Assignment 5

Big data and Text Mining

Wendy Gao

In [124]:
```
import os
import shutil
import sh
import csv
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pyspark.sql.functions import udf
%matplotlib inline
```

In [12]:
```
hdfsdir = "hdfs:///user/wendygao16/hw5/"
path = "hdfs:///user/kadochnikov/Airlines/"
filename = "200*.csv"
csv_hdfs_file = path+filename
csv_hdfs_file_copy = hdfsdir+filename
```

In [13]:
```
try:
    sh.hdfs('dfs','-cp',csv_hdfs_file, hdfsdir)
except:
    print('*** Exists ***')
```

```
*** Exists ***
```

In [14]:
```
print(sh.hdfs('dfs', '-ls', hdfsdir))
```

```
Found 2 items
-rw-r--r--   3 wendygao16 wendygao16  702878193 2017-05-25 16:30 hdfs:///user
/wendygao16/hw5/2007.csv
-rw-r--r--   3 wendygao16 wendygao16  689413344 2017-05-25 16:30 hdfs:///user
/wendygao16/hw5/2008.csv
```

In [15]:
```
flights_df = sqlContext.read.format('com.databricks.spark.csv').\
options(header='true', inferschema='true', delimiter=',', quote='"').load(csv
_hdfs_file).cache()
```

In [17]:
```
flights_df.printSchema()
```

```
root
 |-- Year: integer (nullable = true)
 |-- Month: integer (nullable = true)
 |-- DayofMonth: integer (nullable = true)
 |-- DayOfWeek: integer (nullable = true)
 |-- DepTime: string (nullable = true)
 |-- CRSDepTime: integer (nullable = true)
 |-- ArrTime: string (nullable = true)
 |-- CRSArrTime: integer (nullable = true)
 |-- UniqueCarrier: string (nullable = true)
 |-- FlightNum: integer (nullable = true)
 |-- TailNum: string (nullable = true)
 |-- ActualElapsedTime: string (nullable = true)
 |-- CRSElapsedTime: string (nullable = true)
 |-- AirTime: string (nullable = true)
 |-- ArrDelay: string (nullable = true)
 |-- DepDelay: string (nullable = true)
 |-- Origin: string (nullable = true)
 |-- Dest: string (nullable = true)
 |-- Distance: integer (nullable = true)
 |-- TaxiIn: string (nullable = true)
 |-- TaxiOut: string (nullable = true)
 |-- Cancelled: integer (nullable = true)
 |-- CancellationCode: string (nullable = true)
 |-- Diverted: integer (nullable = true)
 |-- CarrierDelay: string (nullable = true)
 |-- WeatherDelay: string (nullable = true)
 |-- NASDelay: string (nullable = true)
 |-- SecurityDelay: string (nullable = true)
 |-- LateAircraftDelay: string (nullable = true)
```

In [18]:
```
flights_df.select('Year', 'Month', 'DepTime', 'ArrTime', 'FlightNum', 'Origin
', 'Dest', 'Distance').show(5)
```

```
+----+-----+-------+-------+---------+------+----+--------+
|Year|Month|DepTime|ArrTime|FlightNum|Origin|Dest|Distance|
+----+-----+-------+-------+---------+------+----+--------+
|2007|    1|   1232|   1341|     2891|   SMF| ONT|     389|
|2007|    1|   1918|   2043|      462|   SMF| PDX|     479|
|2007|    1|   2206|   2334|     1229|   SMF| PDX|     479|
|2007|    1|   1230|   1356|     1355|   SMF| PDX|     479|
|2007|    1|    831|    957|     2278|   SMF| PDX|     479|
+----+-----+-------+-------+---------+------+----+--------+
only showing top 5 rows
```

In [19]:
```
delays = flights_df.select('Year', 'Month','DayofMonth','DayOfWeek', 'Origin'
, 'Dest', 'Distance','DepDelay','ArrDelay')
delays.printSchema()
```

```
root
 |-- Year: integer (nullable = true)
 |-- Month: integer (nullable = true)
 |-- DayofMonth: integer (nullable = true)
 |-- DayOfWeek: integer (nullable = true)
 |-- Origin: string (nullable = true)
 |-- Dest: string (nullable = true)
 |-- Distance: integer (nullable = true)
 |-- DepDelay: string (nullable = true)
 |-- ArrDelay: string (nullable = true)
```

In [20]:
```
delays.createOrReplaceTempView("delays")
sqlContext.tables().filter('tablename Like "delays"').show()
```

```
+---------+-----------+
|tableName|isTemporary|
+---------+-----------+
|   delays|       true|
+---------+-----------+
```

In [21]:
```
delays = delays.withColumn('DepDelay',delays.DepDelay.cast('int').cast('int')
)\
.withColumn('ArrDelay', delays.ArrDelay.cast('int').cast('int'))
```

In [15]:
```
delays.show(10)
delays.cache()
```

```
+----+-----+----------+---------+------+----+--------+--------+--------+
|Year|Month|DayofMonth|DayOfWeek|Origin|Dest|Distance|DepDelay|ArrDelay|
+----+-----+----------+---------+------+----+--------+--------+--------+
|2007|    1|         1|        1|   SMF| ONT|     389|       7|       1|
|2007|    1|         1|        1|   SMF| PDX|     479|      13|       8|
|2007|    1|         1|        1|   SMF| PDX|     479|      36|      34|
|2007|    1|         1|        1|   SMF| PDX|     479|      30|      26|
|2007|    1|         1|        1|   SMF| PDX|     479|       1|      -3|
|2007|    1|         1|        1|   SMF| PDX|     479|      10|       3|
|2007|    1|         1|        1|   SMF| PHX|     647|      56|      47|
|2007|    1|         1|        1|   SMF| PHX|     647|       9|      -2|
|2007|    1|         1|        1|   SMF| PHX|     647|      47|      44|
|2007|    1|         1|        1|   SMF| PHX|     647|       3|      -7|
+----+-----+----------+---------+------+----+--------+--------+--------+
only showing top 10 rows
```

Out[15]:
```
DataFrame[Year: int, Month: int, DayofMonth: int, DayOfWeek: int, Origin: str
ing, Dest: string, Distance: int, DepDelay: int, ArrDelay: int]
```

In [22]:
```
flights_df.createOrReplaceTempView("delays")
sqlContext.tables().filter("tableName LIKE '%delays%'").show()
```

```
+---------+-----------+
|tableName|isTemporary|
+---------+-----------+
|   delays|       true|
+---------+-----------+
```

In [126]:
```
flight_delays = sqlContext.sql("\
           select CONCAT(Origin, ' ', Dest) AS pairs, Year, Month, DepDelay, 
ArrDelay \
           from delays").cache()
```

In [127]:
```
flight_delays.createOrReplaceTempView("flight_delays")
```

In [128]:
```
flight_delays.show(10)
```

```
+-------+----+-----+--------+--------+
|  pairs|Year|Month|DepDelay|ArrDelay|
+-------+----+-----+--------+--------+
|SMF ONT|2007|    1|       7|       1|
|SMF PDX|2007|    1|      13|       8|
|SMF PDX|2007|    1|      36|      34|
|SMF PDX|2007|    1|      30|      26|
|SMF PDX|2007|    1|       1|      -3|
|SMF PDX|2007|    1|      10|       3|
|SMF PHX|2007|    1|      56|      47|
|SMF PHX|2007|    1|       9|      -2|
|SMF PHX|2007|    1|      47|      44|
|SMF PHX|2007|    1|       3|      -7|
+-------+----+-----+--------+--------+
only showing top 10 rows
```

Determine which locations had the worst delays for both arrivals (ArrDelay) and departures (DepDelay).

In order to decide which locations have the worst delays, we could calculate the sum and the average of the delayed time for arrivals and departures. And the locations will be represented by the group of origin and destination.

Arrivals Delay

In [93]:
```
sum_arr_delay = sqlCtx.sql("\
    select pairs, sum(ArrDelay) as sum_arr_delay\
    from flight_delays\
    where ArrDelay > 0\
    group by pairs\
    order by sum_arr_delay desc\
    limit 10").show()
```

```
+-------+-------------+
|  pairs|sum_arr_delay|
+-------+-------------+
|ORD LGA|     568919.0|
|LGA ORD|     542897.0|
|ORD EWR|     494749.0|
|ATL LGA|     441876.0|
|LAX SFO|     441369.0|
|EWR ORD|     440388.0|
|ATL EWR|     413431.0|
|LGA ATL|     395765.0|
|SFO LAX|     390683.0|
|MSP ORD|     383868.0|
+-------+-------------+
```

In [138]:
```
# locations with the average of arrival delays time
avg_arr_delay = sqlCtx.sql("\
    select pairs, avg(ArrDelay) as avg_arr_delay\
    from flight_delays\
    where ArrDelay > 0\
    group by pairs\
    order by avg_arr_delay desc")
avg_arr_delay.createOrReplaceTempView("avg_arr_delay")
avg_arr_delay.show(10)
```

```
+-------+-------------+
|  pairs|avg_arr_delay|
+-------+-------------+
|CMI SPI|        575.0|
|ONT IAD|        370.0|
|ELP MFE|        316.0|
|ACY MYR|        252.0|
|BHM JFK|        252.0|
|SLC KOA|        246.0|
|RIC ORF|        227.0|
|JAX CMH|        217.0|
|ATW DSM|        210.0|
|AVP BUF|        195.0|
+-------+-------------+
only showing top 10 rows
```

Departures Delay

In [139]:

```
# locations the sum of departure delays time
sum_dep_delay =sqlCtx.sql("\
    select pairs, sum(DepDelay) as sum_depdelay \
    from flight_delays \
    where DepDelay > 0 \
    group by pairs \
    order by sum_depdelay desc")
sum_dep_delay.createOrReplaceTempView("sum_dep_delay")
sum_dep_delay.show(10)
```

```
+-------+------------+
|  pairs|sum_depdelay|
+-------+------------+
|ORD LGA|    490557.0|
|ORD EWR|    470926.0|
|LGA ORD|    431418.0|
|LAX SFO|    406650.0|
|ATL EWR|    382377.0|
|EWR ORD|    368333.0|
|SFO LAX|    365042.0|
|ATL LGA|    363961.0|
|DFW ORD|    348280.0|
|ORD MSP|    339275.0|
+-------+------------+
only showing top 10 rows
```

In [140]:

```
# locations with the average of departure delays time
avg_dep_delay =sqlCtx.sql("\
    select pairs, avg(DepDelay) as avg_depdelay \
    from flight_delays \
    where DepDelay > 0 \
    group by pairs \
    order by avg_depdelay desc")
avg_dep_delay.createOrReplaceTempView("avg_dep_delay")
avg_dep_delay.show(10)
```

```
+-------+-----------+
|  pairs|avg_depdelay|
+-------+-----------+
|CMI SPI|      587.0|
|ONT IAD|      386.0|
|ABQ GJT|      366.0|
|SDF SPI|      329.0|
|SLC KOA|      317.5|
|ELP MFE|      307.0|
|HPN PIA|      298.0|
|OKC GJT|      270.0|
|TUL PIA|      243.0|
|ACY MYR|      222.0|
+-------+-----------+
only showing top 10 rows
```

Determine which locations had fewest delays.

In [102]:
```
# count total number of flights for each unique routepairs
total_flights = sqlCtx.sql("\
    select pairs, count(pairs) as total_num \
    from flight_delays \
    group by pairs \
    order by total_num desc\
    limit 10").cache()
```

In [103]:
```
total_flights.show()
```

```
+-------+---------+
|  pairs|total_num|
+-------+---------+
|OGG HNL|    28482|
|HNL OGG|    27890|
|LAX LAS|    26158|
|SFO LAX|    25747|
|LAS LAX|    25544|
|LAX SFO|    25182|
|BOS LGA|    24292|
|LGA BOS|    24257|
|LAX SAN|    24024|
|SAN LAX|    24003|
+-------+---------+
```

In [104]:
```
total_flights.createOrReplaceTempView("total_flights")
```

Departure

In [116]:
```
dep_delay = sqlCtx.sql("\
    select pairs, count(pairs) as dep_delay\
    from flight_delays \
    where DepDelay > 0 \
    group by pairs \
    order by dep_delay").cache()
dep_delay.createOrReplaceTempView("dep_delay")
```

In [156]:
```
dep_delay_pct = sqlCtx.sql("\
    select d.pairs as pairs, \
    dep_delay/total_num as dep_delay_pct \
    from total_flights t \
    INNER JOIN \
    dep_delay d ON t.pairs = d.pairs \
    order by dep_delay_pct desc").cache()
dep_delay_pct.createOrReplaceTempView("dep_delay_pct")
dep_delay_pct.show(10)
```

```
+-------+-------------------+
|  pairs|      dep_delay_pct|
+-------+-------------------+
|LAX SFO| 0.4492494639027877|
|LAS LAX| 0.4453492013780144|
|LAX LAS|0.41662206590717943|
|SFO LAX|0.38097642443779856|
|LAX SAN|0.27505827505827507|
|SAN LAX| 0.2498021080698246|
|BOS LGA|0.23719743125308743|
|LGA BOS| 0.2041472564620522|
|OGG HNL|0.18660908644055896|
|HNL OGG|0.18633918967371818|
+-------+-------------------+
```

Arrival

In [118]:
```
arr_delay = sqlCtx.sql("\
    select pairs, count(pairs) as arr_delay \
    from flight_delays \
    where ArrDelay > 0 \
    group by pairs \
    order by arr_delay").cache()
```

In [119]:
```
arr_delay.createOrReplaceTempView("arr_delay")
```

In [155]:
```
arr_delay_pct = sqlCtx.sql("\
    select d.pairs as pairs, \
    arr_delay/total_num as arr_delay_pct \
    from total_flights t \
    INNER JOIN \
    arr_delay d ON t.pairs = d.pairs \
    order by arr_delay_pct desc").cache()
arr_delay_pct.createOrReplaceTempView("arr_delay_pct")
arr_delay_pct.show(10)
```

```
+-------+------------------+
|  pairs|     arr_delay_pct|
+-------+------------------+
|LAS LAX|0.4522001252740369|
|SFO LAX| 0.447896842350565|
|LAX SFO|  0.44420617901675|
|LAX LAS| 0.437801055126538|
|BOS LGA|0.4159805697348921|
|LAX SAN| 0.378246753246753|
|LGA BOS| 0.360390815022467|
|SAN LAX| 0.338749323001291|
|OGG HNL|0.2974159118039463|
|HNL OGG| 0.287020437432771|
+-------+------------------+
```

Data visualization

In [162]:
```
arr_delay_ts = flight_delays.filter('ArrDelay>0').crosstab('year', 'month')
arr_delay_ts.show(10)
```

```
+----------+------+------+------+------+------+------+------+------+------+--
----+------+------+
|year_month|     1|    10|    11|    12|     2|     3|     4|     5|     6|
7|     8|     9|
+----------+------+------+------+------+------+------+------+------+------+--
----+------+------+
|      2008|279427|183582|181506|280493|278902|294556|256142|254673|295897|26
4630|239737|169959|
|      2007|286334|270098|242722|332449|284152|293360|273055|275332|326446|32
6559|317197|225751|
+----------+------+------+------+------+------+------+------+------+------+--
----+------+------+
```

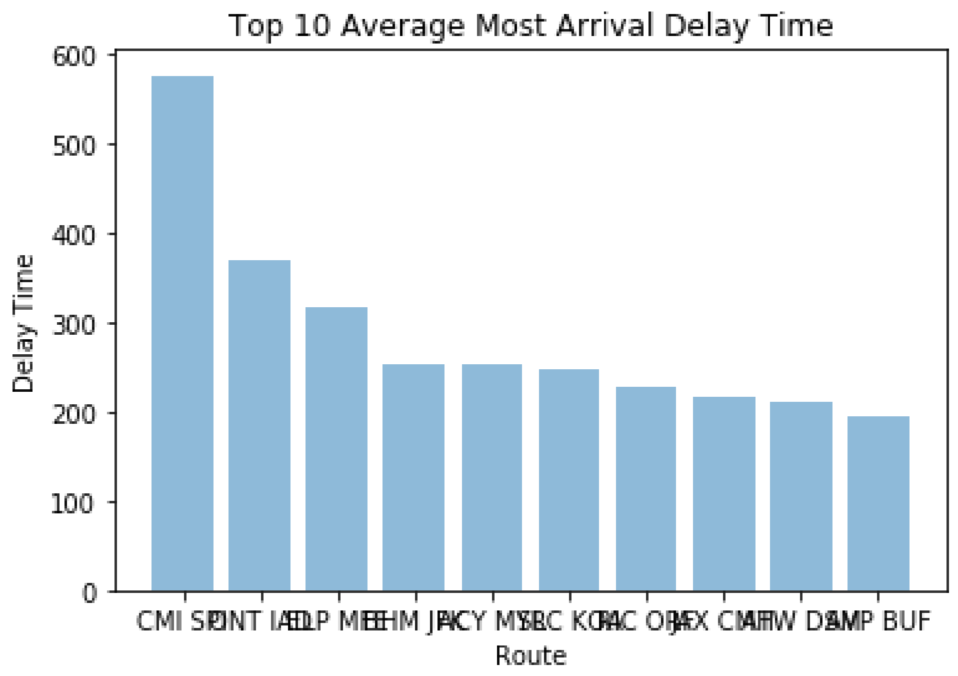
In [163]:
```
dep_delay_ts = flight_delays.filter('DepDelay>0').crosstab('year', 'month')
dep_delay_ts.show(10)
```

```
+----------+------+------+------+------+------+------+------+------+------+--
----+------+------+
|year_month|     1|    10|    11|    12|     2|     3|     4|     5|     6|
7|     8|     9|
+----------+------+------+------+------+------+------+------+------+------+--
----+------+------+
|      2008|247948|162531|157278|263949|252765|271969|220864|220614|271014|25
3632|231349|147061|
|      2007|255777|231129|217557|304011|259288|276261|249097|241699|307986|30
7864|298530|195615|
+----------+------+------+------+------+------+------+------+------+------+--
----+------+------+
```

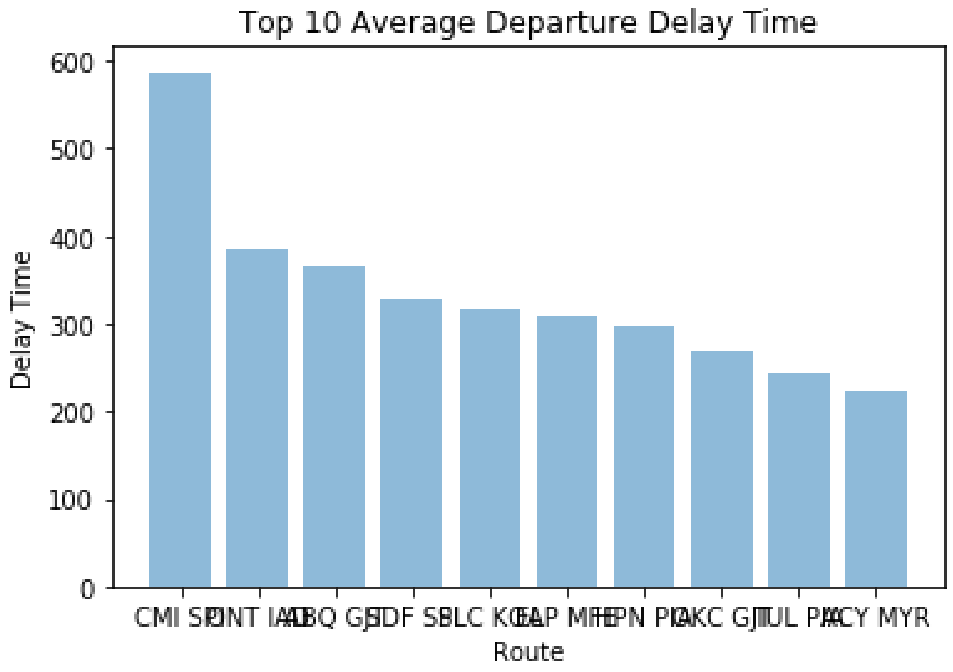
In [172]:
```
# Most frequent departure and arrival delay
most_arr_avg = avg_arr_delay.toPandas()
most_dep_avg = avg_dep_delay.toPandas()

# Least frequent departure and arrival delay
least_arr_pct = arr_delay_pct.toPandas()
least_dep_pct = dep_delay_pct.toPandas()
```

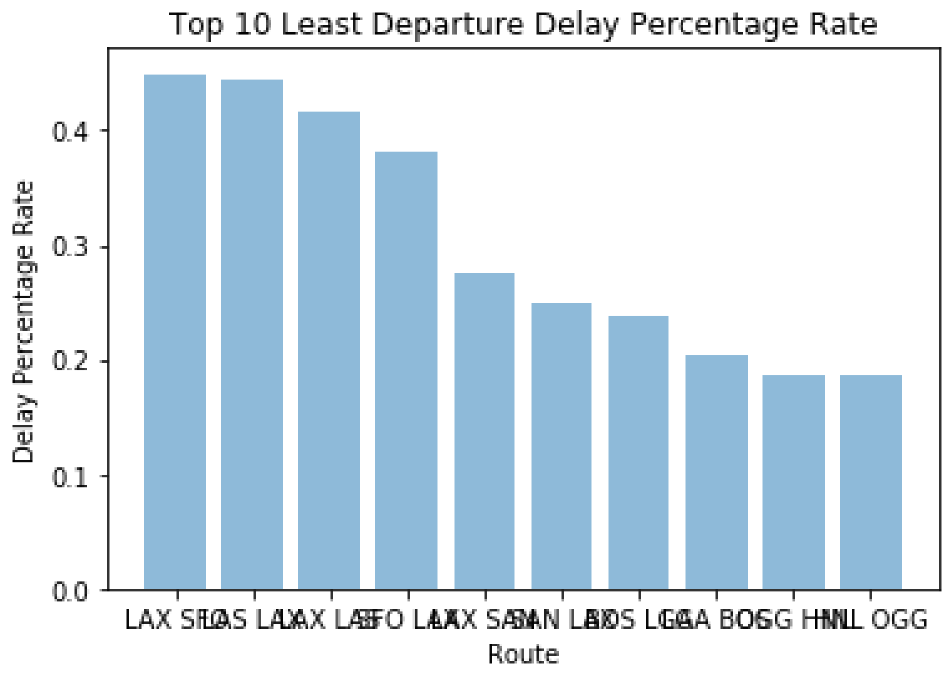
In [148]:
```
# plot top 10 with most average arrival delay time
delays = most_arr_avg[0:10]['avg_arr_delay']
objects = most_arr_avg[0:10]['pairs']
plt.bar(range(10) ,delays, align='center', alpha=0.5)
plt.xticks(range(10), objects)
plt.ylabel('Delay Time')
plt.xlabel('Route')
plt.title('Top 10 Average Most Arrival Delay Time')
plt.show()
```



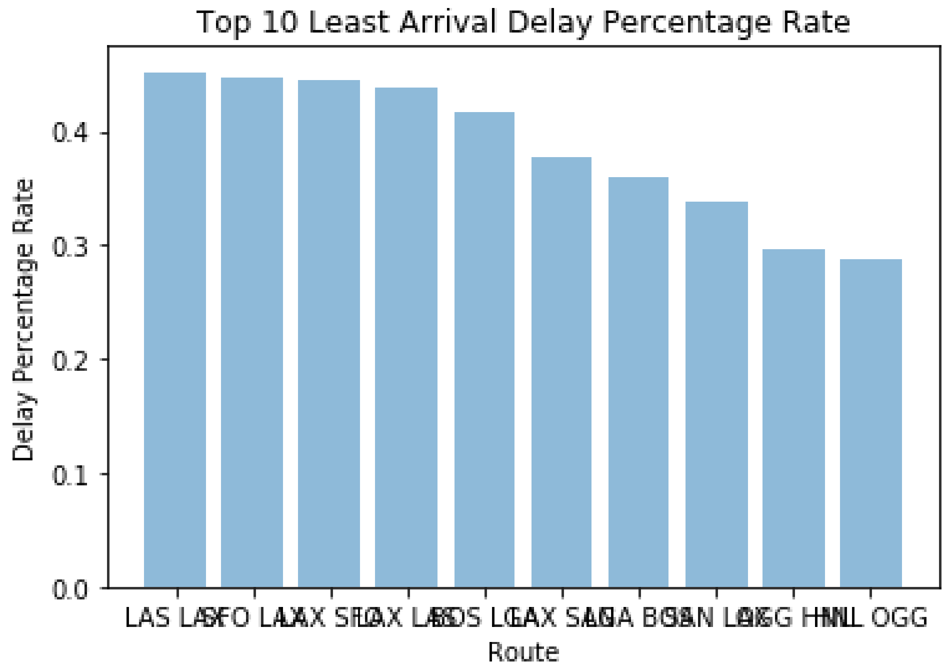
In [147]:
```
# plot the top 10 average departure delay time
delays = most_dep_avg[0:10]['avg_depdelay']
objects = most_dep_avg[0:10]['pairs']
plt.bar(range(10) ,delays, align='center', alpha=0.5)
plt.xticks(range(10), objects)
plt.ylabel('Delay Time')
plt.xlabel('Route')
plt.title('Top 10 Average Departure Delay Time')
plt.show()
```



In [173]:
```
# plot top 10 with least departure delay percentage
delays = least_dep_pct[0:10]['dep_delay_pct']
objects = least_dep_pct[0:10]['pairs']
plt.bar(range(10) ,delays, align='center', alpha=0.5)
plt.xticks(range(10), objects)
plt.ylabel('Delay Percentage Rate')
plt.xlabel('Route')
plt.title('Top 10 Least Departure Delay Percentage Rate')
plt.show()
```



In [179]:
```python
# plot top 10 with least arrival delay percentage
delays = least_arr_pct[0:10]['arr_delay_pct']
objects = least_arr_pct[0:10]['pairs']
plt.bar(range(10) ,delays, align='center', alpha=0.5)
plt.xticks(range(10), objects)
plt.ylabel('Delay Percentage Rate')
plt.xlabel('Route')
plt.title('Top 10 Least Arrival Delay Percentage Rate')
plt.show()
```



From the above plots we could see that for the worst average delays time in departure, routepairs include: CMI-SPI, ONT-IAD, ABQ-GJT, SDF-SPI,SLC-KOA. And for the worst average delays time in arrival, routepairs include: CMI-SPI, ONT-IAD, ELP-MFE, BHM-JFK, ACY-MYR. So it could be noticed that routepairs such as CMI-SPI and ONT-IAD are the top two in both lists, which indicates that these two routes have the worst delay in both departure and arrival.

Seasonality analysis

In [164]:
```
dep_delay_season = dep_delay_ts.toPandas()
arr_delay_season = arr_delay_ts.toPandas()
```

In [166]:
```
# Rearrange the order
cols = dep_delay_season.columns.tolist()
print(cols)
cols_reorder = cols[1:2] + cols[5:] + cols[2:5]
print(cols_reorder)

dep_delay_season = dep_delay_season[cols_reorder]
dep_delay_season

arr_delay_season = arr_delay_season[cols_reorder]
arr_delay_season
```

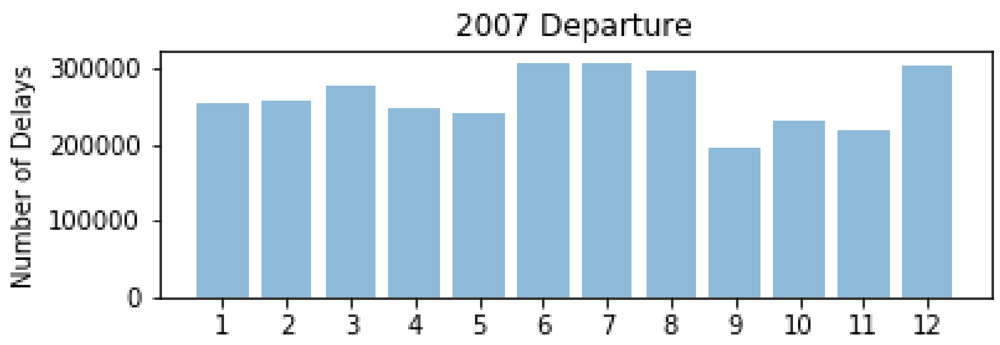
```
['year_month', '1', '10', '11', '12', '2', '3', '4', '5', '6', '7', '8', '9']
['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
```

Out[166]:

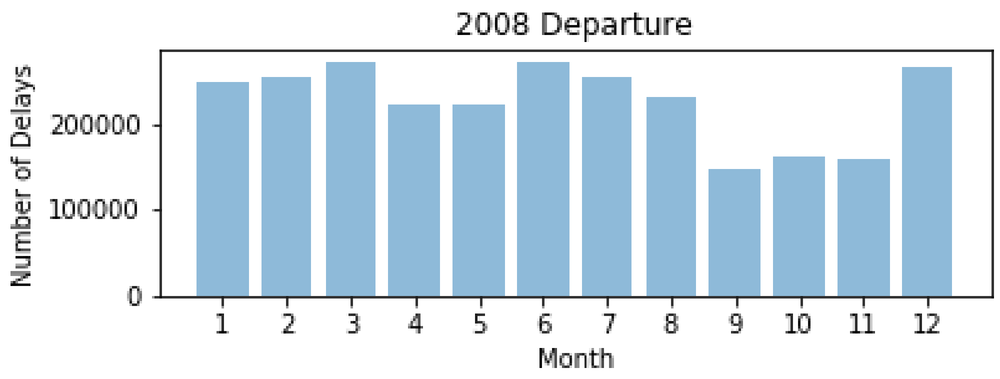
	1	2	3	4	5	6	7	8	9	10	11	12
0	279427	278902	294556	256142	254673	295897	264630	239737	169959	183582	181506	280493
1	286334	284152	293360	273055	275332	326446	326559	317197	225751	270098	242722	332449

In [167]:
```python
# Plot time series of departure delay
# 2007 Time Series Plot
dep_ts_delays_2007 = dep_delay_season.transpose()[1]
months = cols_reorder
plt.figure(1)
plt.subplot(211)
plt.bar(range(12), dep_ts_delays_2007, align = 'center', alpha = 0.5)
plt.xticks(range(12), months)
plt.ylabel('Number of Delays')
plt.title('2007 Departure')
```

Out[167]:
```
<matplotlib.text.Text at 0x6141ed0>
```

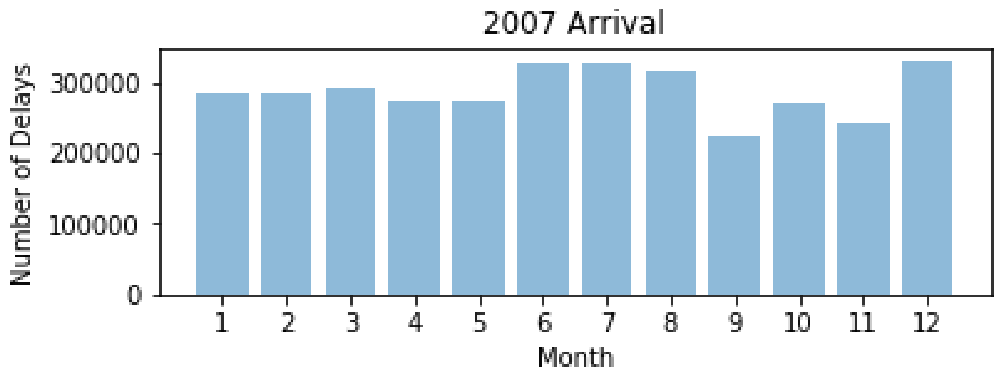


In [168]:
```python
# 2008 Time Series Plot
dep_ts_delays_2008 = dep_delay_season.transpose()[0]
plt.subplot(212)
plt.bar(range(12), dep_ts_delays_2008, align = 'center', alpha = 0.5)
plt.xticks(range(12), months)
plt.ylabel('Number of Delays')
plt.xlabel('Month')
plt.title('2008 Departure')
plt.show()
```

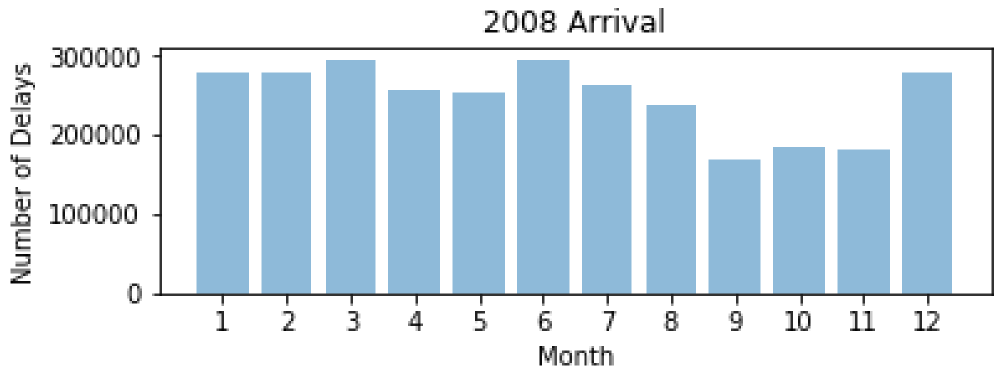


In [180]:
```
# Plot time series of arrival delay
# 2007 Time Series Plot
arr_ts_delays_2007 = arr_delay_season.transpose()[1]
months = cols_reorder
plt.figure(1)
plt.subplot(211)
plt.bar(range(12), arr_ts_delays_2007, align = 'center', alpha = 0.5)
plt.xticks(range(12), months)
plt.ylabel('Number of Delays')
plt.xlabel('Month')
plt.title('2007 Arrival')
```

Out[180]:
```
<matplotlib.text.Text at 0x653f6d0>
```



In [170]:
```python
# 2008 Time Series Plot
arr_ts_delays_2008 = arr_delay_season.transpose()[0]
plt.subplot(212)
plt.bar(range(12), arr_ts_delays_2008, align = 'center', alpha = 0.5)
plt.xticks(range(12), months)
plt.ylabel('Number of Delays')
plt.xlabel('Month')
plt.title('2008 Arrival')
plt.show()
```



From the above four plots we could see that seasonality exists and the patterns are very similiar. The high frequency departure delay happens in June, July and August as well as December, and the low frequency departure delays are in September, October and November. And the similar trend could be found for arrival delay.