Big Data Analytics Techniques and Applications

Homework 2

309652022 黃伯丞

A. 資料集描述

這次作業所使用的資料集是航班相關的資料集，包含 2000~2005、2007~2008 年的資料，資料是以年為單位的，每份介於 0.5~0.6GB 之間，總共 4GB。

B. 使用工具

我主要利用 PySpark 在 Google Colab 上進行資料處理，第 3 題用 Tableau 建表格。

C. 環境設定

這次的作業題目每題所需要的區間都是以年為單位的，所以不用像上次作業一樣把資料合併在一起，可以針對題目需求選用資料，也因為 PySpark 是分散式處理，所以在速度上相當快速。在 Colab 上使用 PySpark 要先安裝套件，接著連結雲端硬碟就能讀取雲端硬碟上的資料。利用 SparkSession 建立 Dataframe 的產生器。

```
!apt-get -y install openjdk-8-jre-headless
!pip install pyspark

from google.colab import drive
drive.mount('/content/drive')

# Import SparkSession
from pyspark.sql import SparkSession
from pyspark.sql.types import IntegerType

# Create SparkSession
spark=SparkSession.builder.getOrCreate()
```

D. 結果

I. Q1

第一題要找出 2007 年每個月的最大延遲時間，航班會有起點跟終點，造成延遲時間長是因為班機比預估時間晚、尚未起飛，假如開始飛行直到降落的過程很順遂的話，實際的 DepDelay 會比預估班機時間來得低，反之則變高，因此最大延遲時間不僅只出現在起飛時的紀錄。我先利用 Pyspark 讀入 2007 年的資料，把兩個欄位的資料型別從字串改為整數，再來利用迴圈設定月份的 filter 找出每個月的 ArrDelay, DepDelay 最大值。

```
df=spark.read.csv('drive/MyDrive/HW2/2007.csv',header=True,inferSchema=True)
df=df.withColumn('ArrDelay',df['ArrDelay'].cast(IntegerType()))
df=df.withColumn('DepDelay',df['DepDelay'].cast(IntegerType()))

for i in range(1,13):
  print(df.filter(df.Month==i).select(max('ArrDelay')).collect()[0],
        df.filter(df.Month==i).select(max('DepDelay')).collect()[0])
```

找出來之後我把數字整理在下方的表格，題目所求的是紅色數字的值。

2007	最大 ArrDelay	最大 DepDelay
1 月	1426	1406
2 月	1359	1340
3 月	1564	1547
4 月	1402	1415
5 月	1429	1416
6 月	1351	1360
7 月	1386	1369
8 月	1472	1449
9 月	1665	1689
10 月	2598	2601
11 月	1146	1137
12 月	1942	1956

II. Q2

第二題要找出在 2000~2005 年間有哪些班機的延遲原因包括安全因素。在讀入資料後，藉由 df_list[i].select('SecurityDelay').distinct().count() 函式我發現 2000~2002 年皆沒有最後 5 欄 (CarrierDelay, WeatherDelay, NASDelay, SecurityDelay, LateAircraftDelay) 的資料，這些欄位被登記為 NA，所以我只統計 2003~2005 年的資料，計算有多少筆資料的 SecurityDelay 大於 0。執行完得到在 2003 年發生了 3740 起，在 2004 年發生了 8158 起，在 2005 年發生了 6627 起。

```
df_list=[]
for i in range(6):
  df_list.append(spark.read.csv('drive/MyDrive/HW2/200'+str(i)+'.csv',header=True,inferSchema=True))

for i in range(6):
  print('200'+str(i)+':',df_list[i].select('SecurityDelay').distinct().count())

for i in range(3,6):
  df=df_list[i]
  df=df.withColumn('SecurityDelay',df['SecurityDelay'].cast(IntegerType()))
  print('200'+str(i)+':',df.filter(df.SecurityDelay>0).count())
```

III. Q3

第三題需要列出發生延遲次數最多與最少次的前五名機場，航班會有起點跟終點，如果 ArrDelay (或 DepDelay) 數值大於 0 表示發生延遲，反之則是提早出發 (或

抵達)，所以要統計數值大於 0 時在不同機場的發生次數。從 2008 年的航班資料來看，利用 groupby, count 可知共有 305 座機場，其中有 303 座是起飛地、302 座是降落地，設定條件統計完發現每座機場都至少發生過一起延遲事件。

首先把資料輸入整理完用 filter 設定條件以及 groupby 聚類 count 計數後就能得到兩個新的 Dataframe，因為要算總和所以需要重新命名同名的欄位再用 join 合併這兩個 Dataframe，得到 Delay_df。因為有 3 座機場只出現在起飛地或降落地，所以要用 fillna 將 NaN 改為 0，最後算出總和。

```
df=spark.read.csv('drive/MyDrive/HW2/2008.csv',header=True,inferSchema=True)
df=df.withColumn('ArrDelay',df['ArrDelay'].cast(IntegerType()))
df=df.withColumn('DepDelay',df['DepDelay'].cast(IntegerType()))

ArrDelay_df=df.filter(df.ArrDelay>0).groupby('Origin').count()
DepDelay_df=df.filter(df.DepDelay>0).groupby('Dest').count()

ArrDelay_df=ArrDelay_df.withColumnRenamed('count','ArrCount')
DepDelay_df=DepDelay_df.withColumnRenamed('count','DepCount')

Delay_df=ArrDelay_df.join(DepDelay_df,ArrDelay_df.Origin==DepDelay_df.Dest,'outer')

Delay_df=Delay_df.fillna(0)

Delay_df=Delay_df.withColumn('total',Delay_df.ArrCount+Delay_df.DepCount)
```

我利用輸出的資料傳到 Tableau 製作以下表格，分別為發生最多次延遲以及最少次延遲的前十名。在最多延遲的排名中，可以發現前五名的起飛及降落延遲排名沒有變動，第六名以後則發生了一點變化。從最少延遲的排名可以發現前五名的次數都是個位數。

IATA	起飛延遲	降落延遲	Total	最大排序	IATA	起飛延遲	降落延遲	Total	最小排序
ATL	204,157	143,629	347,786	1	OGD	0	1	1	1
ORD	167,536	135,780	303,316	2	CYS	0	1	1	1
DFW	134,254	95,262	229,516	3	PUB	2	0	2	3
DEN	110,344	89,988	200,332	4	TUP	3	1	4	4
LAX	86,615	89,000	175,615	5	PIR	3	3	6	5
PHX	79,520	74,445	153,965	6	BJI	12	15	27	6
IAH	84,673	64,512	149,185	7	INL	11	18	29	7
LAS	68,647	76,174	144,821	8	BLI	13	23	36	8
EWR	72,364	68,308	140,672	9	HTS	21	38	59	9
SFO	62,449	62,554	125,003	10	SUX	31	32	63	10

E. 問題討論

這次的作業利用 PySpark 來處理 0.5GB 的資料，可以發現利用 Spark 建立的 Dataframe 在查詢方面的速度相當迅速，短時間內即能得到所需結果。上網尋找 PySpark 相關資料時發現可以利用 pandas 以及 SQL 的語法來處理它的 Dataframe，對於熟悉至少一種語法的人來說很容易上手。