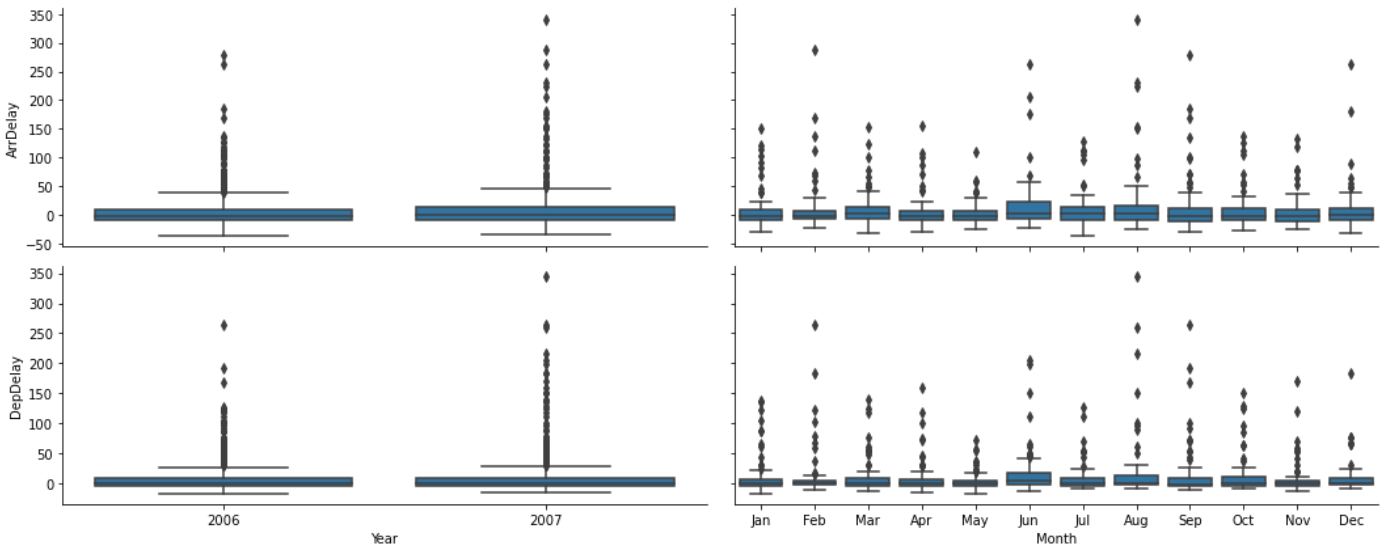


Observations: ArrDelay and DepDelay are highly correlated with one another. Which is what expected since a flight which is delayed at departure should normally have a delay at arrival. The higher the delay is at departure, the higher it should be at arrival. We can also see that there are earlier flight in departure (more than 16 hours earlier than scheduled) which are late at arrival (they might be outliers).

let's look at the correlation between `ArrDelay` , `DepDelay` and `Year` , `Month` , `DayOfWeek` , `DayOfMonth`

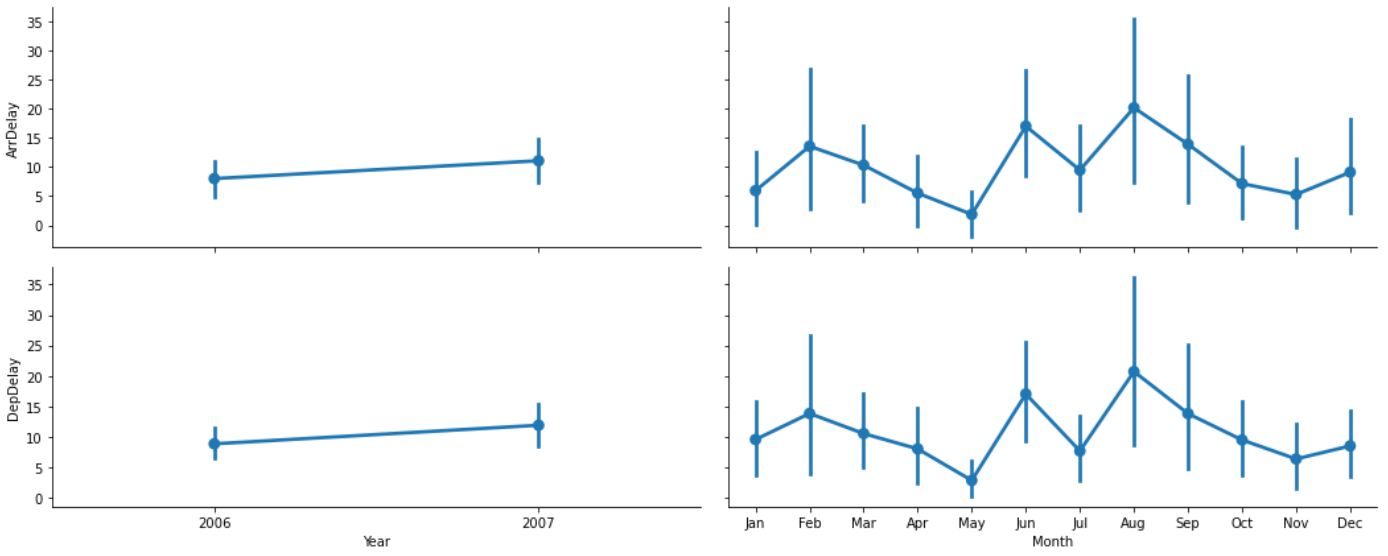
In [49]:
```
# plot matrix of numeric features against categorical features.
# can use a larger sample since there are fewer plots and they're simpler in nature.

categoric_vars = ['Year', 'Month']

samples = np.random.choice(df_flight.shape[0], 10000, replace = False)
fligtht_samp = df_flight.loc[samples,:]

def boxgrid(x, y, **kwargs):
    """ Quick hack for creating box plots with seaborn's PairGrid. """
    default_color = sb.color_palette()[0]
    sb.boxplot(data=flight_samp, x=x, y=y, color = default_color)

plt.figure(figsize = [20, 13]);
g = sb.PairGrid(data = flight_samp, y_vars = ['ArrDelay', 'DepDelay'], x_vars = categori
                height = 3, aspect = 2.5)
g.map(boxgrid)
plt.show()
```

```
<Figure size 1440x936 with 0 Axes>
```



In [50]:
```
# plot matrix of numeric features against categorical features.
# can use a larger sample since there are fewer plots and they're simpler in nature.

categoric_vars = ['Year', 'Month']

samples = np.random.choice(df_flight.shape[0], 10000, replace = False)
fligtht_samp = df_flight.loc[samples,:]

def pointgrid(x, y, **kwargs):
    """ Quick hack for creating box plots with seaborn's PairGrid. """
    default_color = sb.color_palette()[0]
    sb.pointplot(data=flight_samp, x=x, y=y, color = default_color)

plt.figure(figsize = [20, 13]);
g = sb.PairGrid(data = flight_samp, y_vars = ['ArrDelay', 'DepDelay'], x_vars = categori
                height = 3, aspect = 2.5)
g.map(pointgrid)
plt.show()
```

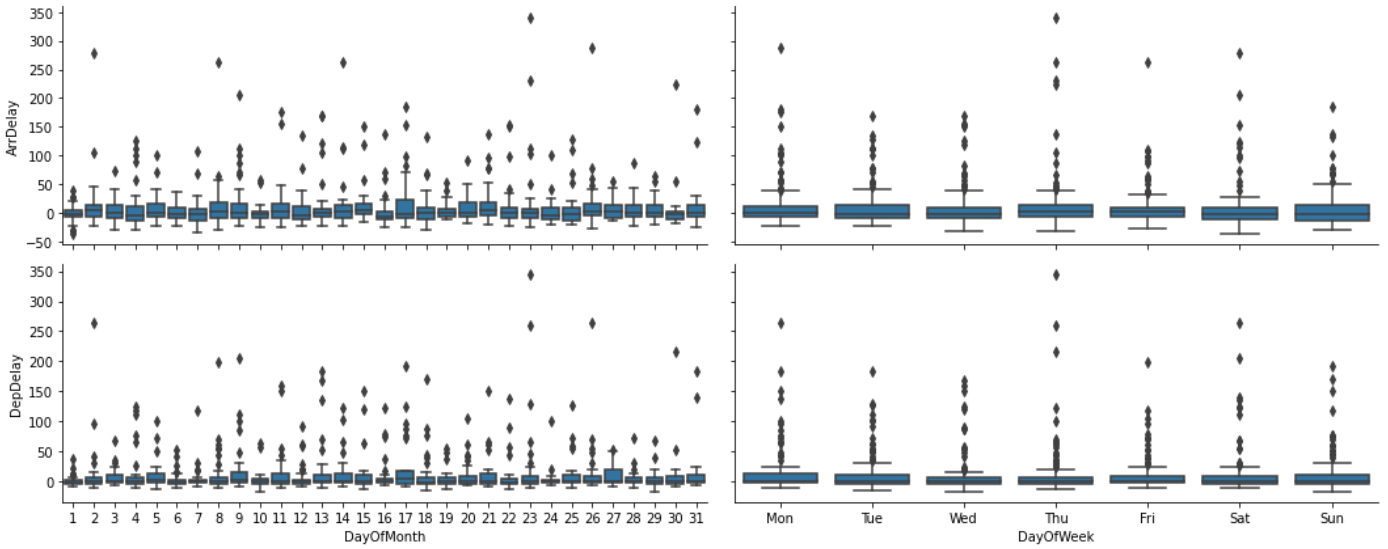
```
<Figure size 1440x936 with 0 Axes>
```



In [51]:
```
# plot matrix of numeric features against categorical features.
# can use a larger sample since there are fewer plots and they're simpler in nature.

categoric_vars = ['DayOfMonth', 'DayOfWeek']

samples = np.random.choice(df_flight.shape[0], 10000, replace = False)
fligtht_samp = df_flight.loc[samples,:]

def boxgrid(x, y, **kwargs):
```

```
    """ Quick hack for creating box plots with seaborn's PairGrid. """
    default_color = sb.color_palette()[0]
    sb.boxplot(data=flight_samp, x=x, y=y, color = default_color)

plt.figure(figsize = [20, 13]);
g = sb.PairGrid(data = flight_samp, y_vars = ['ArrDelay', 'DepDelay'], x_vars = categori
                height = 3, aspect = 2.5)
g.map(boxgrid)
plt.show()
```

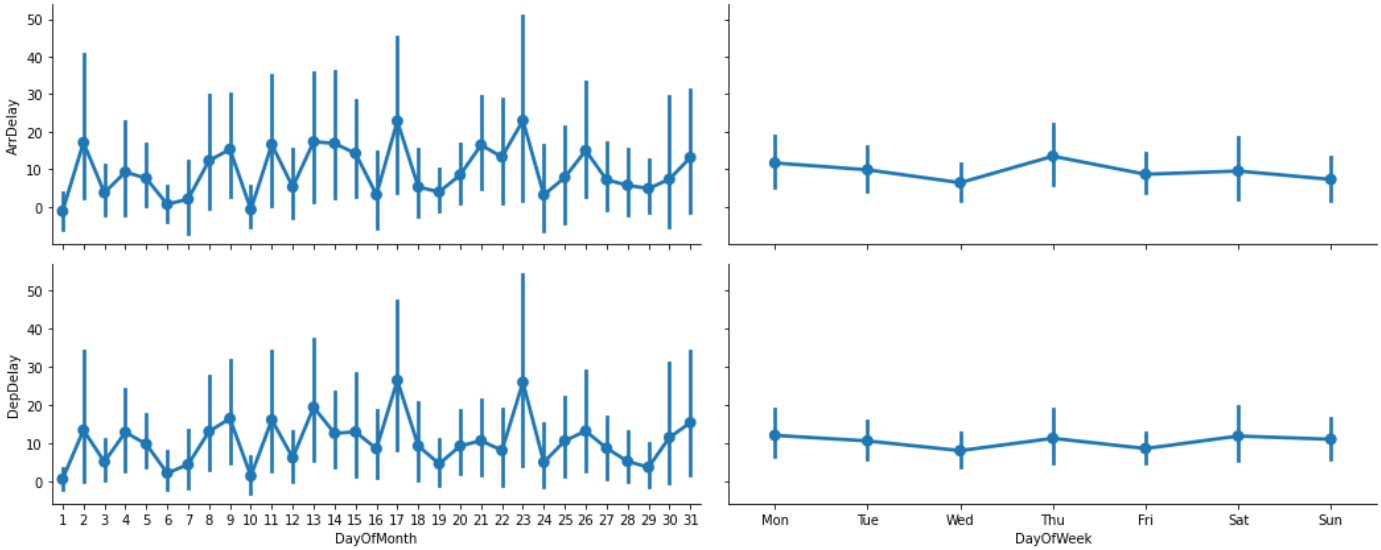
```
<Figure size 1440x936 with 0 Axes>
```



In [52]:

```
# plot matrix of numeric features against categorical features.
# can use a larger sample since there are fewer plots and they're simpler in nature.

categoric_vars = ['DayOfMonth', 'DayOfWeek']

samples = np.random.choice(df_flight.shape[0], 10000, replace = False)
fligtht_samp = df_flight.loc[samples,:]

def pointgrid(x, y, **kwargs):
    """ Quick hack for creating box plots with seaborn's PairGrid. """
    default_color = sb.color_palette()[0]
    sb.pointplot(data=flight_samp, x=x, y=y, color = default_color)

plt.figure(figsize = [20, 13]);
g = sb.PairGrid(data = flight_samp, y_vars = ['ArrDelay', 'DepDelay'], x_vars = categori
                height = 3, aspect = 2.5)
g.map(pointgrid)
plt.show()
```

```
<Figure size 1440x936 with 0 Axes>
```

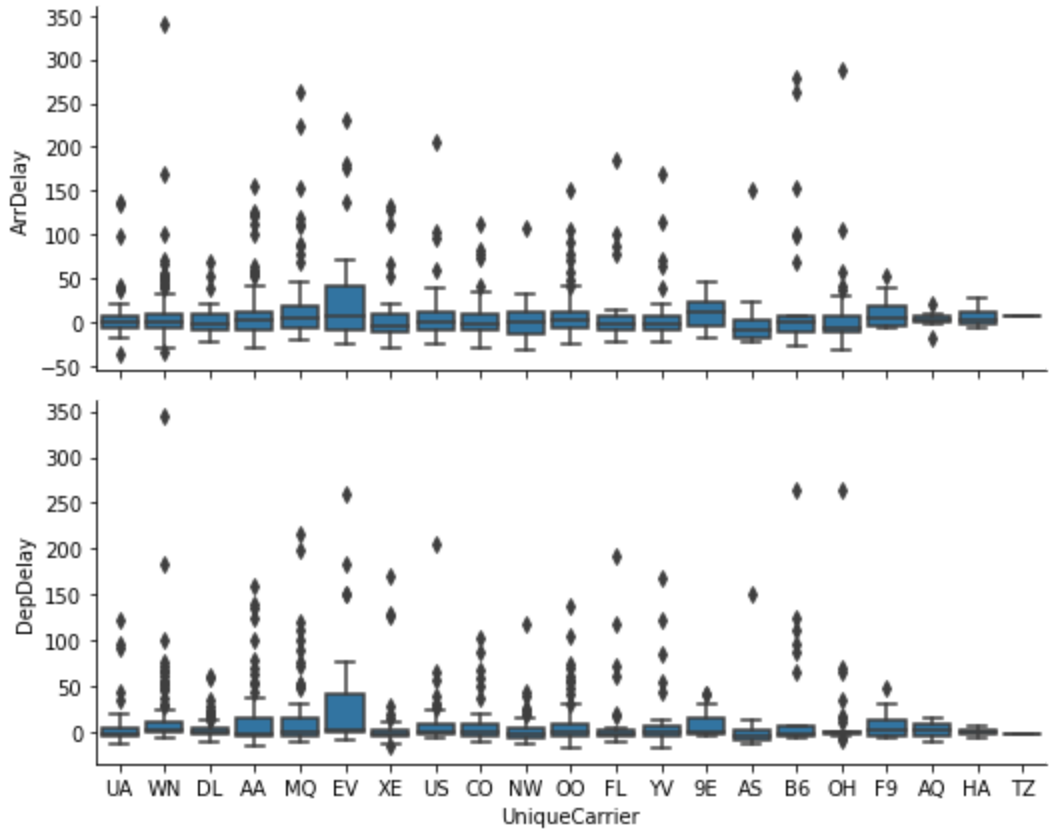


Observations: From these plots, we can see that `ArrDelay` and `DepDelay` are pretty the same accross `Year`, `Month`, `DayOfMonth` and `DayOfWeek`. The medians of Delays are almost around 0 minute in 2006 and 2007, same for months, days of month and days of week. But there are months like January, Jully, August and December with larger delay, may be due to the fact that they are busiest flights months. Same for 2nd, 12th, 16th, 19th, 21th and 28th day of the month with higher average of delay. From days of the week, we have days like Monday, and Friday, with larger delay. Saturday which was the less busiest flight day, is still the day with less average of delay.

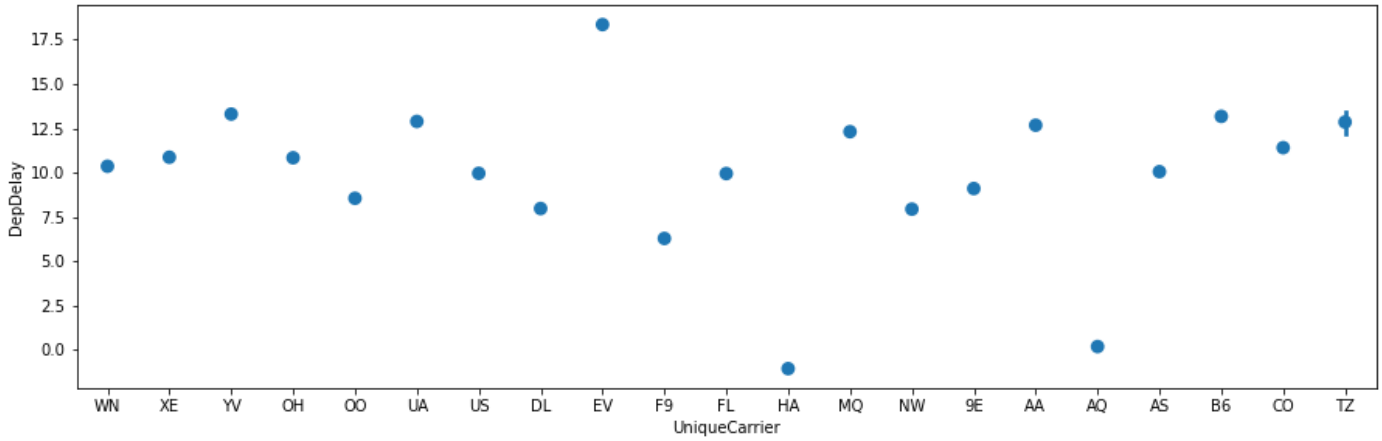
> Relationship between Delay and UniqueCarrier

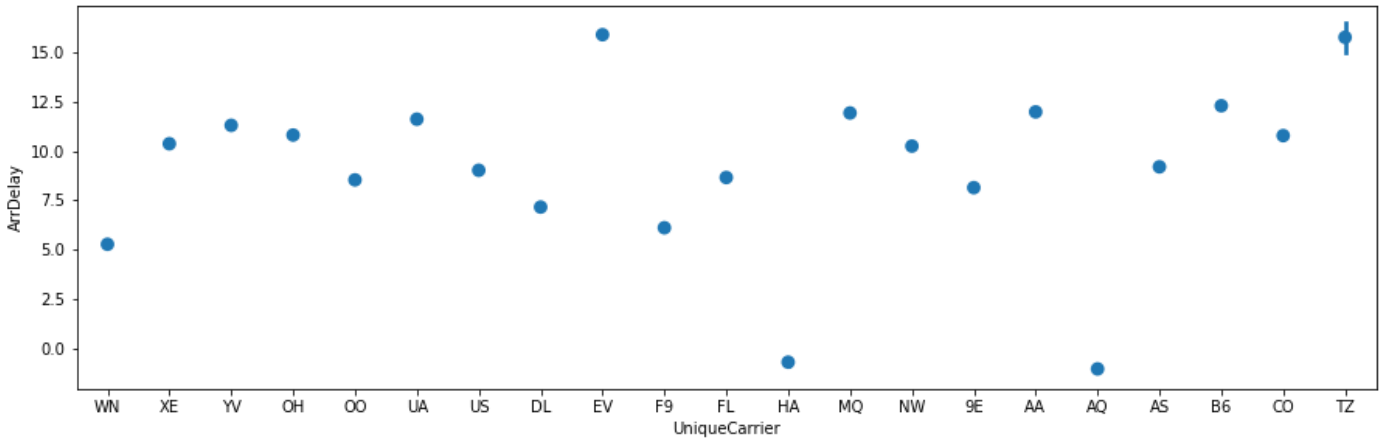
In [53]:
```
# plot matrix of numeric features against categorical features.
# can use a larger sample since there are fewer plots and they're simpler in nature.

categoric_vars = ['UniqueCarrier']

samples = np.random.choice(df_flight.shape[0], 50000, replace = False)
fligtht_samp = df_flight.loc[samples,:]

def boxgrid(x, y, **kwargs):
    """ Quick hack for creating box plots with seaborn's PairGrid. """
    default_color = sb.color_palette()[0]
    sb.boxplot(data=flight_samp, x=x, y=y, color = default_color)

plt.figure(figsize = [20, 13]);
g = sb.PairGrid(data = flight_samp, y_vars = ['ArrDelay', 'DepDelay'], x_vars = categori
                height = 3, aspect = 2.5)
g.map(boxgrid)
plt.show()
```

```
<Figure size 1440x936 with 0 Axes>
```



In [54]:
```
plt.figure(figsize=[15, 10])

base_color = sb.color_palette()[0]

plt.subplot(2,1,1)
sb.pointplot(data = df_flight, x = 'UniqueCarrier', y = 'DepDelay',
             linestyles = '', color = base_color)

plt.subplot(2,1,2)
sb.pointplot(data = df_flight, x = 'UniqueCarrier', y = 'ArrDelay',
             linestyles = '', color = base_color)

plt.show()
```

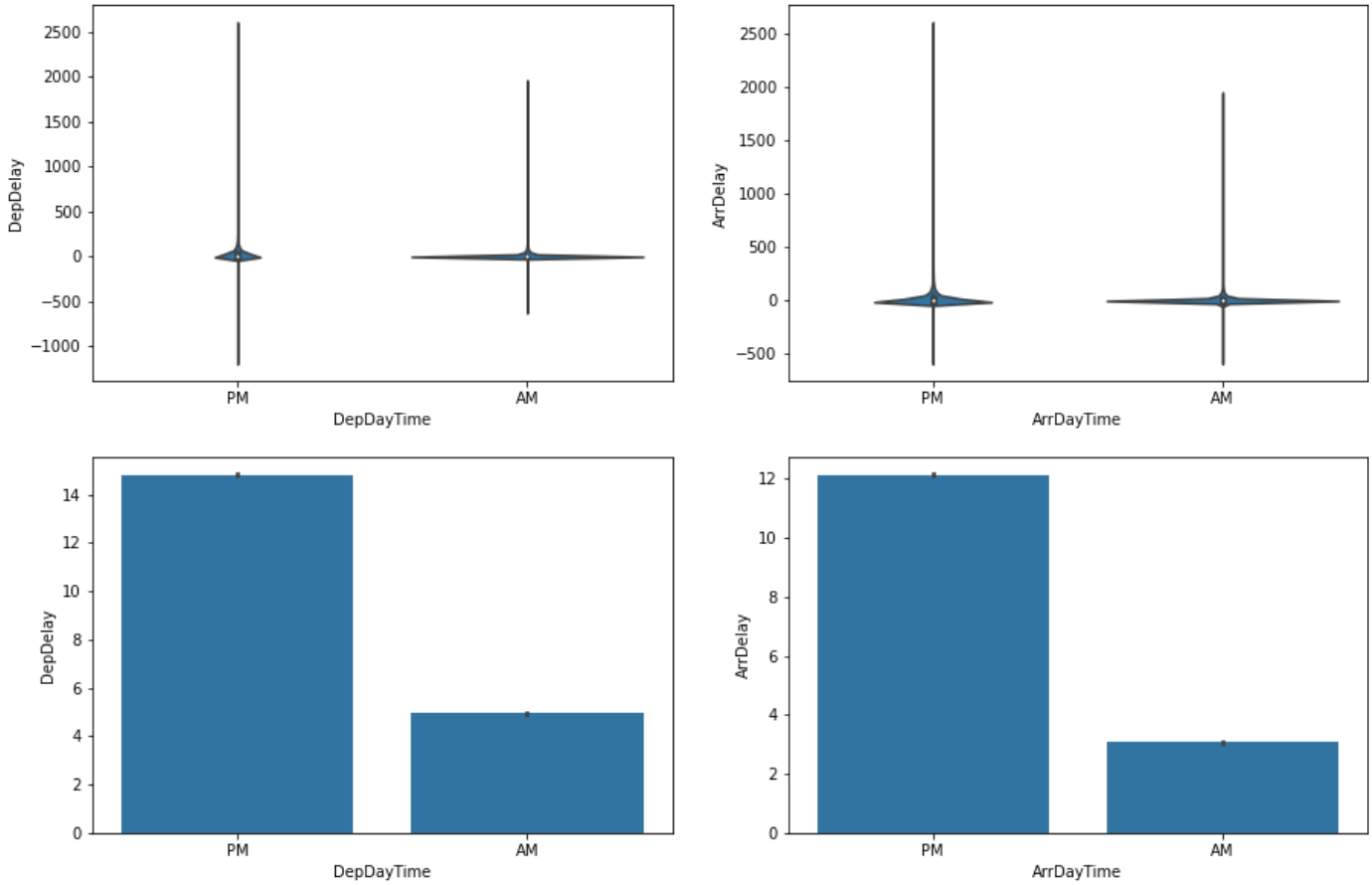




Observations: Delays are almost the same accross all unique carrier, with a median around 0 minute. We can see some carriers like MQ, CO, AQ, OO, EV, YV and 9E, with larger delays. WN which was the carrier with larger number of flights, does not have larger delay, but seems to have the highest median. But means vary a lot EV, and TZ have the highest mean, where AH and AQ have the less mean both for ArrDelay and DepDelay.

Relationship between delay and day time

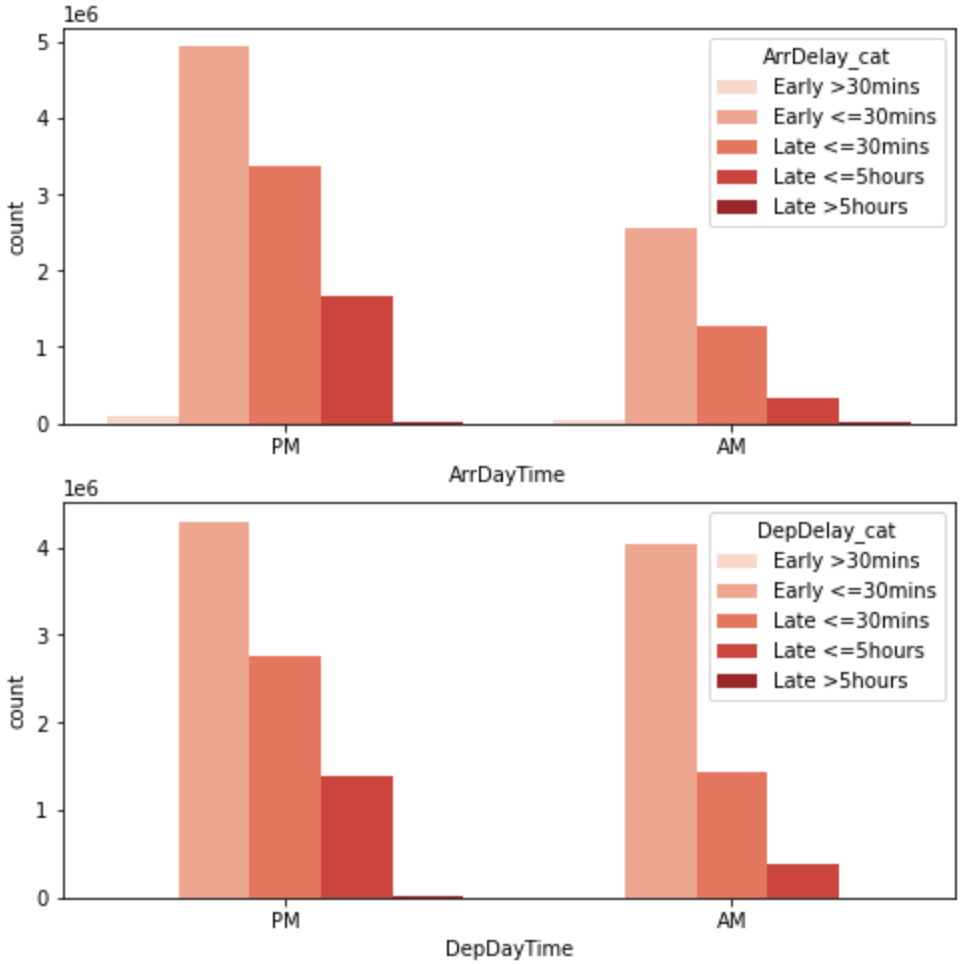
In [55]:
```
plt.figure(figsize=[15, 10])

base_color = sb.color_palette()[0]

plt.subplot(2, 2, 1)
sb.violinplot(data = df_flight, x = 'DepDayTime', y = 'DepDelay',
              color = base_color)

plt.subplot(2,2,2)
sb.violinplot(data = df_flight, x = 'ArrDayTime', y = 'ArrDelay',
              color = base_color)

plt.subplot(2,2,3)
sb.barplot(data = df_flight, x = 'DepDayTime', y = 'DepDelay',
              color = base_color)

plt.subplot(2,2,4)
sb.barplot(data = df_flight, x = 'ArrDayTime', y = 'ArrDelay',
              color = base_color)

plt.show()
```



Observations: ArrDelay and DepDelay seem to be much more larger in the evening with a mean of 12 minutes for ArrDelay and 14 minutes for DepDelay. We can also see that Delay in PM time are very large. There are late flights departure (late arrival too) of 2500 minutes as well as earlier flight departure of 1000 minutes for evening scheduled flights. We will understand better the relation with delays categories.

Relatoin between Day Time and delays categories

In [56]:
```
# since there's only two subplots to create, using the full data should be fine.
plt.figure(figsize = [8, 8])

# subplot 1: ArrDayTime vs ArrDelay_cat
plt.subplot(2, 1, 1)
sb.countplot(data = df_flight, x = 'ArrDayTime', hue = 'ArrDelay_cat', palette = 'Reds')

# subplot 2: DepDayTime vs. DepDelay_cat
ax = plt.subplot(2, 1, 2)
sb.countplot(data = df_flight, x = 'DepDayTime', hue = 'DepDelay_cat', palette = 'Reds')
#ax.legend(ncol = 2) # re-arrange legend to reduce overlapping

plt.show()
```

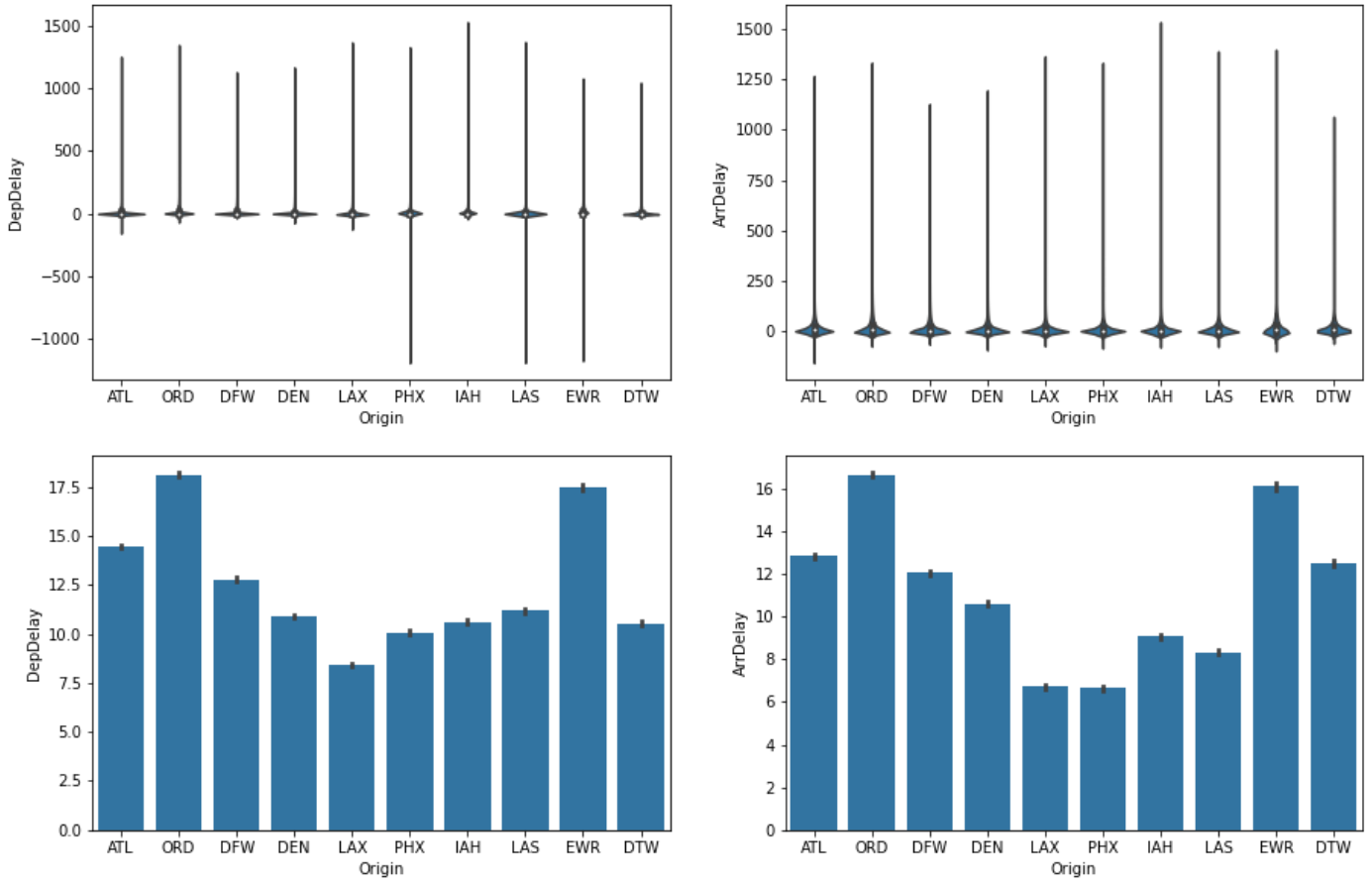


Observations: As seen before, there are more flights in the evening(PM) than morning(AM). It also appears that, there are more earlier flight departure and arrival (less than 30 minutes earlier) than any other flight delay category both in the morning and in the evening. We can also see that there are more earlier flight departure (less than 30 minutes earlier) than earlier flight arrival. Late flight departure and arrival (less than 30 minutes delay and less than 5hours delays) are prety same both in the morning and in the evening.

Relation between delays and top 10 busiest airports (origin and destination)

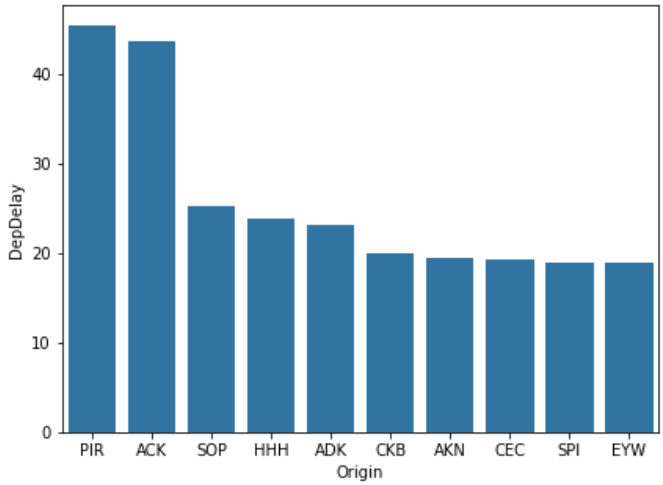
In [57]:
```
#let's put busiest airports in a list
airports = df_flight.Origin.value_counts().head(10).index

#filter data on busiest airports
df_airport = df_flight.loc[df_flight.Origin.isin(airports)]


ordered_airport = pd.api.types.CategoricalDtype(ordered = True,
                                                categories = airports)

df_airport['Origin'] = df_airport['Origin'].astype(ordered_airport)

plt.figure(figsize=[15, 10])

base_color = sb.color_palette()[0]

plt.subplot(2, 2, 1)
sb.violinplot(data = df_airport, x = 'Origin', y = 'DepDelay',
              color = base_color)

plt.subplot(2,2,2)
sb.violinplot(data = df_airport, x = 'Origin', y = 'ArrDelay',
```

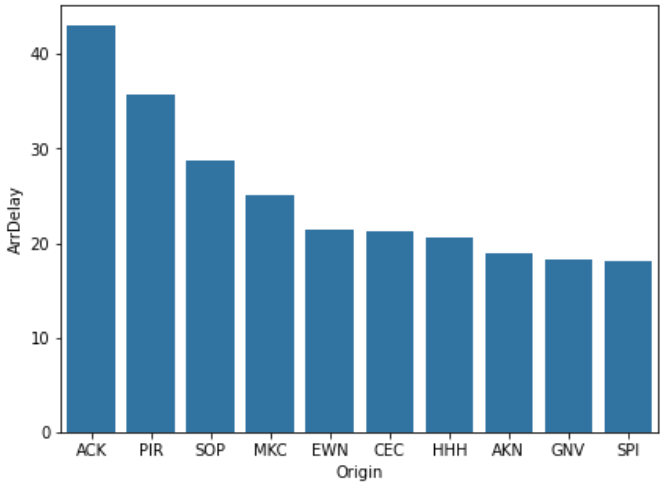
```
                color = base_color)

plt.subplot(2,2,3)
sb.barplot(data = df_airport, x = 'Origin', y = 'DepDelay',
                color = base_color)

plt.subplot(2,2,4)
sb.barplot(data = df_airport, x = 'Origin', y = 'ArrDelay',
                color = base_color)

plt.show()
```



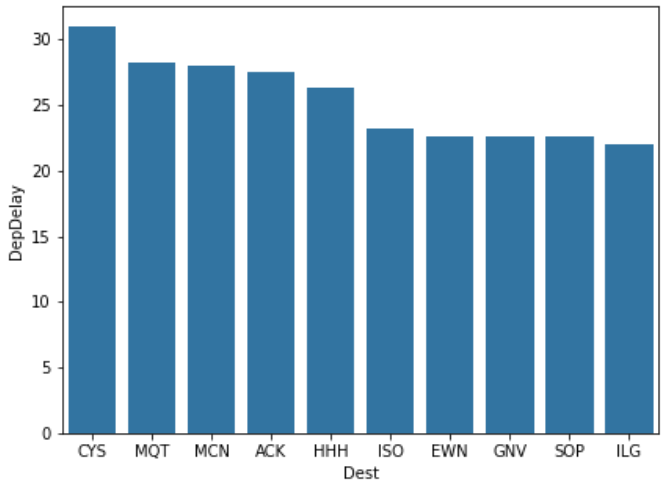
In [58]:

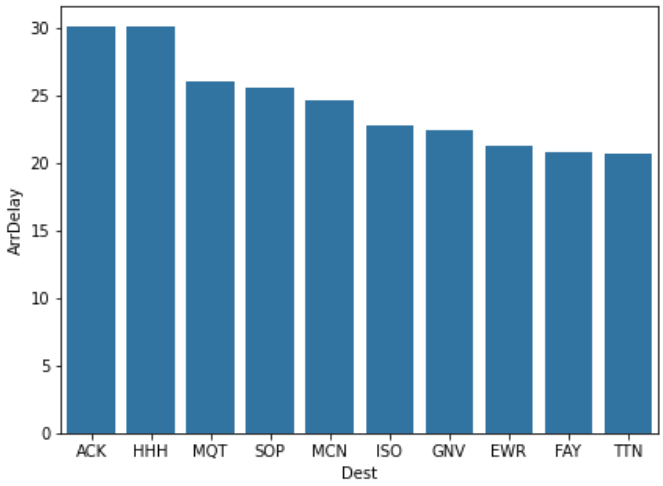
```
df_airport_deplay = df_flight.groupby('Origin')['DepDelay', 'ArrDelay'].mean().sort_valu

df_airport_arrdelay = df_flight.groupby('Origin')['DepDelay', 'ArrDelay'].mean().sort_va

plt.figure(figsize=[15, 5])

plt.subplot(1,2,1)
sb.barplot(data = df_airport_deplay, x = df_airport_deplay.index, y = 'DepDelay',
                color = base_color)

plt.subplot(1,2,2)
sb.barplot(data = df_airport_arrdelay, x = df_airport_arrdelay.index, y = 'ArrDelay',
                color = base_color)

plt.show()
```





In [59]:
```
df_airport_deplay = df_flight.groupby('Dest')['DepDelay', 'ArrDelay'].mean().sort_values

df_airport_arrdelay = df_flight.groupby('Dest')['DepDelay', 'ArrDelay'].mean().sort_valu

plt.figure(figsize=[15, 5])

plt.subplot(1,2,1)
sb.barplot(data = df_airport_deplay, x = df_airport_deplay.index, y = 'DepDelay',
                color = base_color)

plt.subplot(1,2,2)
sb.barplot(data = df_airport_arrdelay, x = df_airport_arrdelay.index, y = 'ArrDelay',
                color = base_color)

plt.show()
```





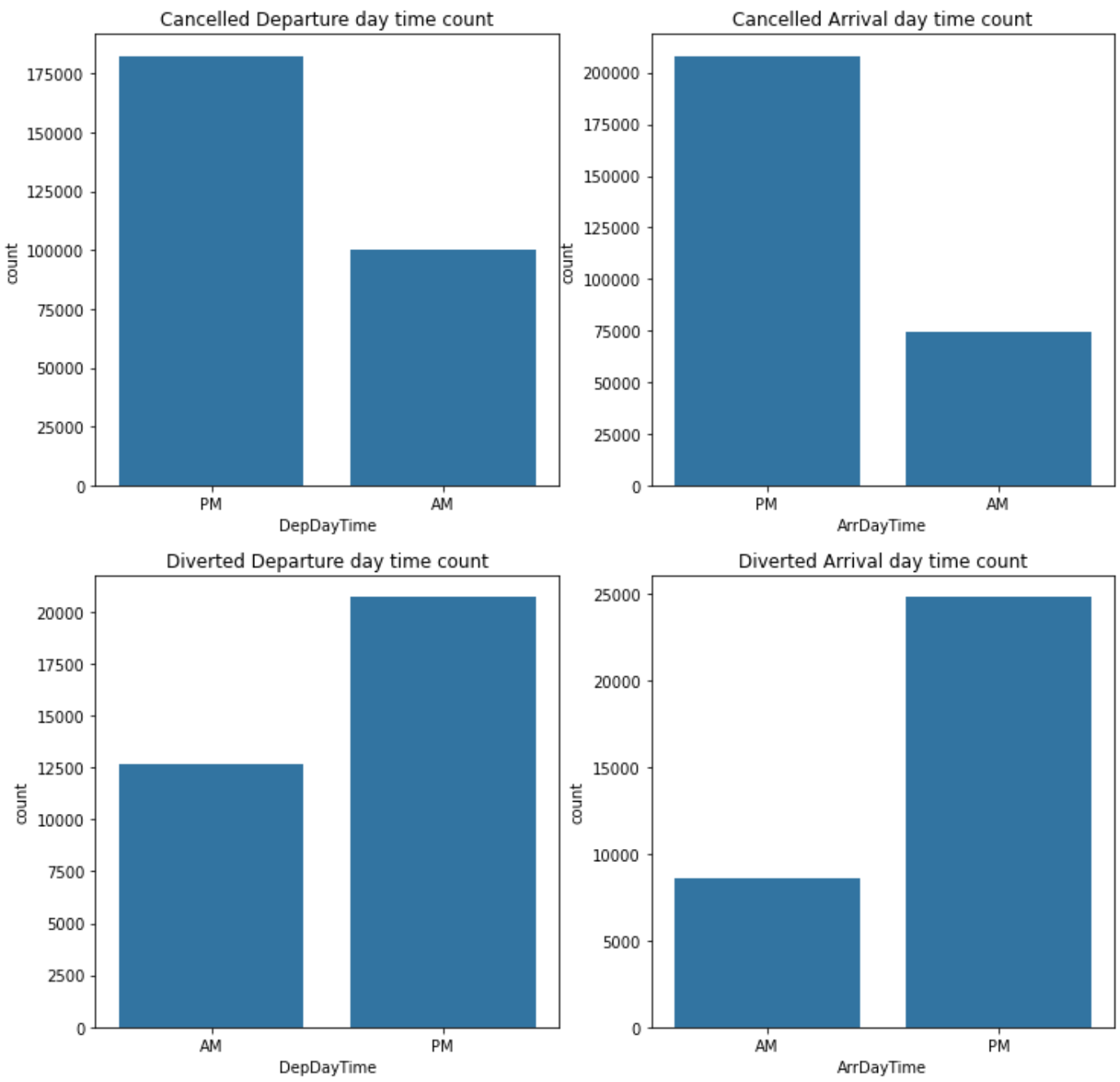
Observations:

- As origin, none of the top 10 busiest airport is amongst airports with maximum average delays. PIP is the airport with maximum average delay in departure flight, while ACK is the one in arrival flight with an average delay of more than 40 minutes each.
- As destination, none of the top 10 busiest airport is amongst airports with maximum average delays. CYS is the airport with maximum average delay in departure, while ACK is the one in arrival with an average delay of around 30 minutes each.
- Amongst the top 10 busiest airports, ORD, and EWR seem to have more delay with a mean of around 17.5 minutes delay in departure and 16 minutes in arrival. ATL which is the most busiest airport only has a mean delay of around 14 minutes in departure and 13 minutes in arrival. LAX

airport is the airport with less mean delay (around 7.5 minutes in departure and 6.5 minutes in arrival). All these ariport have a median delay (both in departure and arrival) of around 0 minute.

Relation between cancelled and Day time, then diverted and day time

In [60]:
```
# sWe will filter data to consider only flight that where canceled and only flight that
plt.figure(figsize = [12, 12])

# subplot 1: DayTime vs Cancelled
plt.subplot(2, 2, 1)
sb.countplot(data = df_flight[df_flight.Cancelled == 1], x = 'DepDayTime', color = base_
plt.title('Cancelled Departure day time count')

plt.subplot(2, 2, 2)
sb.countplot(data = df_flight[df_flight.Cancelled == 1], x = 'ArrDayTime', color = base_
plt.title('Cancelled Arrival day time count')

# subplot 2: DayTime vs. Diverted
plt.subplot(2, 2, 3)
sb.countplot(data = df_flight[df_flight.Diverted == 1], x = 'DepDayTime', color = base_c
plt.title('Diverted Departure day time count')

plt.subplot(2, 2, 4)
sb.countplot(data = df_flight[df_flight.Diverted == 1], x = 'ArrDayTime', color = base_c
plt.title('Diverted Arrival day time count')


plt.show()
```

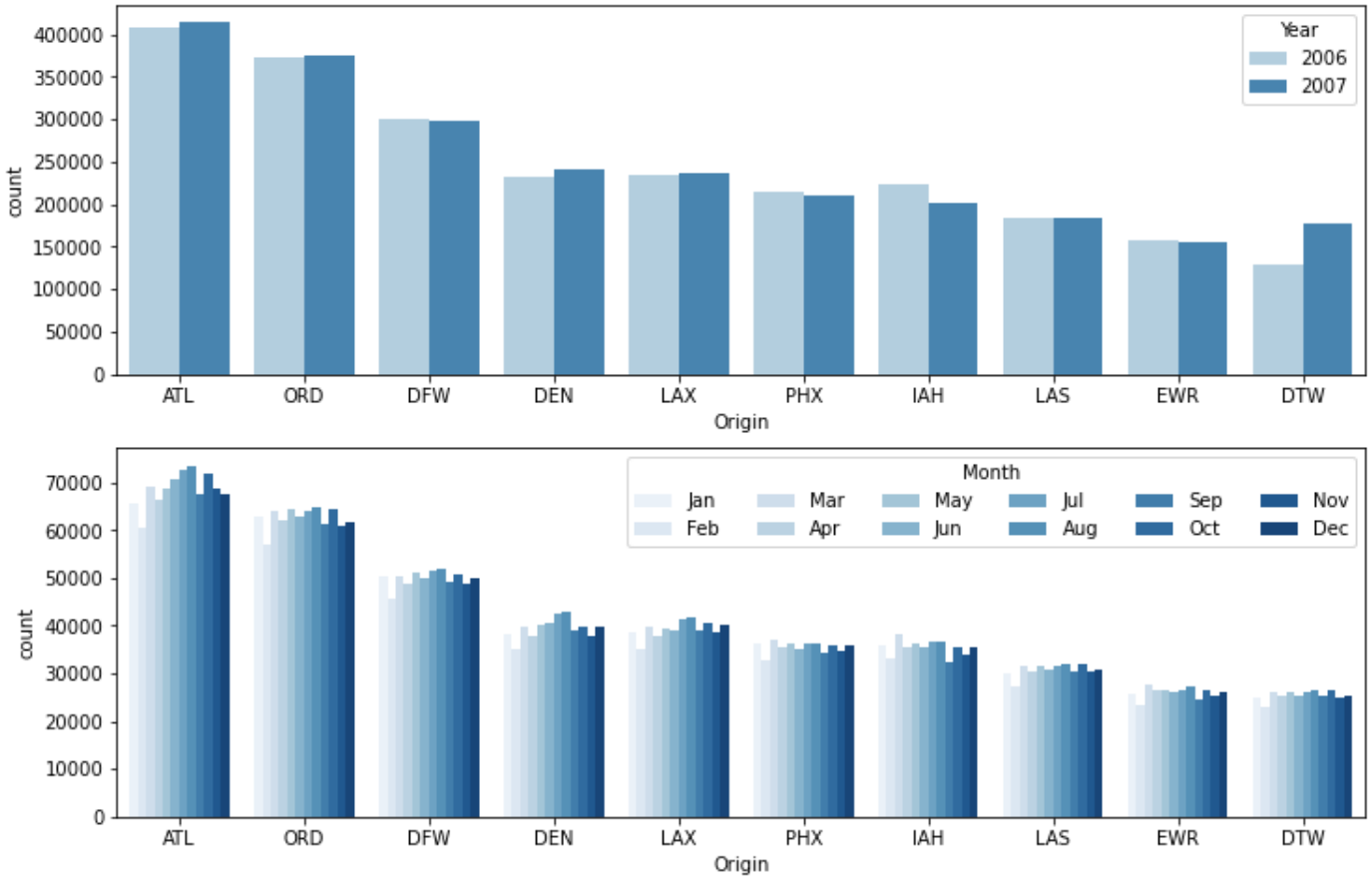


Observations: From these plots, we see that flights scheduled in the evening are mostly subject to cancellation or divertion.

Relation between top 10 busiest Airport and Year, Month

In [61]:
```
#let's put busiest airports in a list
airports = df_flight.Origin.value_counts().head(10).index

#filter data on busiest airports
df_airport = df_flight.loc[df_flight.Origin.isin(airports)]


ordered_airport = pd.api.types.CategoricalDtype(ordered = True,
                                                categories = airports)

df_airport['Origin'] = df_airport['Origin'].astype(ordered_airport)

# since there's only two subplots to create, using the full data should be fine.
plt.figure(figsize = [12, 8])

# subplot 1: ArrDayTime vs ArrDelay_cat
```

```
plt.subplot(2, 1, 1)
sb.countplot(data = df_airport, x = 'Origin', hue = 'Year', palette = 'Blues')

# subplot 2: DepDayTime vs. DepDelay_cat
ax = plt.subplot(2, 1, 2)
sb.countplot(data = df_airport, x = 'Origin', hue = 'Month', palette = 'Blues')
ax.legend(ncol = 6, title='Month') # re-arrange legend to reduce overlapping

plt.show()
```



Observations: We can see that the number of flight in 2006 and 2007, does not vary a lot for each top 10 busiest airport except that, there was much more flights in 2006 and 2007 at IAH, and DTW respectively. Variation of number of flights per month is prety same for each top 10 busiest airport, where February is the less busiest month and August the most busiest month.

Talk about some of the relationships you observed in this part of the investigation. How did the feature(s) of interest vary with other features in the dataset?

> We dsicover that ArrDelay is highly correlated with DepDelay which is normal, if a flight is delayed at departure, we should expect a delay at arrival, the higher the delay at departure should conduct to a higher delay at arrival. Delay does not vary with year. There is a litle variation with month where busiest flight months like August, July and December seem to have more average flight delay. Is it also true for days of the week, where Saturday which was the less busiest flight day also has less average delay. We also discover that flights with higher average delay are flights scheduled in the evening(PM). There is no trend between delay and busiest airport. Amongst the busiest airports, ORD and EWR have more average flight delay than the busiest aiport (ATL). Is it also true with the unique carrier, EV and TZ which are not the most busiest carriers has the highest

> average delay, while WN which is the most busiest carrier is in the middle with an average of 5 minutes delay in arrival and 10 minutes in departure.

Did you observe any interesting relationships between the other features (not the main feature(s) of interest)?

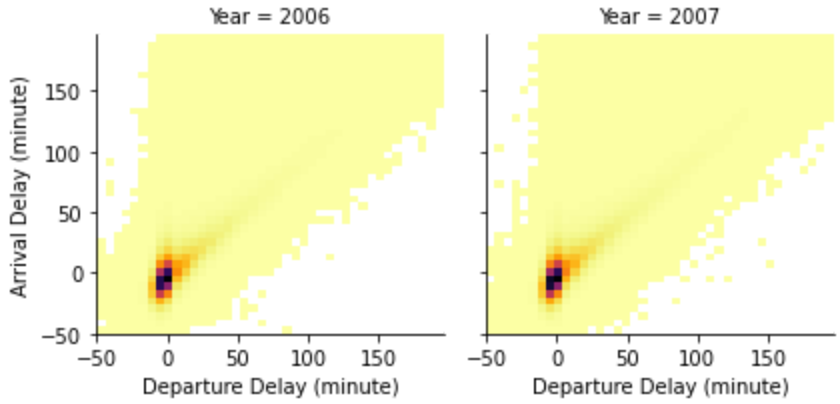
> Cancelled and Diverted are not really correlated with any other variable, but cancelled and diverted flights are mostly flights scheduled in the evening (PM). The Variation of number of flight per month and year is prety the same for each top 10 busiest airport. ORD is the airport with more cancelled flights.

Multivariate Exploration

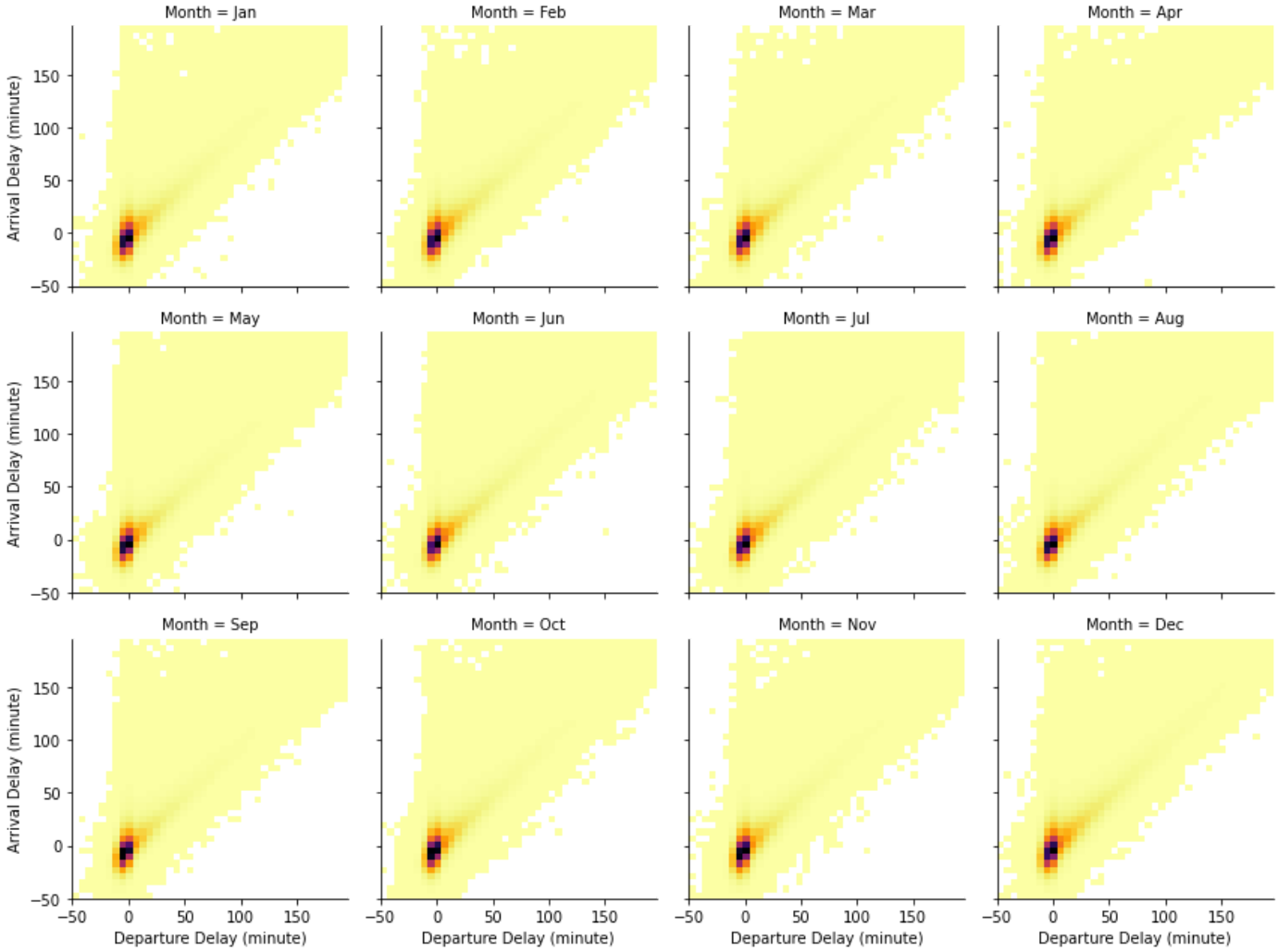
> The main purpose here is to check the relationship between categorical variables (Day time, month, year, Day of week, day of month, origin and dest, unique carrrier) and the delay, cancelled and diverted variables.

In [62]:
```
def hist2dgrid(x, y, **kwargs):
    """ Quick hack for creating heat maps with seaborn's PairGrid. """
    palette = kwargs.pop('color')
    x_min, x_max = -50, 200
    bins_x = np.arange(x_min, x_max, 6)
    bins_y = np.arange(x_min, x_max, 6)
    plt.hist2d(x, y, bins = [bins_x, bins_y], cmap = palette, cmin = 0.3)
    plt.xticks(np.arange(x_min, x_max, 50))
    plt.yticks(np.arange(x_min, x_max, 50))
```

In [63]:
```
# create faceted heat maps on Year variable
g = sb.FacetGrid(data = df_flight, col = 'Year', col_wrap = 2, height = 3,
                 xlim = [-50, 200])
g.map(hist2dgrid, 'DepDelay', 'ArrDelay', color = 'inferno_r')
g.set_xlabels('Departure Delay (minute)')
g.set_ylabels('Arrival Delay (minute)')

plt.show()
```

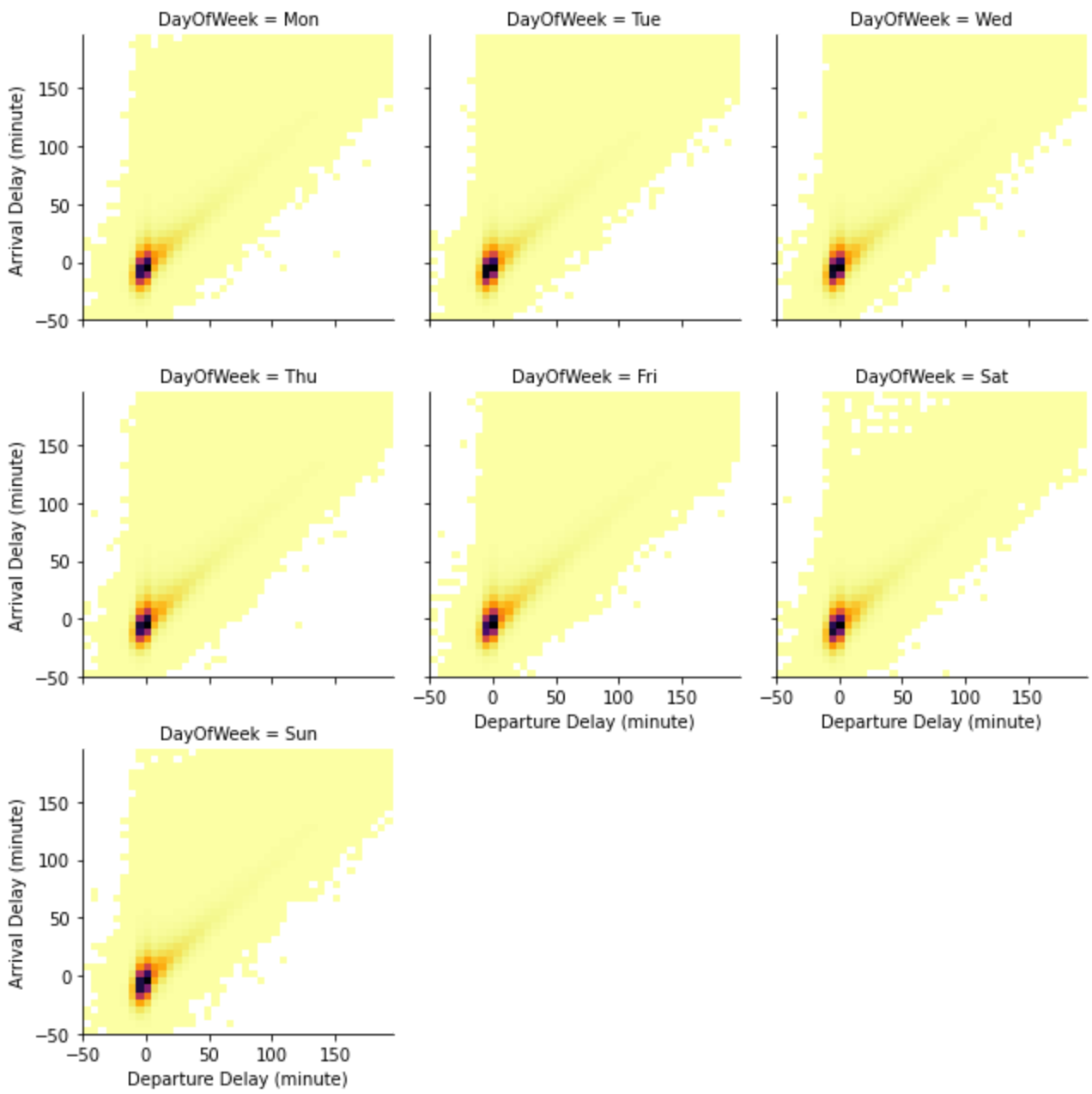


In [64]:
```
# create faceted heat maps on Month variable
g = sb.FacetGrid(data = df_flight, col = 'Month', col_wrap = 4, height = 3,
                 xlim = [-50, 200])
g.map(hist2dgrid, 'DepDelay', 'ArrDelay', color = 'inferno_r')
g.set_xlabels('Departure Delay (minute)')
g.set_ylabels('Arrival Delay (minute)')

plt.show()
```



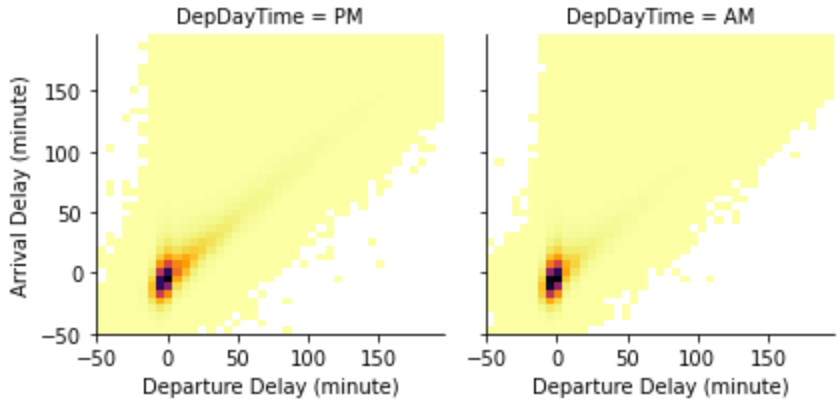
In [65]:

```
# create faceted heat maps on DayOfWeek variable
g = sb.FacetGrid(data = df_flight, col = 'DayOfWeek', col_wrap = 3, height = 3,
                 xlim = [-50, 200])
g.map(hist2dgrid, 'DepDelay', 'ArrDelay', color = 'inferno_r')
g.set_xlabels('Departure Delay (minute)')
g.set_ylabels('Arrival Delay (minute)')

plt.show()
```



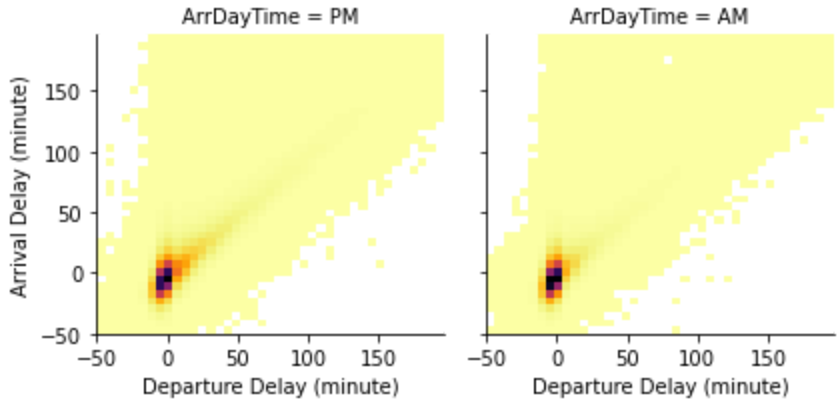
Observations:In each of the faceted heat maps, we can see the relationship of date against Departure delay and Arrivl delay, no variation. As the year, month, day of week changes, the 'cloud' of points is spreader around 0 minute (between -50 and 50 minutes).

In [66]:
```
# create faceted heat maps on DepDayTime variable
g = sb.FacetGrid(data = df_flight, col = 'DepDayTime', col_wrap = 2, height = 3,
                 xlim = [-50, 200])
g.map(hist2dgrid, 'DepDelay', 'ArrDelay', color = 'inferno_r')
g.set_xlabels('Departure Delay (minute)')
g.set_ylabels('Arrival Delay (minute)')

plt.show()
```



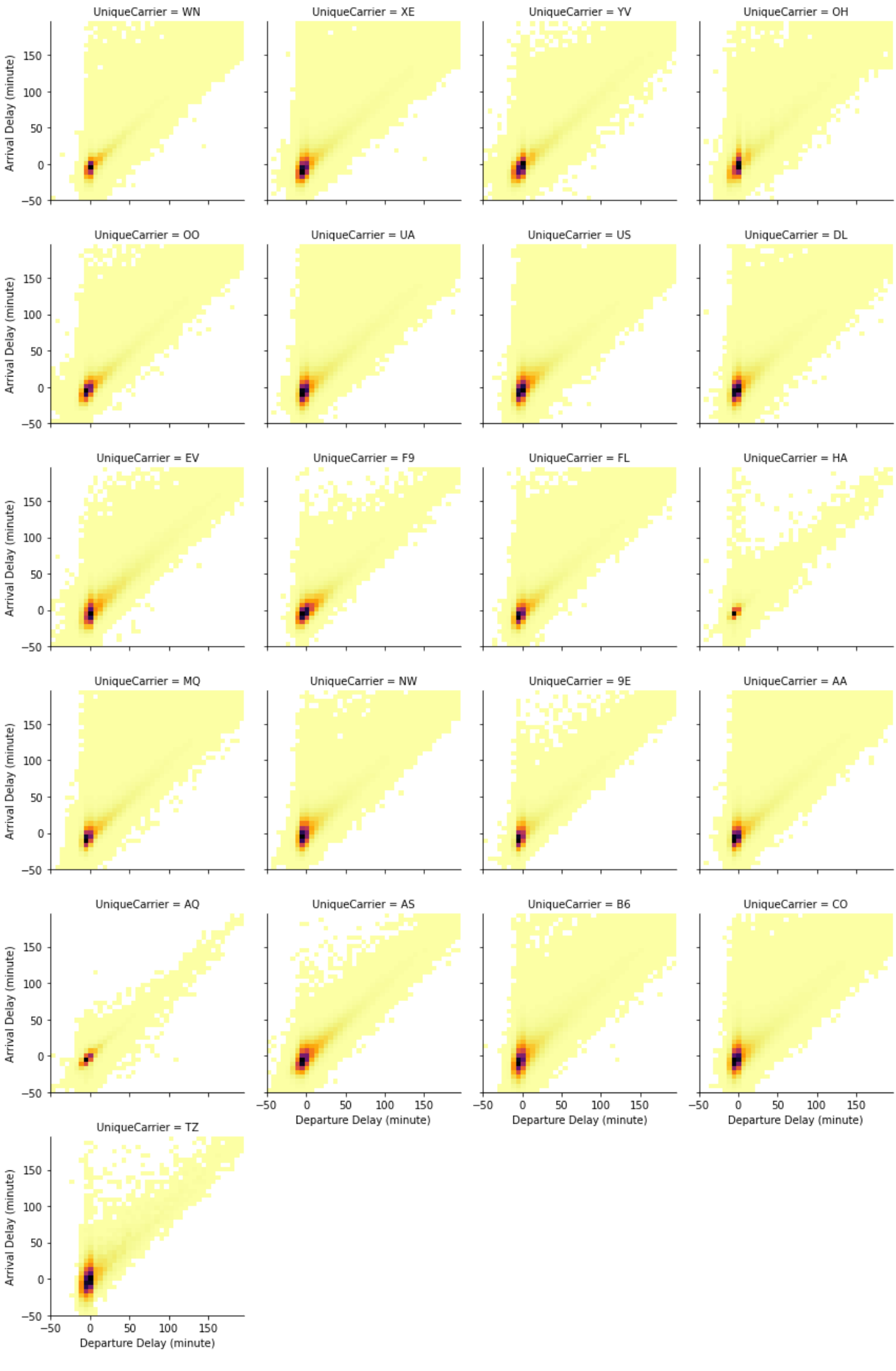
In [67]:
```
# create faceted heat maps on ArrDayTime variable
g = sb.FacetGrid(data = df_flight, col = 'ArrDayTime', col_wrap = 2, height = 3,
                 xlim = [-50, 200])
g.map(hist2dgrid, 'DepDelay', 'ArrDelay', color = 'inferno_r')
g.set_xlabels('Departure Delay (minute)')
g.set_ylabels('Arrival Delay (minute)')

plt.show()
```

```
The history saving thread hit an unexpected error (OperationalError('database is locke
d')).History will not be written to the database.
```



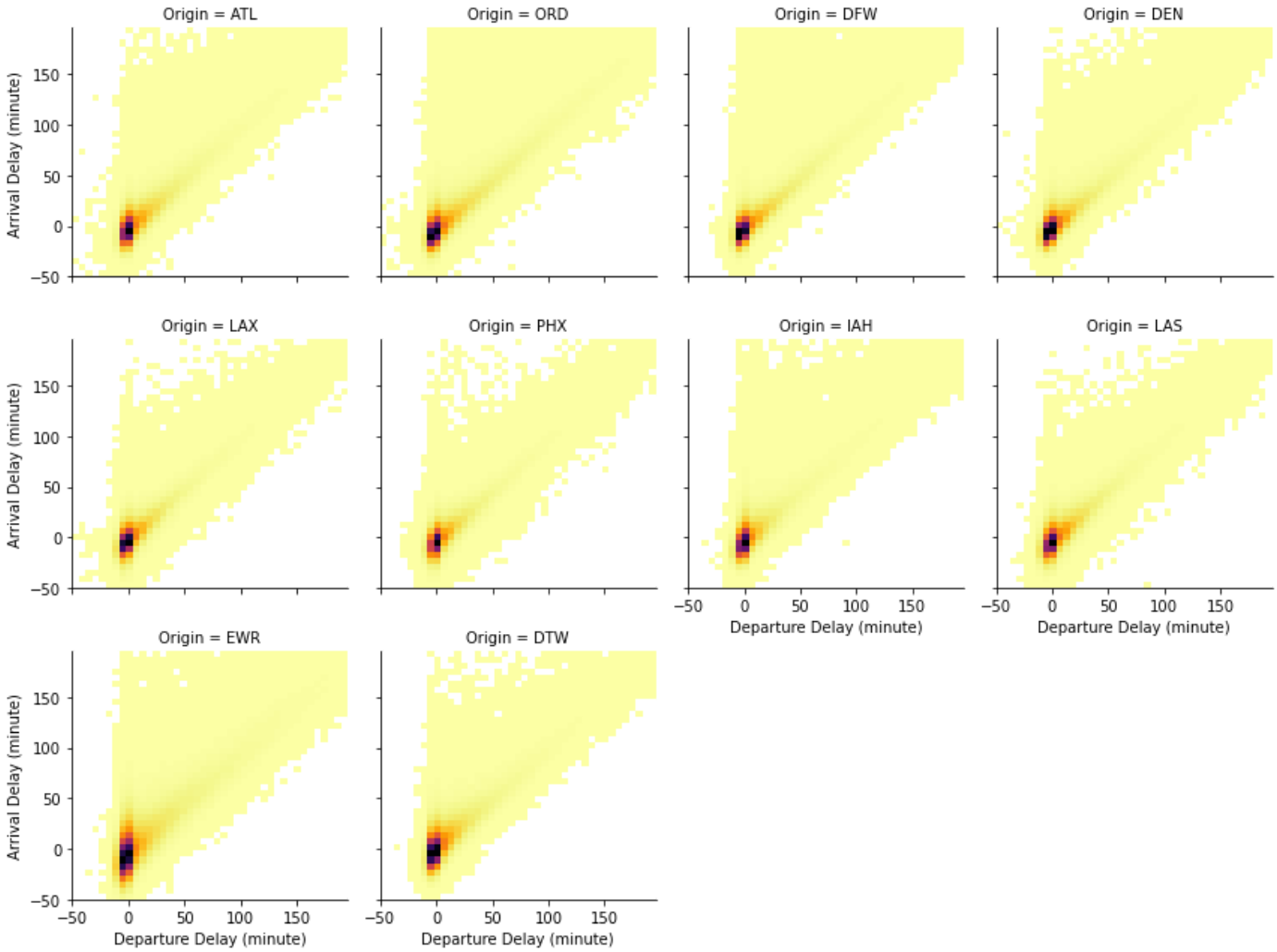
Observations: In the PM day time, the cloud of points are spreader around 0 minute than in the AM day time.

In [68]:
```
# create faceted heat maps on Unique Carrier variable
g = sb.FacetGrid(data = df_flight, col = 'UniqueCarrier', col_wrap = 4, height = 3,
                 xlim = [-50, 200])
g.map(hist2dgrid, 'DepDelay', 'ArrDelay', color = 'inferno_r')
g.set_xlabels('Departure Delay (minute)')
g.set_ylabels('Arrival Delay (minute)')

plt.show()
```


UniqueCarrier = WN
UniqueCarrier = XE
UniqueCarrier = YV
UniqueCarrier = OH
UniqueCarrier = OO
UniqueCarrier = UA
UniqueCarrier = US
UniqueCarrier = DL
UniqueCarrier = EV
UniqueCarrier = F9
UniqueCarrier = FL
UniqueCarrier = HA
UniqueCarrier = MQ
UniqueCarrier = NW
UniqueCarrier = 9E
UniqueCarrier = AA
UniqueCarrier = AQ
UniqueCarrier = AS
UniqueCarrier = B6
UniqueCarrier = CO
UniqueCarrier = TZ
Arrival Delay (minute)
Departure Delay (minute)
−50
0
50
100
150

Observations: The busier a Carrier is, the spreader is the cloud of points around 0 minute.

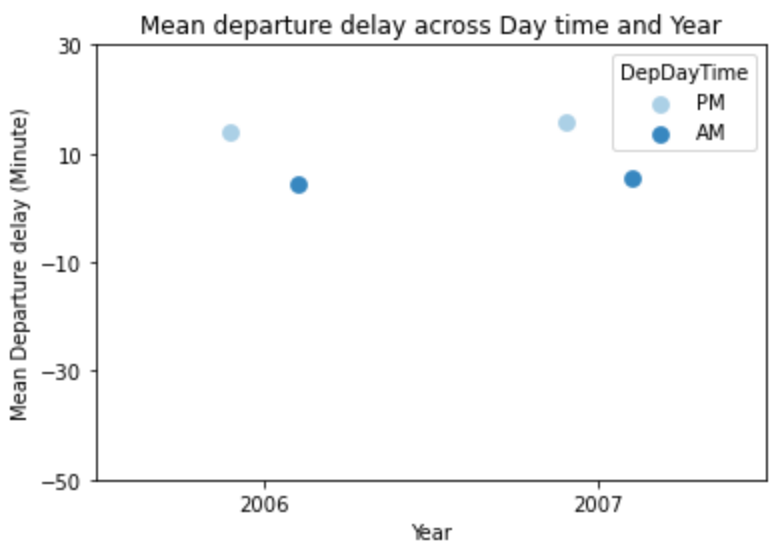
In [69]:
```
# create faceted heat maps on top 10 busiest airport variable
g = sb.FacetGrid(data = df_airport, col = 'Origin', col_wrap = 4, height = 3,
                 xlim = [-50, 200])
g.map(hist2dgrid, 'DepDelay', 'ArrDelay', color = 'inferno_r')
g.set_xlabels('Departure Delay (minute)')
g.set_ylabels('Arrival Delay (minute)')

plt.show()
```



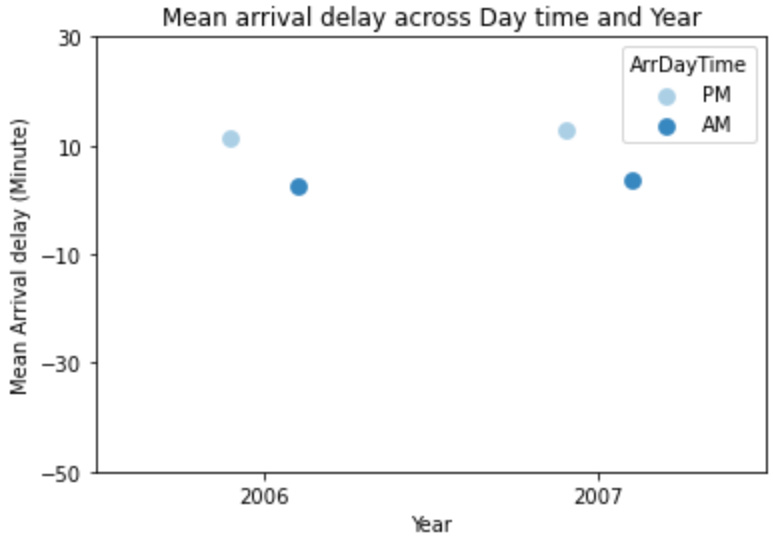
Observations: The busier an airport is, the spreader is the cloud of points around 0 minute.

Relation between delay, day time and year

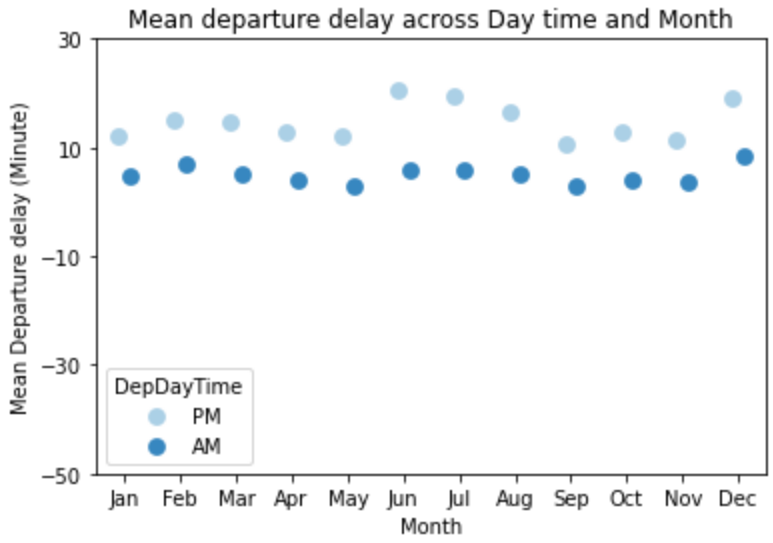
In [70]:
```
ax = sb.pointplot(data = df_flight, x = 'Year', y = 'DepDelay', hue = 'DepDayTime',
           palette = 'Blues', linestyles = '', dodge = 0.2)
plt.title('Mean departure delay across Day time and Year')
plt.ylabel('Mean Departure delay (Minute)')

x_min, x_max = -50, 50

plt.yticks(np.arange(x_min, x_max, 20));
```



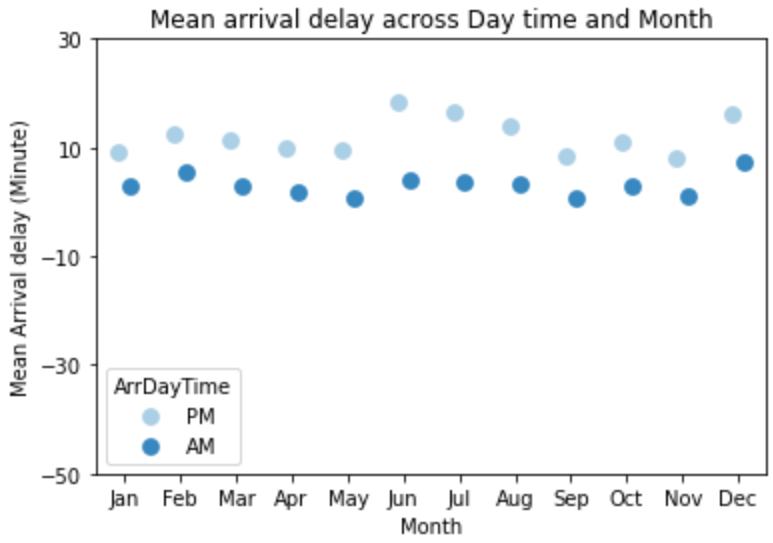
In [71]:
```
ax = sb.pointplot(data = df_flight, x = 'Year', y = 'ArrDelay', hue = 'ArrDayTime',
           palette = 'Blues', linestyles = '', dodge = 0.2)
plt.title('Mean arrival delay across Day time and Year')
plt.ylabel('Mean Arrival delay (Minute)')

x_min, x_max = -50, 50

plt.yticks(np.arange(x_min, x_max, 20));

plt.show();
```



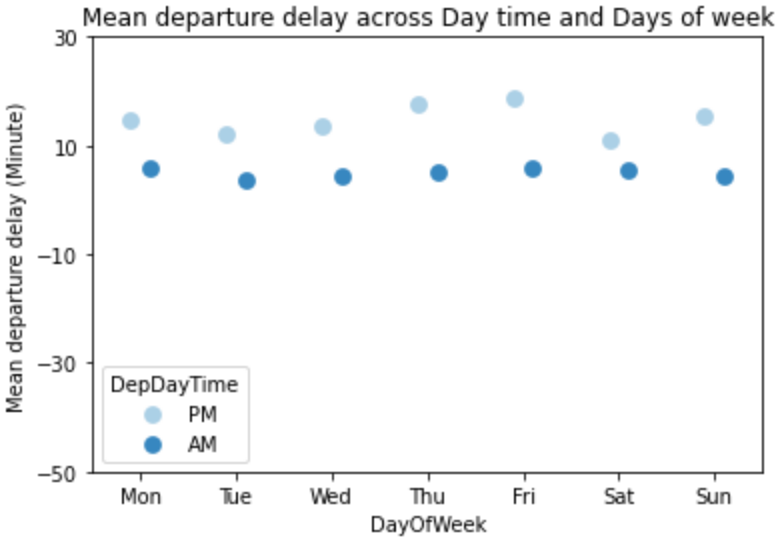
In [72]:
```
ax = sb.pointplot(data = df_flight, x = 'Month', y = 'DepDelay', hue = 'DepDayTime',
           palette = 'Blues', linestyles = '', dodge = 0.2)
plt.title('Mean departure delay across Day time and Month')
plt.ylabel('Mean Departure delay (Minute)')

x_min, x_max = -50, 50

plt.yticks(np.arange(x_min, x_max, 20));
```



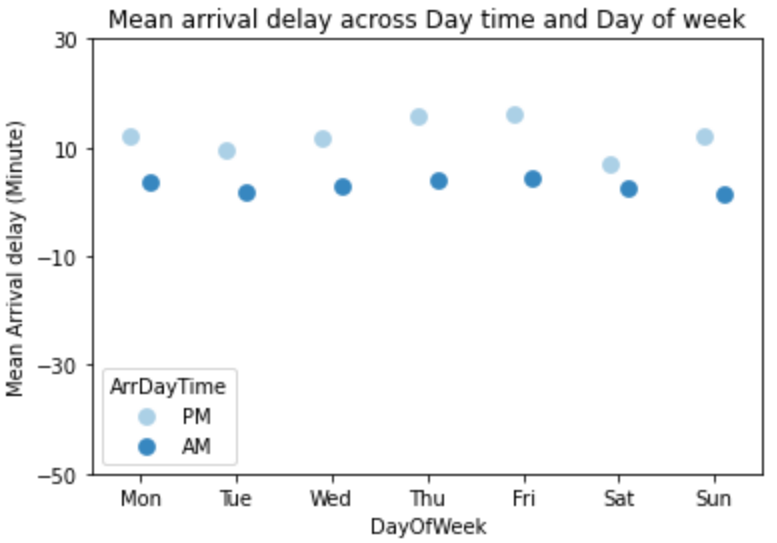
In [73]:
```
ax = sb.pointplot(data = df_flight, x = 'Month', y = 'ArrDelay', hue = 'ArrDayTime',
           palette = 'Blues', linestyles = '', dodge = 0.2)
plt.title('Mean arrival delay across Day time and Month')
plt.ylabel('Mean Arrival delay (Minute)')

x_min, x_max = -50, 50

plt.yticks(np.arange(x_min, x_max, 20));

plt.show();
```



In [74]:
```
ax = sb.pointplot(data = df_flight, x = 'DayOfWeek', y = 'DepDelay', hue = 'DepDayTime',
           palette = 'Blues', linestyles = '', dodge = 0.2)
plt.title('Mean departure delay across Day time and Days of week')
plt.ylabel('Mean departure delay (Minute)')

x_min, x_max = -50, 50

plt.yticks(np.arange(x_min, x_max, 20));

plt.show();
```



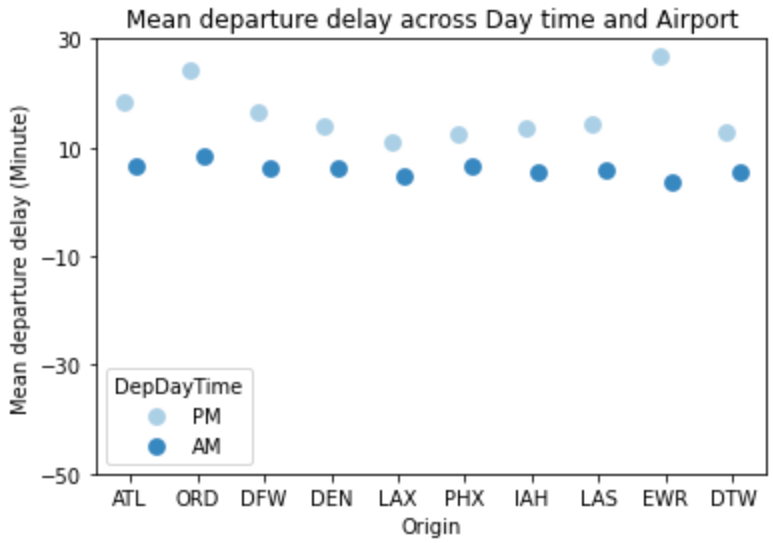
In [75]:
```
ax = sb.pointplot(data = df_flight, x = 'DayOfWeek', y = 'ArrDelay', hue = 'ArrDayTime',
           palette = 'Blues', linestyles = '', dodge = 0.2)
plt.title('Mean arrival delay across Day time and Day of week')
plt.ylabel('Mean Arrival delay (Minute)')

x_min, x_max = -50, 50

plt.yticks(np.arange(x_min, x_max, 20));

plt.show();
```



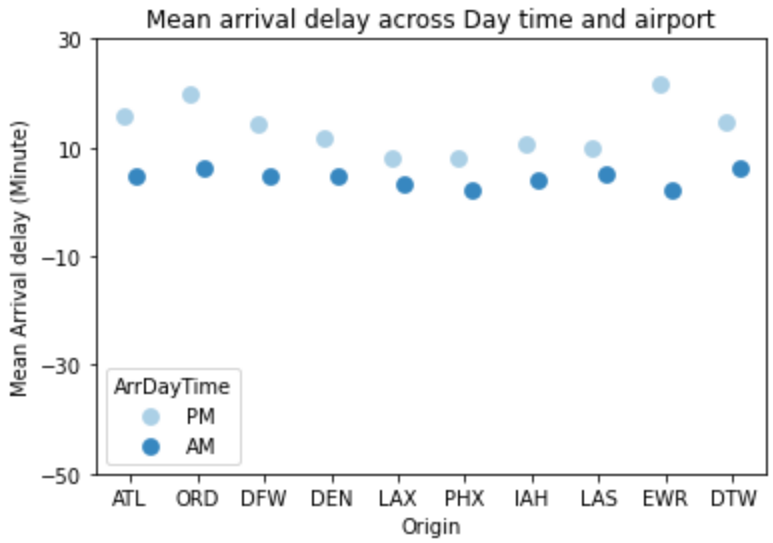
Observations: Delays across each month and year are prety same, with more delays in the evening flight. Same for days of the week, but for saturday, average delay for evening flight is the smallest one, and Sunday has the smallest average delay for morning flight.

In [76]:
```
#let's put busiest airports in a list
airports = df_flight.Origin.value_counts().head(10).index

#filter data on busiest airports
df_airport = df_flight.loc[df_flight.Origin.isin(airports)]


ordered_airport = pd.api.types.CategoricalDtype(ordered = True,
                                                categories = airports)

df_airport['Origin'] = df_airport['Origin'].astype(ordered_airport)
```

```
ax = sb.pointplot(data = df_airport, x = 'Origin', y = 'DepDelay', hue = 'DepDayTime',
           palette = 'Blues', linestyles = '', dodge = 0.2)
plt.title('Mean departure delay across Day time and Airport')
plt.ylabel('Mean departure delay (Minute)')

x_min, x_max = -50, 50

plt.yticks(np.arange(x_min, x_max, 20));

plt.show();
```



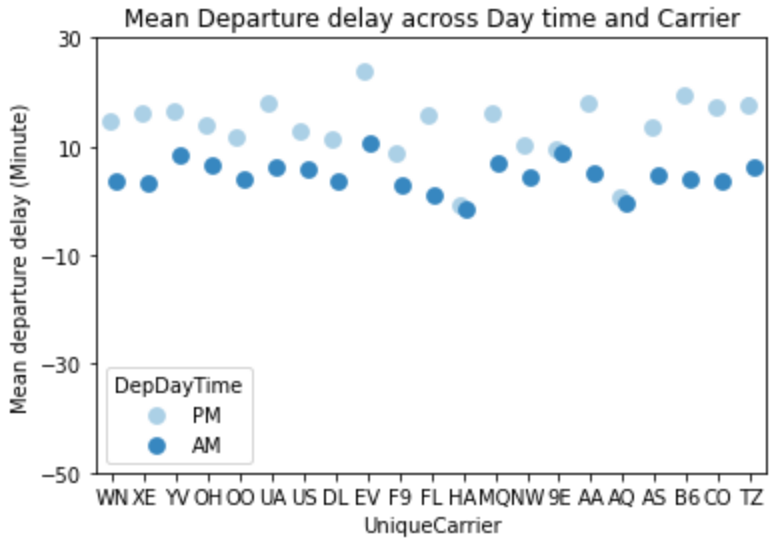
In [77]:
```
ax = sb.pointplot(data = df_airport, x = 'Origin', y = 'ArrDelay', hue = 'ArrDayTime',
           palette = 'Blues', linestyles = '', dodge = 0.2)
plt.title('Mean arrival delay across Day time and airport')
plt.ylabel('Mean Arrival delay (Minute)')

x_min, x_max = -50, 50

plt.yticks(np.arange(x_min, x_max, 20));

plt.show();
```



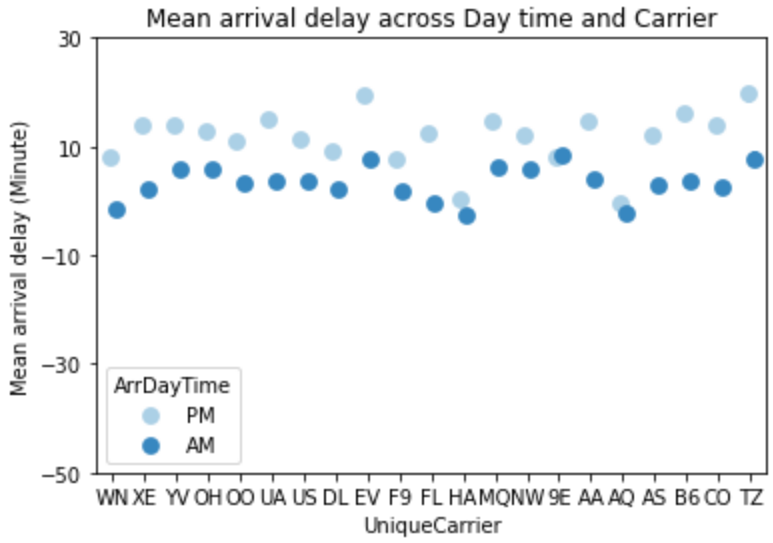
Observations: All top 10 busiest airport face more flight delay in the evening than morning. EWR being the airport with more evening flight delay, and still the airport with less moring flight delay.

In [78]:
```
ax = sb.pointplot(data = df_flight, x = 'UniqueCarrier', y = 'DepDelay', hue = 'DepDayTi
           palette = 'Blues', linestyles = '', dodge = 0.2)
plt.title('Mean Departure delay across Day time and Carrier')
```

```
plt.ylabel('Mean departure delay (Minute)')

x_min, x_max = -50, 50

plt.yticks(np.arange(x_min, x_max, 20));

plt.show();
```



In [79]:
```
ax = sb.pointplot(data = df_flight, x = 'UniqueCarrier', y = 'ArrDelay', hue = 'ArrDayTi
           palette = 'Blues', linestyles = '', dodge = 0.2)
plt.title('Mean arrival delay across Day time and Carrier')
plt.ylabel('Mean arrival delay (Minute)')

x_min, x_max = -50, 50

plt.yticks(np.arange(x_min, x_max, 20));

plt.show();
```

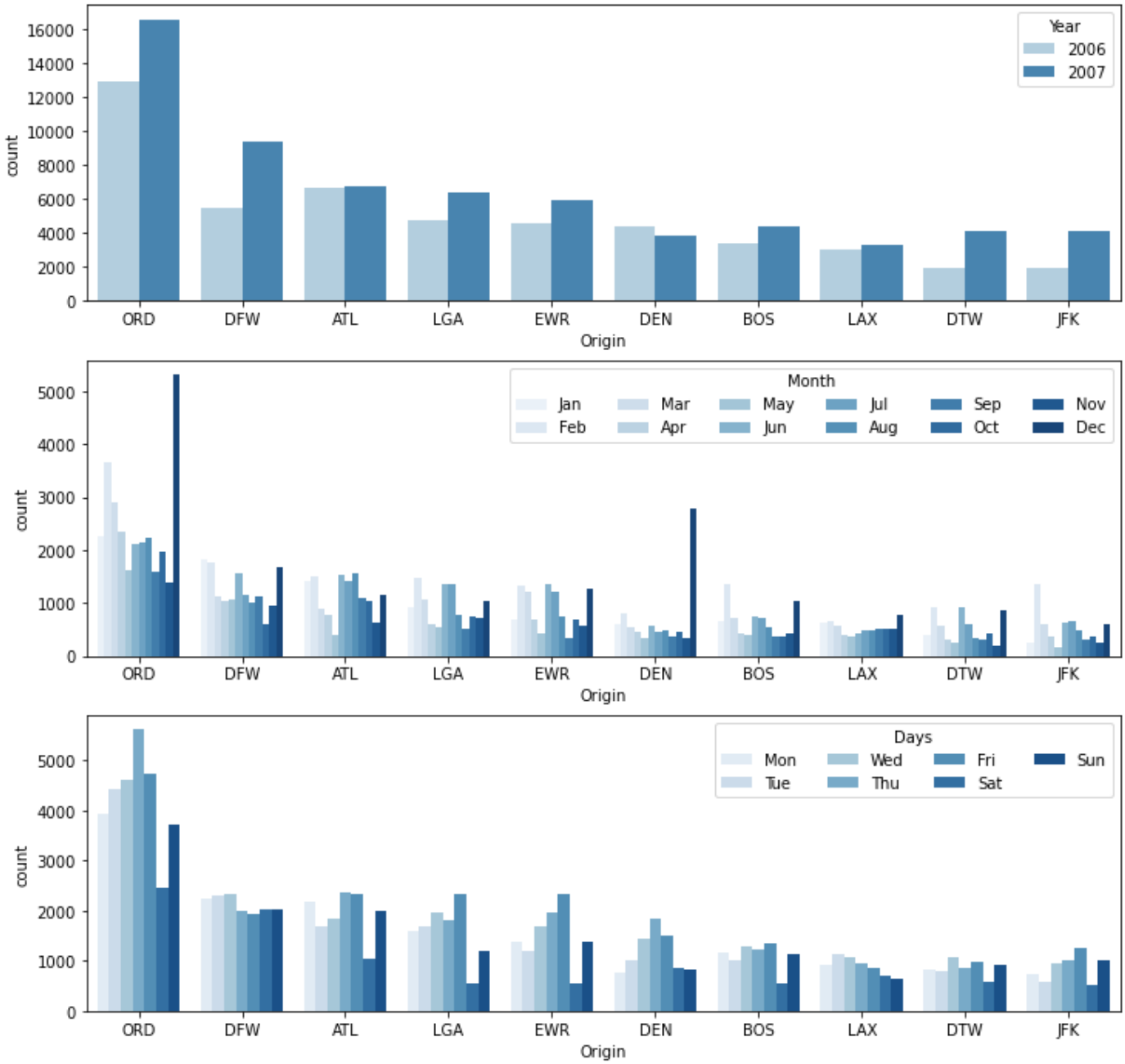


Observations: All unique carrier have more average delay on evening flight, except 9E, and AQ, where the average delay both in the moring and evening are prety same.

Relation between Cancelled, Diverted flight and airport

In [80]:
```
#let's put busiest airports in a list
df_cancel = df_flight[df_flight.Cancelled == 1]
airports_cancel = df_cancel.Origin.value_counts().head(10).index
```

```
#filter data on busiest airports
df_airport_cancel = df_cancel.loc[df_cancel.Origin.isin(airports_cancel)]


ordered_airport = pd.api.types.CategoricalDtype(ordered = True,
                                                categories = airports_cancel)

df_airport_cancel['Origin'] = df_airport_cancel['Origin'].astype(ordered_airport)

# since there's only two subplots to create, using the full data should be fine.
plt.figure(figsize = [12, 12])

# subplot 1: ArrDayTime vs ArrDelay_cat
plt.subplot(3, 1, 1)
sb.countplot(data = df_airport_cancel, x = 'Origin', hue = 'Year', palette = 'Blues')

# subplot 2: DepDayTime vs. DepDelay_cat
ax = plt.subplot(3, 1, 2)
sb.countplot(data = df_airport_cancel, x = 'Origin', hue = 'Month', palette = 'Blues')
ax.legend(ncol = 6, title='Month') # re-arrange legend to reduce overlapping

# subplot 2: DepDayTime vs. DepDelay_cat
ax = plt.subplot(3, 1, 3)
sb.countplot(data = df_airport_cancel, x = 'Origin', hue = 'DayOfWeek', palette = 'Blues
ax.legend(ncol = 4, title='Days') # re-arrange legend to reduce overlapping

plt.show()
```

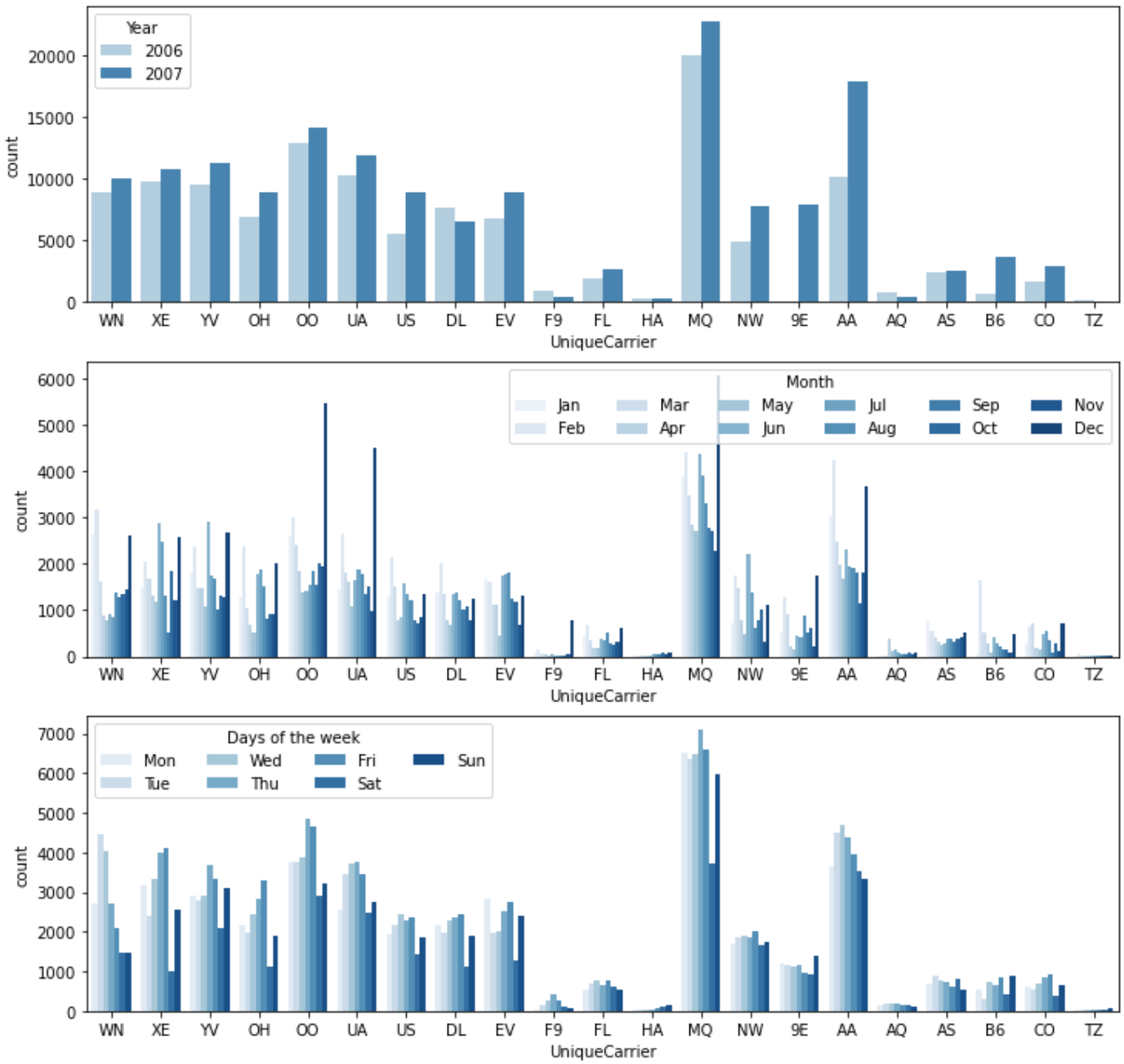


Observations: ORD which was the second busiest airport, is the airport with more cancelled flights. All top 10 cancelled flights airport, has more cancelled flight in 2007 than 2006, except DEN airport. ATL which is the busiest flight airport is amongst top 10 cancelled flight airport. December is the month with more cancelled flight at these 10 airports.

Relationship between Cancelled, carrier and date

In [81]:
```
# since there's only two subplots to create, using the full data should be fine.
plt.figure(figsize = [12, 12])

# subplot 1: ArrDayTime vs ArrDelay_cat
plt.subplot(3, 1, 1)
sb.countplot(data = df_cancel, x = 'UniqueCarrier', hue = 'Year', palette = 'Blues')

# subplot 2: DepDayTime vs. DepDelay_cat
ax = plt.subplot(3, 1, 2)
sb.countplot(data = df_cancel, x = 'UniqueCarrier', hue = 'Month', palette = 'Blues')
ax.legend(ncol = 6, title='Month') # re-arrange legend to reduce overlapping

# subplot 2: DepDayTime vs. DepDelay_cat
ax = plt.subplot(3, 1, 3)
```

```
sb.countplot(data = df_cancel, x = 'UniqueCarrier', hue = 'DayOfWeek', palette = 'Blues'
ax.legend(ncol = 4, title='Days of the week') # re-arrange legend to reduce overlapping

plt.show()
```

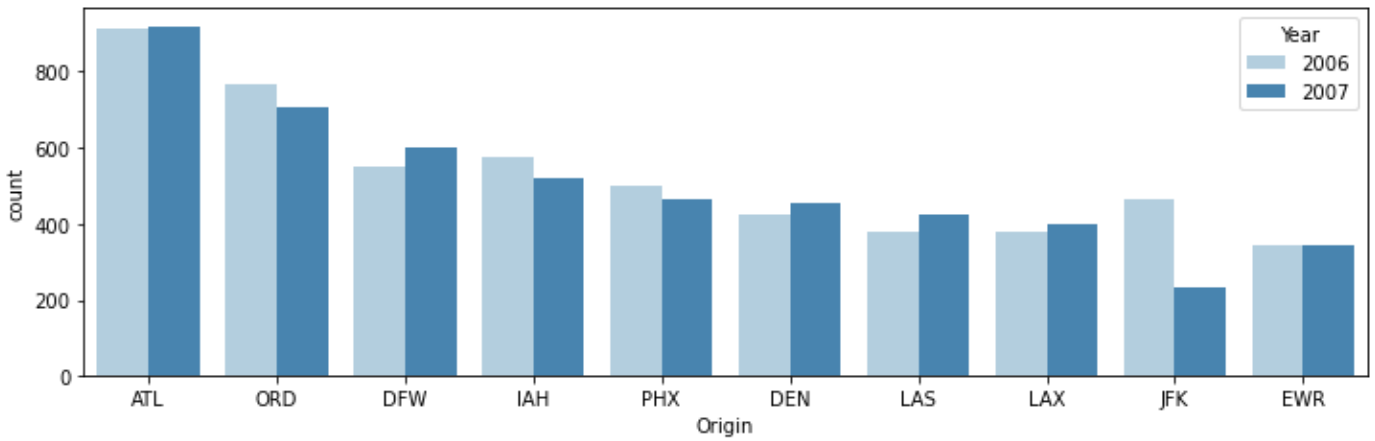


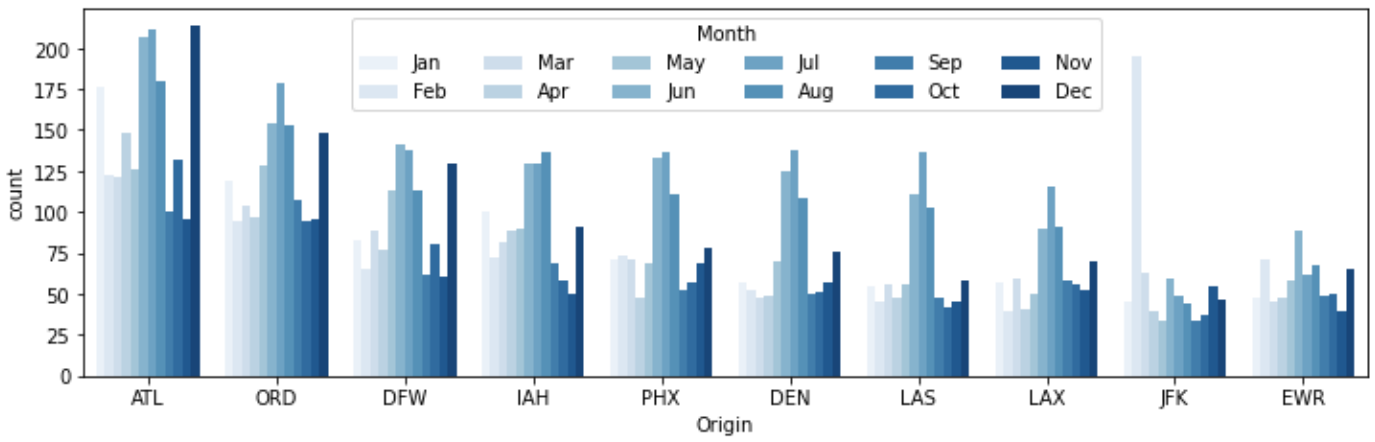
Observations: Number of cancelled flight per carrier varies a lot. There are more cancelled flight in 2017 than 2006. MQ is the carrier with more cancelled flights. December is the month with more cancelled flight for almost all carrier. Saturday seems to be the day with less cancelled flight for almost all carrier.

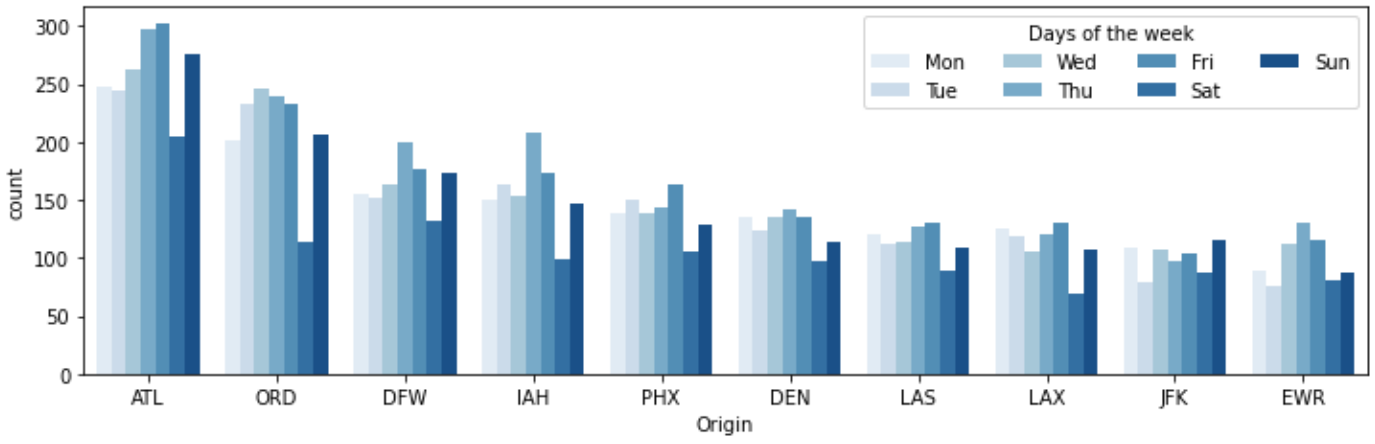
Relationship between Diverted, airport and date.

In [82]:
```
#let's put busiest airports in a list
df_divert = df_flight[df_flight.Diverted == 1]
airports_divert = df_divert.Origin.value_counts().head(10).index

#filter data on busiest airports
df_airport_divert = df_divert.loc[df_divert.Origin.isin(airports_divert)]


ordered_airport = pd.api.types.CategoricalDtype(ordered = True,
```

```
                                                categories = airports_divert)

df_airport_divert['Origin'] = df_airport_divert['Origin'].astype(ordered_airport)

# since there's only two subplots to create, using the full data should be fine.
plt.figure(figsize = [12, 12])

# subplot 1: ArrDayTime vs ArrDelay_cat
plt.subplot(3, 1, 1)
sb.countplot(data = df_airport_divert, x = 'Origin', hue = 'Year', palette = 'Blues')

# subplot 2: DepDayTime vs. DepDelay_cat
ax = plt.subplot(3, 1, 2)
sb.countplot(data = df_airport_divert, x = 'Origin', hue = 'Month', palette = 'Blues')
ax.legend(ncol = 6, title='Month') # re-arrange legend to reduce overlapping

# subplot 2: DepDayTime vs. DepDelay_cat
ax = plt.subplot(3, 1, 3)
sb.countplot(data = df_airport_divert, x = 'Origin', hue = 'DayOfWeek', palette = 'Blues
ax.legend(ncol = 4, title='Days of the week') # re-arrange legend to reduce overlapping

plt.show()
```

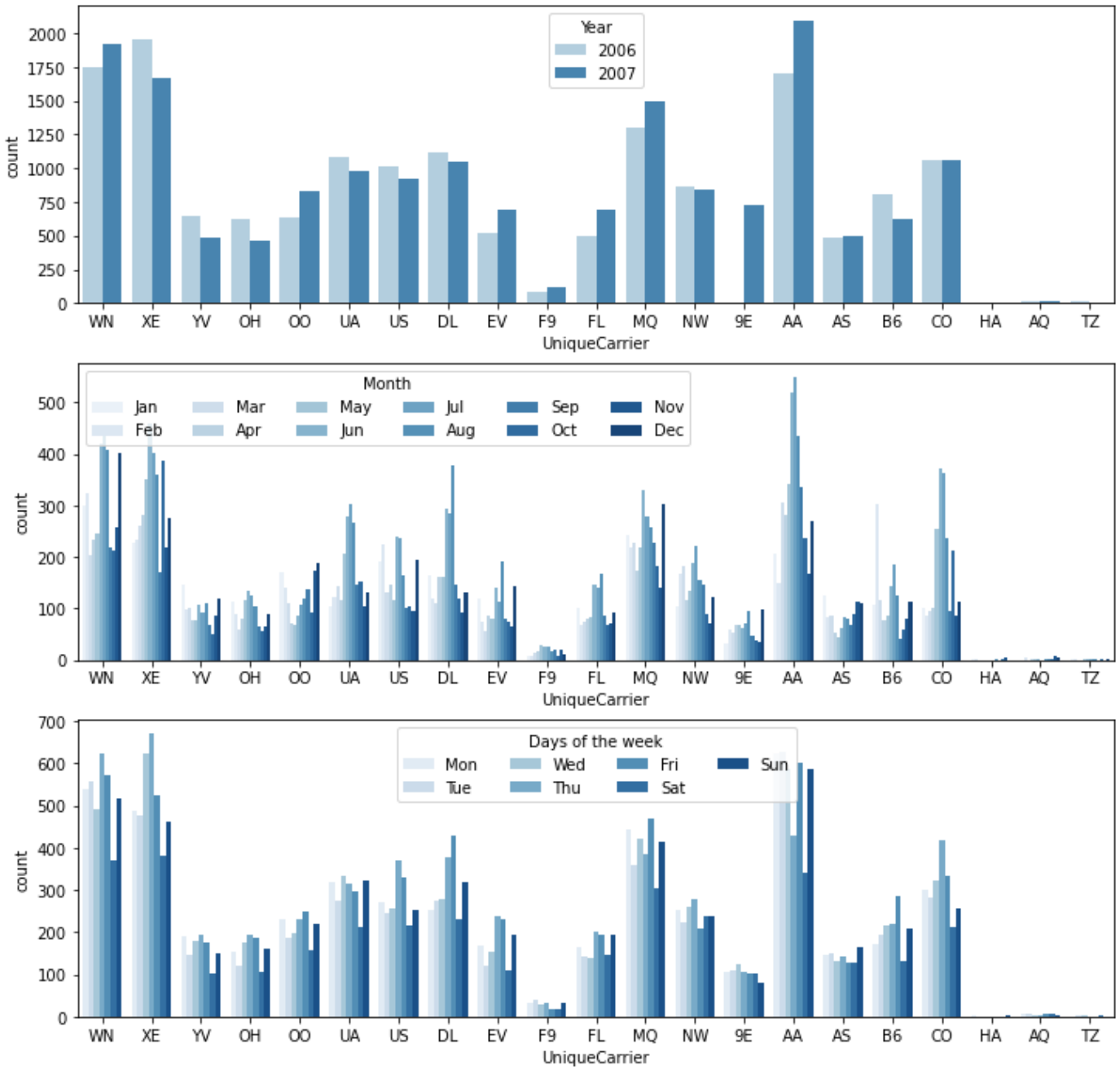






Observations: ATL which was the most busiest airport, is the airport with more cancelled flights. Some airport (ATL, DFW, DEN, LAS and LAX) amongst top 10 diverted flights airport has more diverted flight in 2007, while others (ORD, IAH, PHX, JFK) has more diverted flights in 2006.

In [83]:
```
# since there's only two subplots to create, using the full data should be fine.
plt.figure(figsize = [12, 12])

# subplot 1: ArrDayTime vs ArrDelay_cat
plt.subplot(3, 1, 1)
sb.countplot(data = df_divert, x = 'UniqueCarrier', hue = 'Year', palette = 'Blues')

# subplot 2: DepDayTime vs. DepDelay_cat
ax = plt.subplot(3, 1, 2)
sb.countplot(data = df_divert, x = 'UniqueCarrier', hue = 'Month', palette = 'Blues')
ax.legend(ncol = 6, title='Month') # re-arrange legend to reduce overlapping

# subplot 2: DepDayTime vs. DepDelay_cat
ax = plt.subplot(3, 1, 3)
sb.countplot(data = df_divert, x = 'UniqueCarrier', hue = 'DayOfWeek', palette = 'Blues'
ax.legend(ncol = 4, title='Days of the week') #

plt.show()
```



Observations: Number of diverted flight per carrier varies a lot per year, month and day of week. AA is the carrier with highest number of diverted flight in 2007, while XE is the carrier with more diverted flight in 2006.

Talk about some of the relationships you observed in this part of the investigation. Were there features that strengthened each other in terms of looking at your feature(s) of interest?

> Multivariate plot just confirm what we saw in the bivariate plot, where most delaying flights are flights scheduled in the evening. It also shows there is no trends with airports or carrier. While some busiest flight airport are subject to delay other busiests airport are less subject to delay. Same with carrier. It also shows that There is no really change of delay in year, month and day of week.

Were there any interesting or surprising interactions between features?

> ORD which was the second busiest airport, is the airport with more cancelled flights. All top 10 cancelled flights airport, has more cancelled flight in 2007 than 2006, except DEN airport. ATL which is the busiest flight airport is amongst top 10 cancelled flight airport. Number of cancelled flight per carrier varies a lot. MQ is the carrier with more cancelled flights. December is the month with more cancelled flight for almost all carrier and airport. Saturday seems to be the day with less cancelled flight for almost all carrier.

Conclusions

> The majority of departure and arrival delays are short (earlier departures (and arrivals), 20 minutes before the scheduled time and late departure (and arrivals) less than 70 minutes), and the longest delays, while unusual, are more heavy loaded in time. We dsicover that ArrDelay is highly correlated with DepDelay which is normal. There is a litle variation of delay with month where busiest flight months like August, July and December seem to have more average flight delay. Is it also true for days of the week, where Saturday which was the less busiest flight day also has less average delay. We also discover that flights with higher average delay are flights scheduled in the evening(PM). There is no trend between delay and busiest airport. Amongst the busiest airports, ORD and EWR have more average flight delay than the busiest aiport (ATL). Is it also true with the unique carrier, EV and TZ which are not the most busiest carriers has the highest average delay, while WN which is the most busiest carrier is in the middle with an average of 5 minutes delay in arrival and 10 minutes in departure.
>
> Cancelled and Diverted are not really correlated with any other variable, but cancelled and diverted flights are mostly flights scheduled in the evening (PM). The Variation of number of flight per month and year is prety the same for each top 10 busiest airport. ORD is the airport with more cancelled flights. MQ is the carrier with more cancelled flights. December is the month with more cancelled flight for almost all carrier and airport. Saturday seems to be the day with less cancelled flight for almost all carrier.

In [ ]: