In [37]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
```

In [38]:
```
df = pd.read_csv('Downloads/2008.csv')
df.head()
```

Out[38]:

	Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarrier	FlightNum	...	TaxiIn	TaxiOut
0	2008	1	3	4	2003.0	1955	2211.0	2225	WN	335	...	4.0	8.0
1	2008	1	3	4	754.0	735	1002.0	1000	WN	3231	...	5.0	10.0
2	2008	1	3	4	628.0	620	804.0	750	WN	448	...	3.0	17.0
3	2008	1	3	4	926.0	930	1054.0	1100	WN	1746	...	3.0	7.0
4	2008	1	3	4	1829.0	1755	1959.0	1925	WN	3920	...	3.0	10.0

5 rows × 29 columns

In [51]:
```
print('Dataframe dimensions:', df.shape)
#cleaning of data
#converting the time which is in minutes into format hh:mm
df['Date'] = pd.to_datetime(df.Year.map(str)+'-'+df.Month.map(str)+'-'+df.DayofMonth.map(str))
```

```
Dataframe dimensions: (7009728, 29)
```

In [52]:
```
# Function that convert the 'HHMM' string to datetime.time
def format_heure(chaine):
    if pd.isnull(chaine):
        return np.nan
    else:
        if chaine == 2400: chaine = 0
        chaine = "{0:04d}".format(int(chaine))
        heure = datetime.time(int(chaine[0:2]), int(chaine[2:4]))
        return heure
#----------------------------------------------------------------------
# Function that combines a date and time to produce a datetime.datetime
def combine_date_heure(x):
    if pd.isnull(x[0]) or pd.isnull(x[1]):
        return np.nan
    else:
        return datetime.datetime.combine(x[0],x[1])
#----------------------------------------------------------------------
# Function that combine two columns of the dataframe to create a datetime format
def create_flight_time(df, col):
    liste = []
    for index, cols in df[['Date', col]].iterrows():
        if pd.isnull(cols[1]):
            liste.append(np.nan)
        elif float(cols[1]) == 2400:
            cols[0] += datetime.timedelta(days=1)
            cols[1] = datetime.time(0,0)
            liste.append(combine_date_heure(cols))
        else:
            cols[1] = format_heure(cols[1])
            liste.append(combine_date_heure(cols))
    return pd.Series(liste)
```

In [53]:
```
df['CRSDepTime'] = create_flight_time(df, 'CRSDepTime')
df['DepTime'] = df['DepTime'].apply(format_heure)
df['CRSArrTime'] = df['CRSArrTime'].apply(format_heure)
df['ArrTime'] = df['ArrTime'].apply(format_heure)
```

In [54]:
```
df.head(3).T
```

Out[54]:

	0	1	2
Year	2008	2008	2008
Month	1	1	1
DayofMonth	3	3	3
DayOfWeek	4	4	4
DepTime	20:03:00	07:54:00	06:28:00
CRSDepTime	2008-01-03 19:55:00	2008-01-03 07:35:00	2008-01-03 06:20:00
ArrTime	22:11:00	10:02:00	08:04:00
CRSArrTime	22:25:00	10:00:00	07:50:00
UniqueCarrier	WN	WN	WN
FlightNum	335	3231	448
TailNum	N712SW	N772SW	N428WN
ActualElapsedTime	128	128	96
CRSElapsedTime	150	145	90
AirTime	116	113	76
ArrDelay	-14	2	14
DepDelay	8	19	8
Origin	IAD	IAD	IND
Dest	TPA	TPA	BWI
Distance	810	810	515
TaxiIn	4	5	3
TaxiOut	8	10	17
Cancelled	0	0	0
CancellationCode	NaN	NaN	NaN
Diverted	0	0	0
CarrierDelay	NaN	NaN	NaN
WeatherDelay	NaN	NaN	NaN

	0	1	2
NASDelay	NaN	NaN	NaN
SecurityDelay	NaN	NaN	NaN
LateAircraftDelay	NaN	NaN	NaN
Date	2008-01-03 00:00:00	2008-01-03 00:00:00	2008-01-03 00:00:00

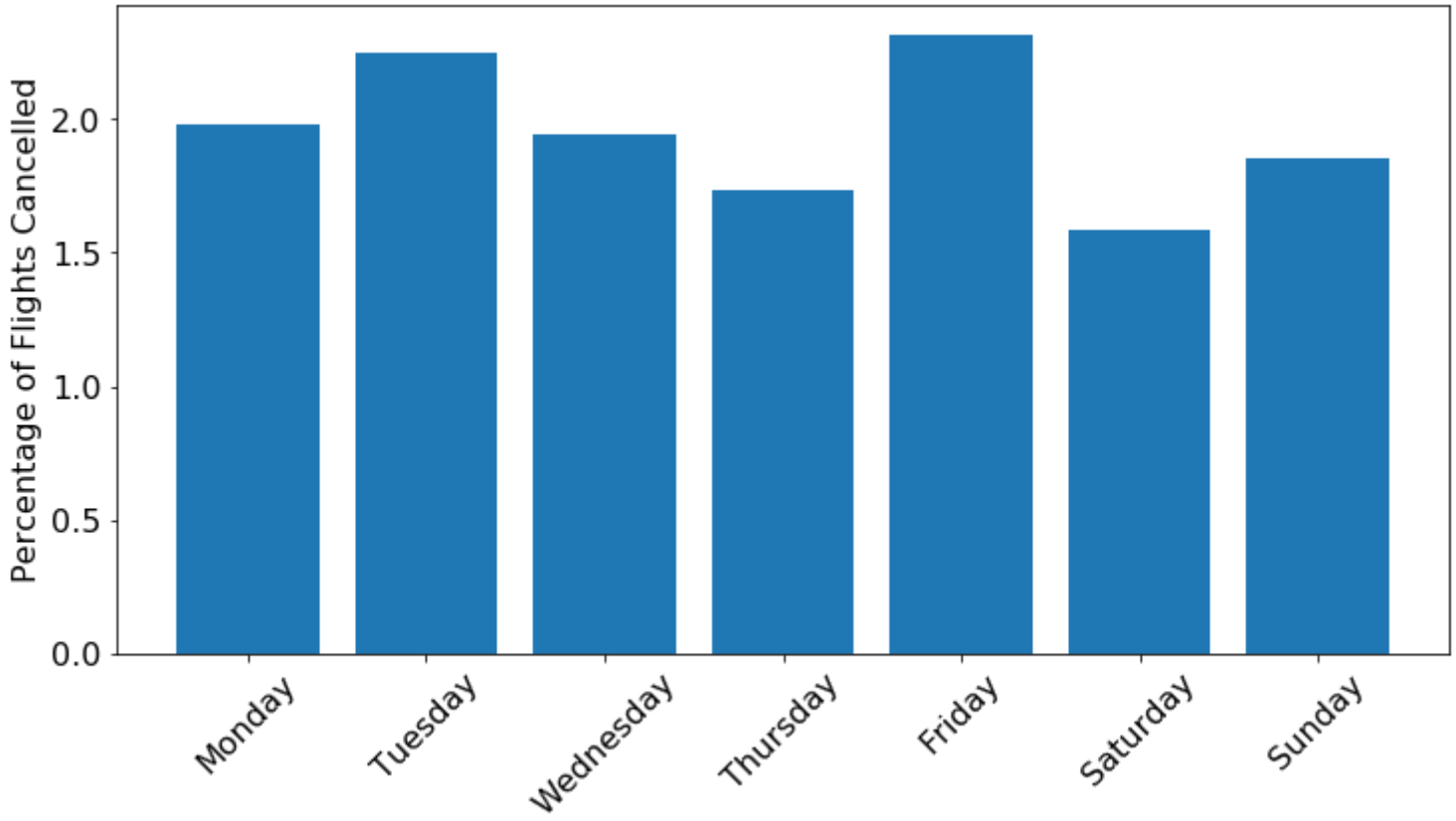
In [59]:
```
df['TaxiOut'].fillna(0, inplace=True)
cancelled = df[df['Cancelled']==1]
cancelled.tail()
```

Out[59]:

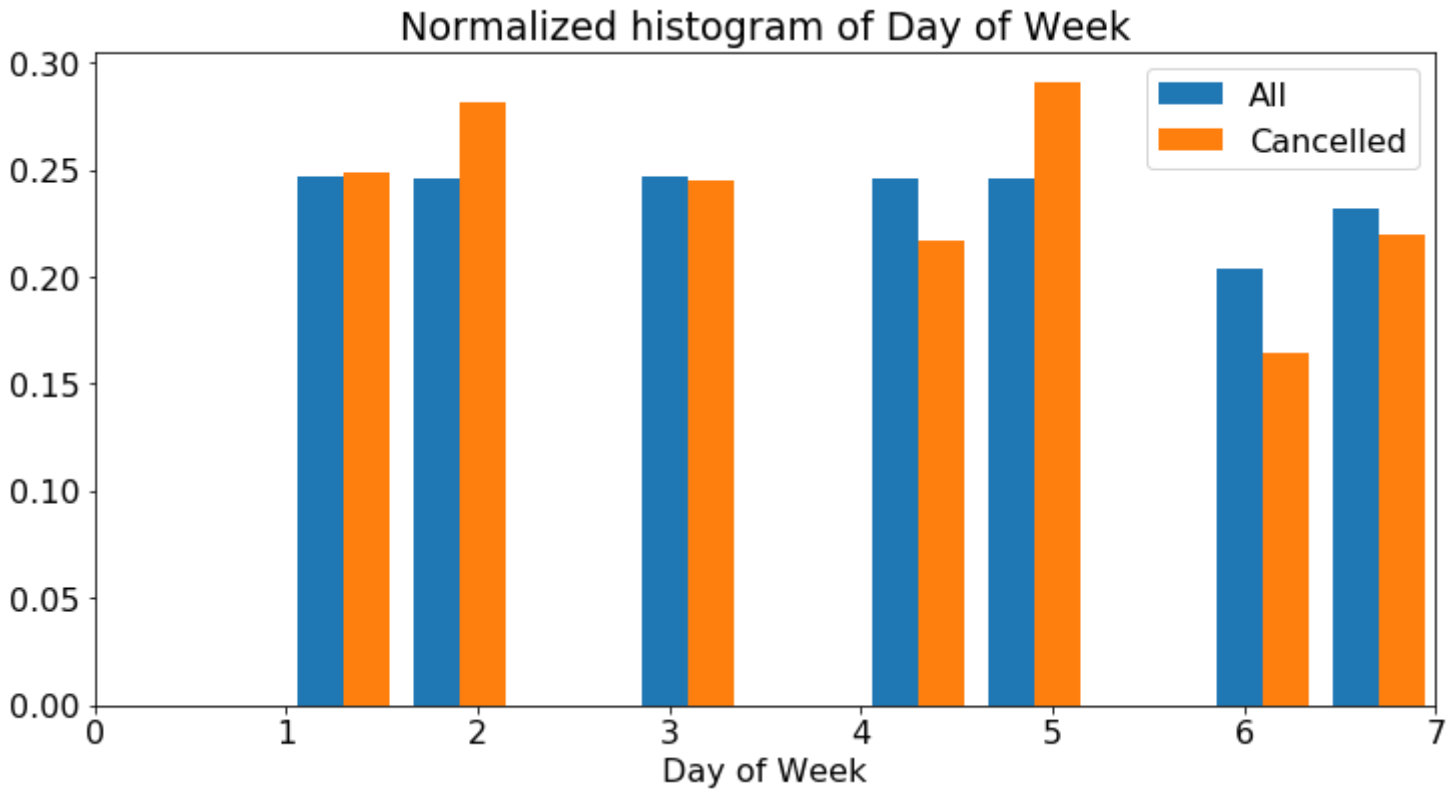
	Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarrier	FlightNum	...	TaxiOut
7009455	2008	12	13	6	NaN	2008-12-13 06:00:00	NaN	08:15:00	DL	1211	...	0.0
7009464	2008	12	13	6	NaN	2008-12-13 19:30:00	NaN	21:29:00	DL	1218	...	0.0
7009564	2008	12	13	6	NaN	2008-12-13 07:00:00	NaN	10:35:00	DL	1421	...	0.0
7009565	2008	12	13	6	NaN	2008-12-13 11:15:00	NaN	14:32:00	DL	1422	...	0.0
7009648	2008	12	13	6	NaN	2008-12-13 10:20:00	NaN	11:26:00	DL	1532	...	0.0

5 rows × 30 columns

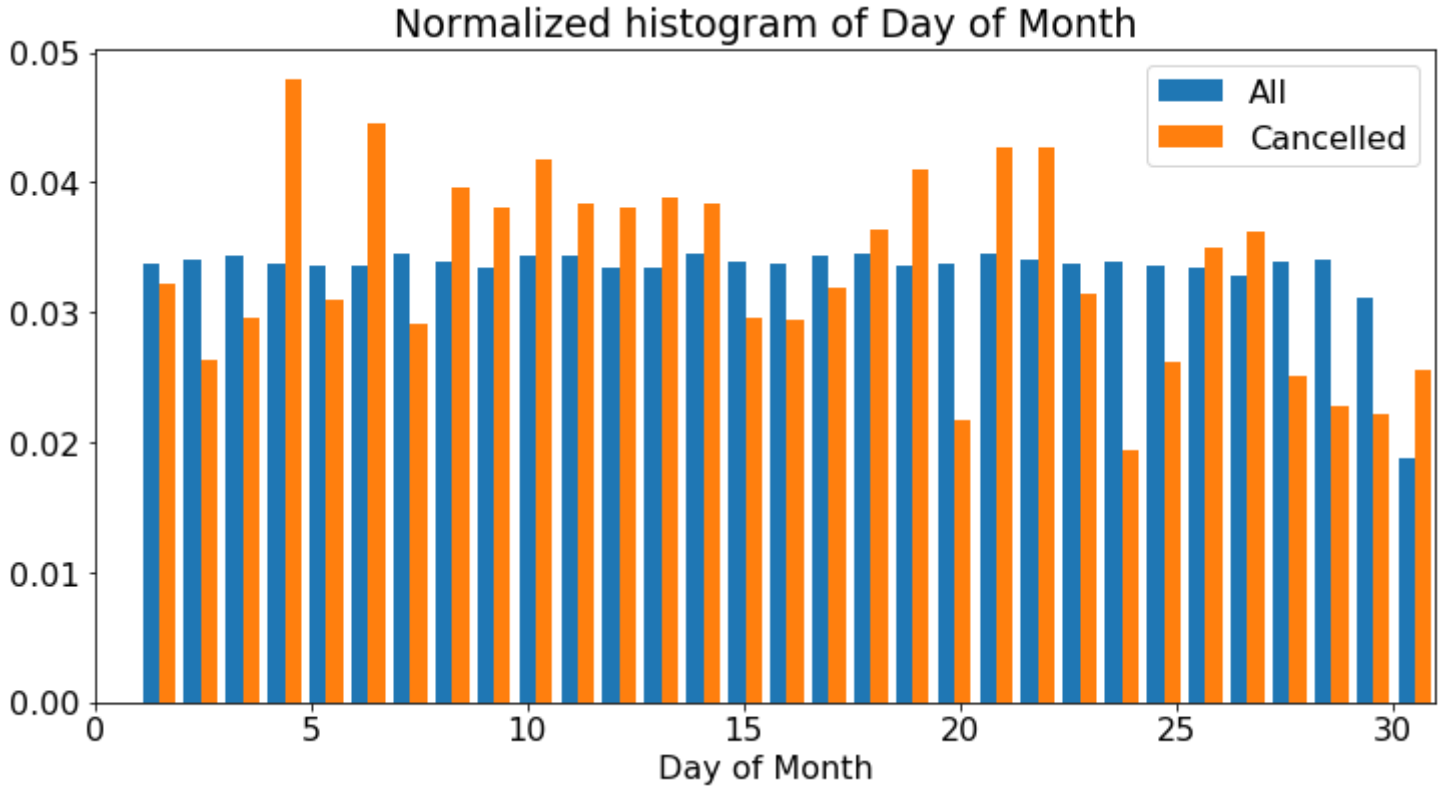
In [61]:
```
font = {'size'   : 16}
plt.rc('font', **font)

days_cancelled = cancelled['Cancelled'].groupby(df['DayOfWeek']).count()
days_total = df['Cancelled'].groupby(df['DayOfWeek']).count()
days_frac = np.divide(days_cancelled, days_total)
x=days_frac.index.values
week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

fig, ax = plt.subplots(figsize = (12,6))
ax.bar(x,days_frac*100, align='center')
ax.set_ylabel('Percentage of Flights Cancelled')
ax.set_xticks(x)
ax.set_xticklabels(week, rotation = 45)

plt.show()
```



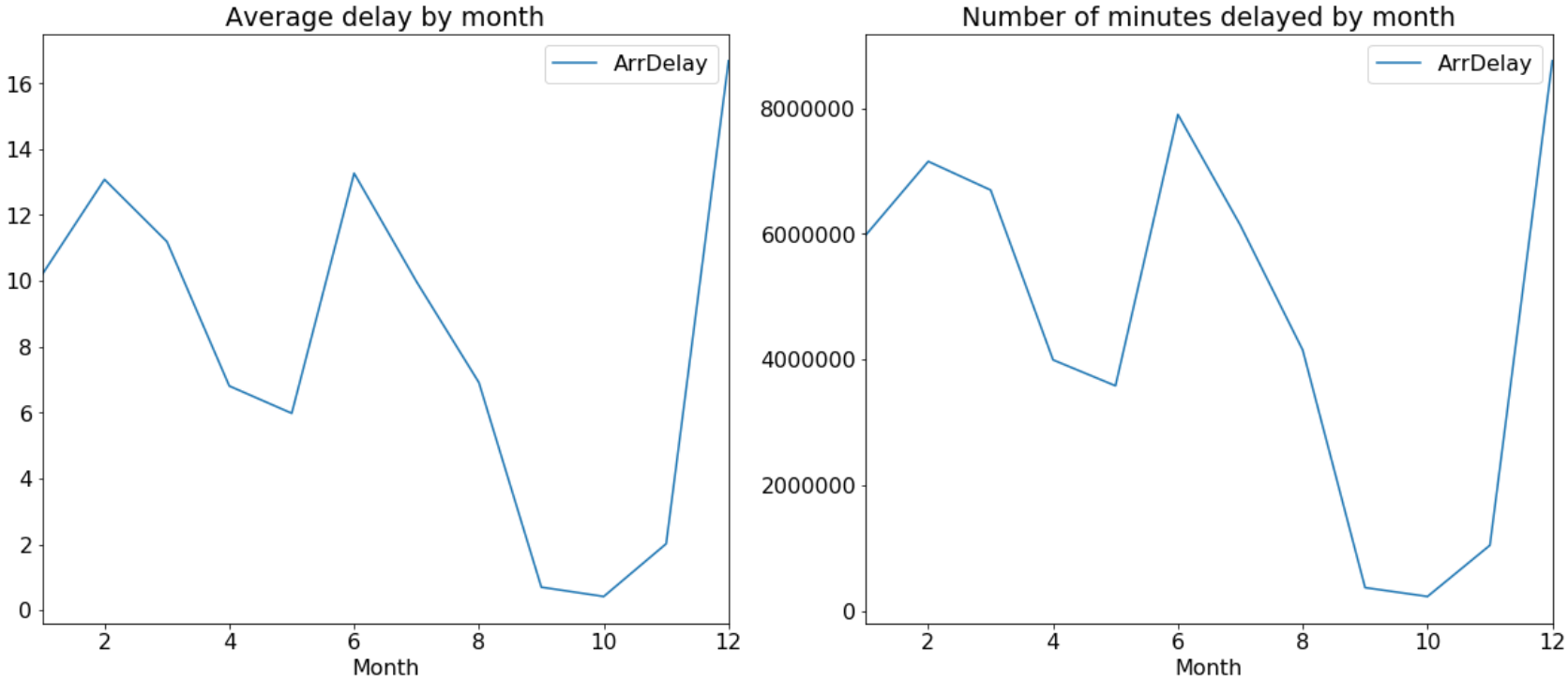
In [88]:
```
fig, ax = plt.subplots(figsize = (12,6))

ax.hist([df['DayOfWeek'], cancelled['DayOfWeek']], density=20, label=['All', 'Cancelled'],align='mid'
)
ax.set_xlim(0,7)
ax.set_xlabel('Day of Week')
ax.set_title('Normalized histogram of Day of Week')

plt.legend()
plt.show()
```



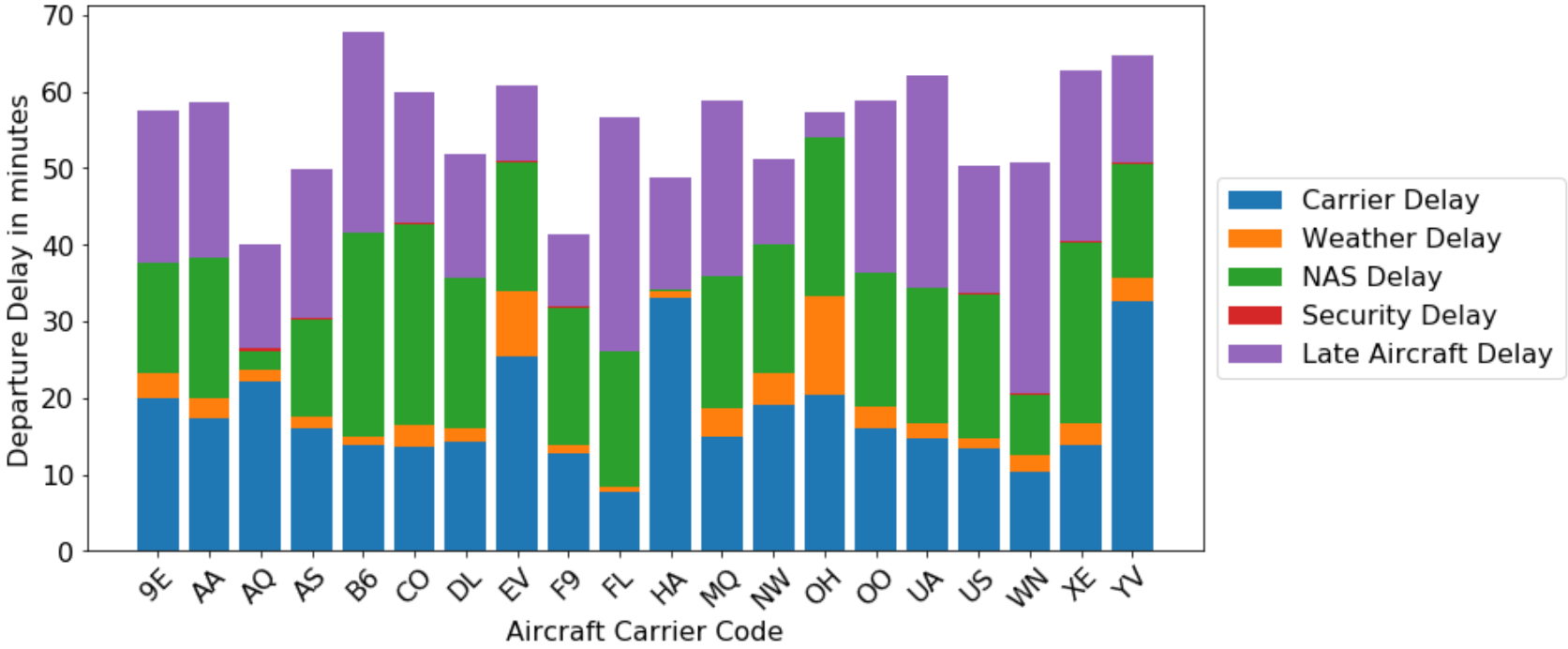
In [73]:
```
fig, ax = plt.subplots(figsize = (12,6))

ax.hist([df['DayofMonth'], cancelled['DayofMonth']], density=1, bins=31, label=['All', 'Cancelled'])

ax.set_xlim(0,31)

ax.set_xlabel('Day of Month')
ax.set_title('Normalized histogram of Day of Month')

plt.legend()
plt.show()
```



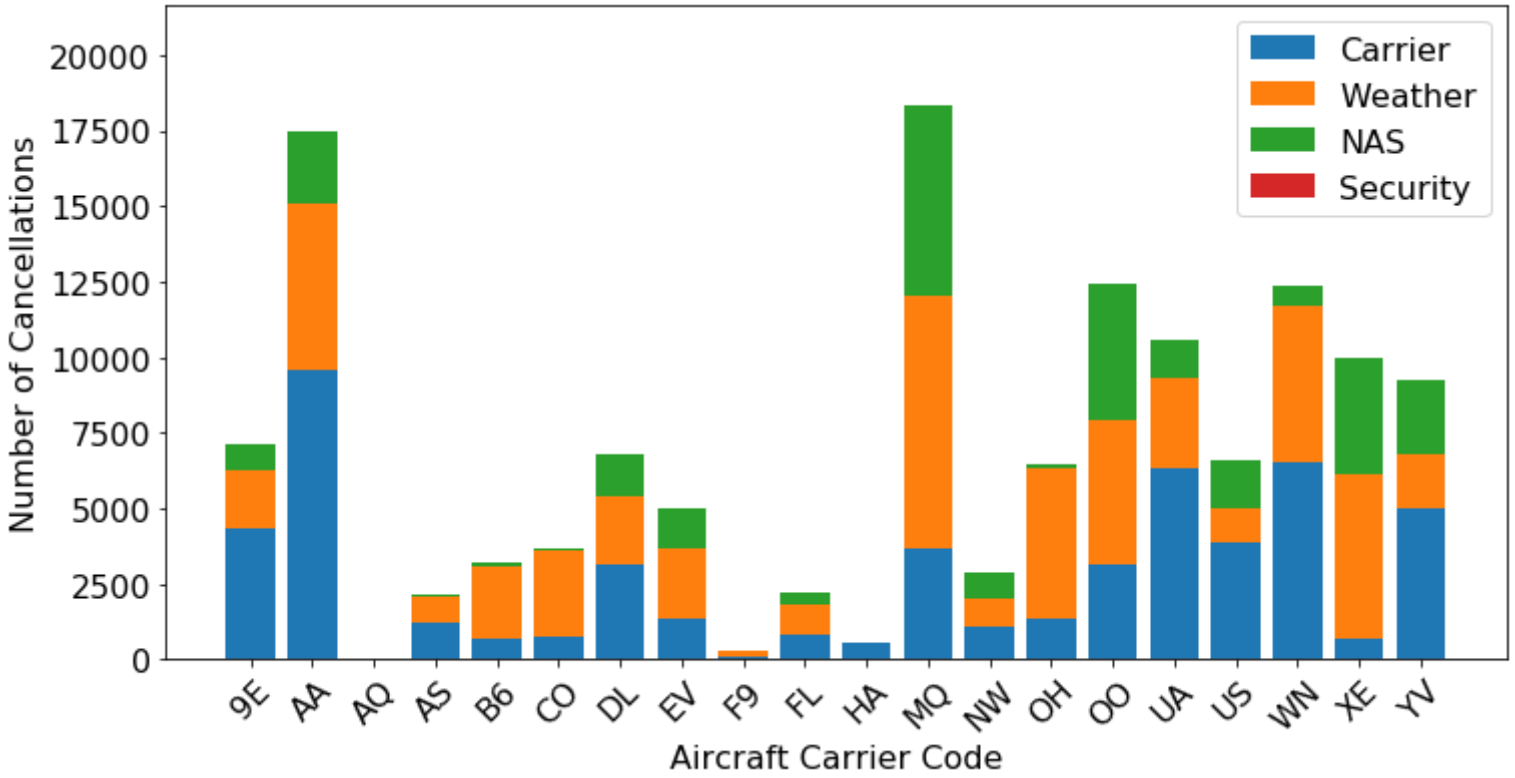
In [115]:
```
f,ax=plt.subplots(1,2,figsize=(20,8))
df[['Month','ArrDelay']].groupby(['Month']).mean().plot(ax=ax[0])
ax[0].set_title('Average delay by month')
df[['Month','ArrDelay']].groupby(['Month']).sum().plot(ax=ax[1])
ax[1].set_title('Number of minutes delayed by month')
plt.show()
```

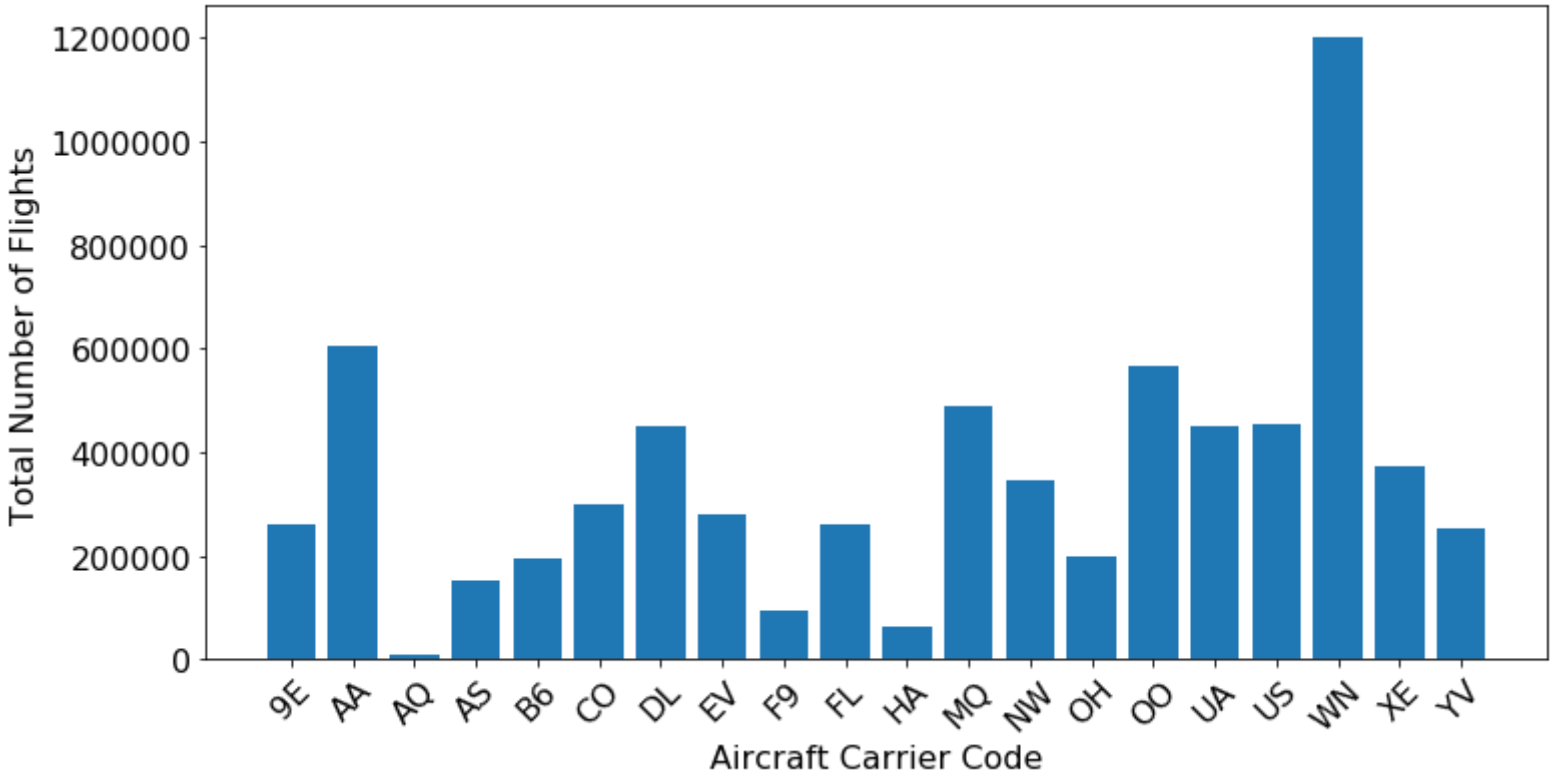


In [79]:
```
df['total_delay'] = (df['CarrierDelay'] + df['WeatherDelay']
             + df['NASDelay'] + df['SecurityDelay'] + df['LateAircraftDelay'])

df_delayed = df[~np.isnan(df['total_delay'])]
df['total_delay'].fillna(0, inplace=True)
df_delayed.head()

carrier_group = df_delayed['CarrierDelay'].groupby(df_delayed['UniqueCarrier']).mean()
weather_group = df_delayed['WeatherDelay'].groupby(df_delayed['UniqueCarrier']).mean()
nas_group = df_delayed['NASDelay'].groupby(df_delayed['UniqueCarrier']).mean()
security_group = df_delayed['SecurityDelay'].groupby(df_delayed['UniqueCarrier']).mean()
late_group = df_delayed['LateAircraftDelay'].groupby(df_delayed['UniqueCarrier']).mean()

w_bottom = carrier_group.values
n_bottom = w_bottom + weather_group.values
s_bottom = n_bottom + nas_group.values
l_bottom = s_bottom + security_group.values

x = carrier_group.index.values

fig, ax = plt.subplots(figsize = (12,6))

ax.set_xticks(np.arange(len(x)))
ax.set_xticklabels(x, rotation = 45)
ax.bar(np.arange(len(x)),carrier_group.values, align='center', label='Carrier Delay')
ax.bar(np.arange(len(x)),weather_group.values, align='center', bottom=w_bottom, label='Weather Delay'
)
ax.bar(np.arange(len(x)),nas_group.values, align='center', bottom=n_bottom, label='NAS Delay')
ax.bar(np.arange(len(x)),security_group.values, align='center', bottom=s_bottom, label='Security Dela
y')
ax.bar(np.arange(len(x)),late_group.values, align='center', bottom=l_bottom, label='Late Aircraft Del
ay')

ax.set_xlabel('Aircraft Carrier Code')
ax.set_ylabel('Departure Delay in minutes')

plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.show()
```

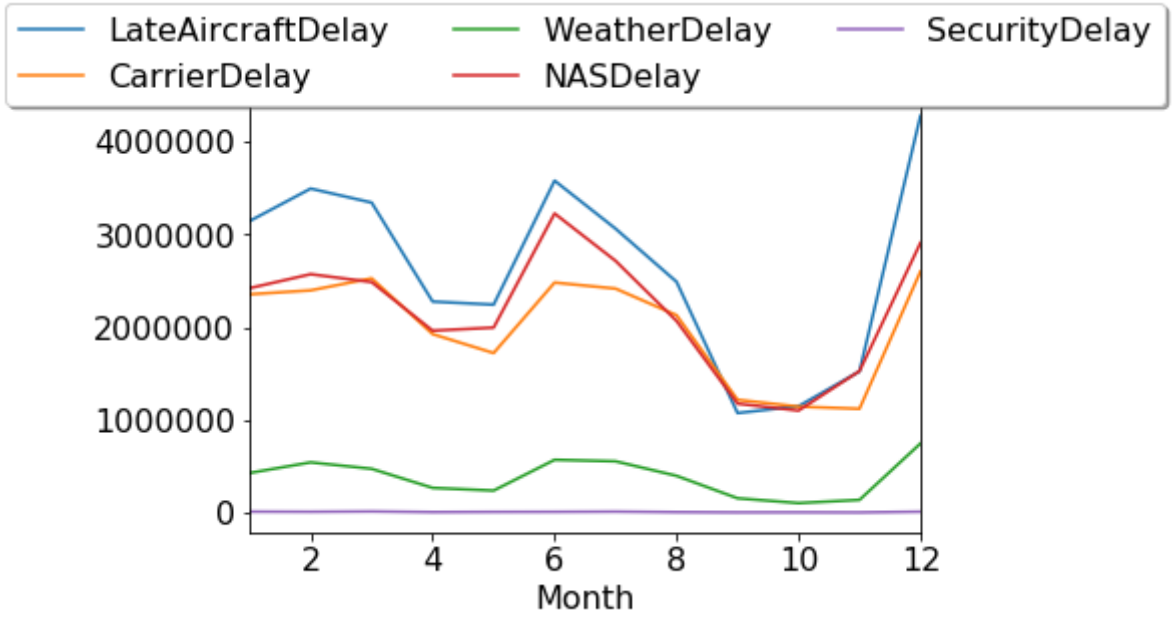

Departure Delay in minutes
0
10
20
30
40
50
60
70
9E
AA
AQ
AS
B6
CO
DL
EV
F9
FL
HA
MQ
NW
OH
OO
UA
US
WN
XE
YV
Aircraft Carrier Code
Carrier Delay
Weather Delay
NAS Delay
Security Delay
Late Aircraft Delay

In [98]:
```
cancelled_group = cancelled.groupby(['UniqueCarrier','CancellationCode']).size().reindex(fill_value=
0.0).unstack()
cg = cancelled_group.fillna(0)

b_bottom = cg.loc[:,'A'].values
c_bottom = b_bottom + cg.loc[:,'B'].values
d_bottom = c_bottom + cg.loc[:,'B'].values

x = cg.loc[:,'A'].index.values

fig, ax = plt.subplots(figsize = (12,6))

ax.set_xticks(np.arange(len(x)))
ax.set_xticklabels(x, rotation = 45)
ax.bar(np.arange(len(x)),cg.loc[:,'A'].values, align='center', label='Carrier')
ax.bar(np.arange(len(x)),cg.loc[:,'B'].values, align='center', bottom=b_bottom, label='Weather')
ax.bar(np.arange(len(x)),cg.loc[:,'C'].values, align='center', bottom=c_bottom, label='NAS')
ax.bar(np.arange(len(x)),cancelled_group.loc[:,'D'].values, align='center', bottom=d_bottom, label='S
ecurity')

ax.set_xlabel('Aircraft Carrier Code')
ax.set_ylabel('Number of Cancellations')

plt.legend()
plt.show()

total_flights_per_carrier = df['UniqueCarrier'].groupby(df['UniqueCarrier']).count()

fig, ax1 = plt.subplots(figsize = (12,6))

x = total_flights_per_carrier.index.values

ax1.set_xticks(np.arange(len(x)))
ax1.set_xticklabels(x, rotation = 45)
ax1.bar(np.arange(len(x)),total_flights_per_carrier.values, align='center')

ax1.set_xlabel('Aircraft Carrier Code')
ax1.set_ylabel('Total Number of Flights')

plt.show()
```


Carrier
Weather
NAS
Security
Number of Cancellations
20000
17500
15000
12500
10000
7500
5000
2500
0
9E
AA
AQ
AS
B6
CO
DL
EV
F9
FL
HA
MQ
NW
OH
OO
UA
US
WN
XE
YV
Aircraft Carrier Code

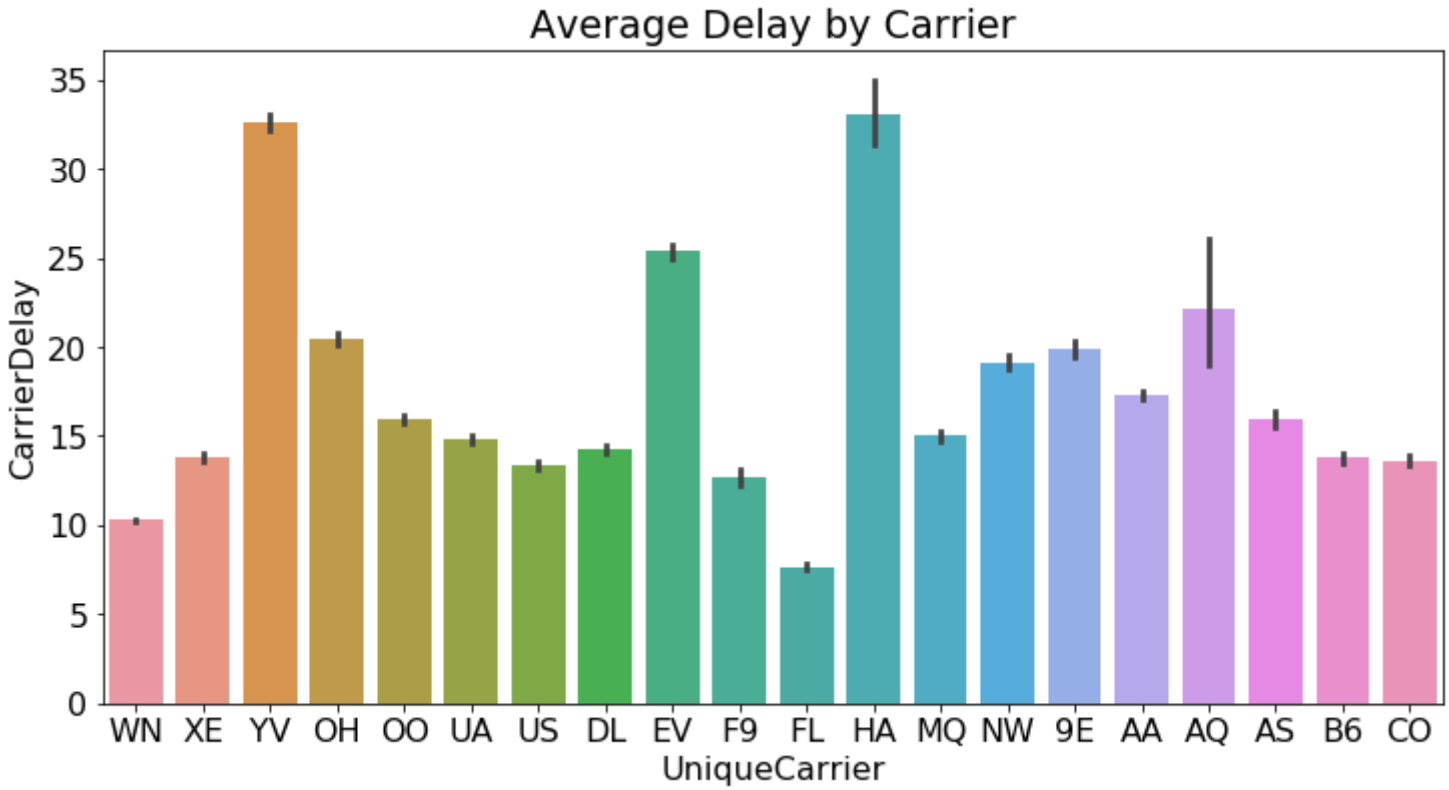

Total Number of Flights
1200000
1000000
800000
600000
400000
200000
0
9E
AA
AQ
AS
B6
CO
DL
EV
F9
FL
HA
MQ
NW
OH
OO
UA
US
WN
XE
YV
Aircraft Carrier Code

In [92]:
```
df2 = df.filter(['Month','CarrierDelay','WeatherDelay','NASDelay','SecurityDelay','LateAircraftDelay'
], axis=1)
df2 = df2.groupby('Month')['LateAircraftDelay','CarrierDelay','WeatherDelay','NASDelay','SecurityDela
y'].sum().plot()
df2.legend(loc='upper center', bbox_to_anchor=(0.5, 1.25), ncol=3, fancybox=True, shadow=True)
plt.show()
```



In [111]:
```
fig, ax = plt.subplots(figsize = (12,6))
sns.barplot('UniqueCarrier','CarrierDelay', data=df,ax=ax)
ax.set_title('Average Delay by Carrier')
```

Out[111]:
```
Text(0.5, 1.0, 'Average Delay by Carrier')
```



In [ ]: