我是在 colab 上使用 spark，因此實驗前配置的 code 如下所示：

【Code】

```
!apt-get -y install openjdk-8-jre-headless
!pip install pyspark
from pyspark.sql import SparkSession
from pyspark import SparkContext
spark = SparkSession.builder.master("local").getOrCreate()
sc = SparkContext.getOrCreate()

from google.colab import drive
import os
drive.mount("/content/gdrive")
Path = os.getcwd() + '/gdrive/MyDrive/Pyspark'
print(os.listdir(Path))
```

Q1: Find the maximal delays (you should consider both ArrDelay and DepDelay) for each month of 2007.

```
file = os.listdir(Path)
path = Path + '/' + file[6]
df = spark.read.csv(path, header = True)
TEMP = df.withColumn("Delay", df ["ArrDelay"].cast('float') + df
["DepDelay"].cast('float'))
TEMP.groupby("Month").agg({"Delay": "max"}).show()
```

【Code Output】

Month	max(Delay)
3	3111.0
1	2832.0
2	2699.0
5	2845.0
4	2817.0
7	2755.0
6	2711.0
8	2921.0
9	3354.0
10	5199.0
11	2283.0
12	3898.0

【Ans】

用 spark 將 ArrDelay 和 DepDelay 相加，找出各個月分的總和最大值。

在處理第二題問題前，我先將 2000 年資料至 2005 年資料做整合：

【Code】

```
import pyspark
from pyspark import SparkContext
from pyspark.sql import SQLContext as sqlc
import pandas as pd
import numpy as np
import gc
file = os.listdir(Path)[1:]
for idx in range(6):
  path = Path + '/' + file[idx]
  if idx == 0:
    DF = spark.read.csv(path, header = True)
    print("批次資料大小:{0}, {1}".format(DF.count(), len(DF.columns)))
  else:
    temp = spark.read.csv(path, header = True)
    DF = DF.union(temp)
    print("批次資料大小:{0}, {1}".format(temp.count(), len(temp.columns)))
print("總資料大小:{0},{1}".format(DF.count(), len(DF.columns)))
```

Q2: How many flights were delayed caused by security between 2000 ~ 2005? Please show the counting for each year.

【Code】

```
file = os.listdir(Path)[1:]
for year in range(2000, 2006):
  path = Path + '/' + file[year - 2000]
  df = spark.read.csv(path, header = True)
  TEMP = df.filter("SecurityDelay > 0")
  print("{0} 年因Security而delay {1} 個航班".format(year, TEMP.count()))

TEMP = DF.filter("SecurityDelay > 0")
print("全部因 Security 而 delay 共有 {0} 個航班".format(TEMP.count()))
```

【Code Output】

```
2000 年因Security而delay 0 個航班
2001 年因Security而delay 0 個航班
2002 年因Security而delay 0 個航班
2003 年因Security而delay 3740 個航班
2004 年因Security而delay 8158 個航班
2005 年因Security而delay 6627 個航班
全部因Security而delay共有 18525 個航班
```

【Ans】

由於如果航班有因 **security** 而 **delay** 的話，則 `SecurityDelay` 應大於零，因此我是以統計出 `SecurityDelay>0` 的資料數量作為統計 **flights were delayed**。實際結果如左圖所示，**2000** 年至 **2002** 年因安全性而延誤的航班為 **0**，而 **2003** 至 **2005** 年則分別為 **3740**、**8158**、**6627** 班航班被安全性問題延誤。在 **2000** 年至 **2005** 年總共有 **18525** 航班受到安全問題而延誤。

第三題資料區間是 2008 年，因此我只重新清空，只讀取 2008 年的資料。其中由於題目是要找出有最多 delays 的機場，因此我先將 delay 大於 0 的資料挑選出來，才再往下進行運算。

```
file = os.listdir(Path)[-1]
path = Path + '/' + file
df = spark.read.csv(path, header = True)
TYPE = df.select(['Origin', 'Dest', 'DepDelay', 'ArrDelay'])
TYPE = TYPE.filter("ArrDelay > 0")
TYPE = TYPE.filter("DepDelay > 0")
TYPE = TYPE.withColumn("DepDelay", TYPE["DepDelay"].cast('float'))
TYPE = TYPE.withColumn("ArrDelay", TYPE["ArrDelay"].cast('float'))
```

Q3: List Top 5 airports which occur delays most and least in 2008. (Please show the IATA airport code)

【Code】

```
typeI = TYPE.groupby('Origin').sum('DepDelay')
typeII = TYPE.groupby('Dest').sum('ArrDelay')
df1 = typeI.toPandas()
df2 = typeII.toPandas()
df1.index = df1['Origin']
df2.index = df2['Dest']
alist = list(df1['Origin'])
alist.remove('PUB')
result = pd.DataFrame(df1.loc[alist]['sum(DepDelay)'] +
df2.loc[alist]['sum(ArrDelay)'], columns = ['Delay'])
temp = pd.DataFrame([df1[df1['Origin'] == 'PUB']['sum(DepDelay)'][0]], index =
['PUB'], columns = ['Delay'])
result = pd.concat([result, temp], axis = 0)
IDXs = np.argsort(result['Delay'])
IDX_MIN = IDXs[:5];IDX_MAX = IDXs[-5:]
print("後五名:\n{0}\n***************\n前五名:\n{1}".format(result.iloc[IDX_MIN],
result.iloc[IDX_MAX]))
```

【Code Output】

```
後五名:
     Delay
TUP   17.0
PUB   54.0
PIR  150.0
BLI  431.0
BJI  473.0
***************
前五名:
          Delay
DEN   5272793.0
EWR   6042575.0
DFW   6803176.0
ATL  10700791.0
ORD  12989801.0
```

【Ans】

考慮到 DepDelay 應該是屬於在 Origin(起始點起飛)以及 ArrDelay 應是屬於在 Dest(終點降落)引此，我先在 Spark 上面分別隊 Origin & DepDelay、ArrDelay & Dest 進行 groupby 的加總運算，之後整併成一個 303rows、1 columns 的資料集，再輸出至 dataframe 上去做統整。
最終我比較在同一個機場下，sum(DepDelay) + sum(ArrDelay)的總和。
最終結果如左表所示。