M3_T02

January 17, 2023

1 Sprint 3

1.1 Tasca M3 T02

1.1.1 Exercici 1

Descarrega el data set Airlines Delay: Airline on-time statistics and delay causes i carrega'l a un Pandas Dataframe. Explora les dades que conté, explica breument quines variables hi ha i queda't únicament amb les columnes que consideris rellevants. Justifica la teva elecció.

->Airlines Delay: Airline on-time statistics and delay causes

[1]:
```
import numpy as np
import statistics as st
import pandas as pd
!pip install tabulate
from tabulate import tabulate
from IPython.display import display

df1=pd.read_csv('DelayedFlights.csv')
df1.head()
```

```
Requirement already satisfied: tabulate in
/Users/franciscodelcampo/opt/anaconda3/lib/python3.8/site-packages (0.9.0)
```

[1]:

	Unnamed: 0	Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	\
0	0	2008	1	3	4	2003.0	1955	
1	1	2008	1	3	4	754.0	735	
2	2	2008	1	3	4	628.0	620	
3	4	2008	1	3	4	1829.0	1755	
4	5	2008	1	3	4	1940.0	1915	

	ArrTime	CRSArrTime	UniqueCarrier	…	TaxiIn	TaxiOut	Cancelled	\
0	2211.0	2225	WN	…	4.0	8.0	0	
1	1002.0	1000	WN	…	5.0	10.0	0	
2	804.0	750	WN	…	3.0	17.0	0	
3	1959.0	1925	WN	…	3.0	10.0	0	
4	2121.0	2110	WN	…	4.0	10.0	0	

```
   CancellationCode  Diverted  CarrierDelay  WeatherDelay NASDelay  \
0                 N         0           NaN           NaN      NaN
1                 N         0           NaN           NaN      NaN
2                 N         0           NaN           NaN      NaN
3                 N         0           2.0           0.0      0.0
4                 N         0           NaN           NaN      NaN

  SecurityDelay  LateAircraftDelay
0           NaN                NaN
1           NaN                NaN
2           NaN                NaN
3           0.0               32.0
4           NaN                NaN

[5 rows x 30 columns]
```

[2]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1936758 entries, 0 to 1936757
Data columns (total 30 columns):
 #   Column             Dtype
---  ------             -----
 0   Unnamed: 0         int64
 1   Year               int64
 2   Month              int64
 3   DayofMonth         int64
 4   DayOfWeek          int64
 5   DepTime            float64
 6   CRSDepTime         int64
 7   ArrTime            float64
 8   CRSArrTime         int64
 9   UniqueCarrier      object
 10  FlightNum          int64
 11  TailNum            object
 12  ActualElapsedTime  float64
 13  CRSElapsedTime     float64
 14  AirTime            float64
 15  ArrDelay           float64
 16  DepDelay           float64
 17  Origin             object
 18  Dest               object
 19  Distance           int64
 20  TaxiIn             float64
 21  TaxiOut            float64
 22  Cancelled          int64
 23  CancellationCode   object
```

```
 24  Diverted            int64
 25  CarrierDelay        float64
 26  WeatherDelay        float64
 27  NASDelay            float64
 28  SecurityDelay       float64
 29  LateAircraftDelay   float64
dtypes: float64(14), int64(11), object(5)
memory usage: 443.3+ MB
```

En el dataset hi ha les columnes de l'any, mes, dia del mes i dia de la setmana que es podria agrupar en una unica columna que representi la data (YYYY-MM-DD).

Tenim el CRSDepTime i el DepTime que ens diuen el temps previst de sortida i el temps real així que podem quedar-nos amb els temps reals i afegir les columnes de ArrDelay i DepDelay que representen la diferencia entre el temps previst i el real.

Els temps en els quals es dona el trasllat de pista a la *gate* tampoc el mantindrem ja que tenim els temps totals amb *ActualElapsedTime* i no crec que calgui aquest nivell de detall en el nostre cas.

Com ja tenim el *CancellationCode*, no cal saber exactament quants minuts han endarrerit per aquell motiu si tenim els temps totals en altres columnes (No ens caldràn les darreres 5 columnes *Delay*).

També, els vols cancel · lats caldrà eliminar-los del dataset ja que no s'han donat, així com tractar els *missing values* que hi trobem.

Opcional: Es podria convertir en km la distància que en principi està en milles.

[3]:
```
# Date column merge and getting the new Date column in a basic-filtered dataset

date=df1[['Year','Month','DayofMonth']]
date_column=pd.to_datetime(date.Year*10000+date.Month*100+date.
 ↪DayofMonth,format='%Y%m%d')
date_column = pd.DataFrame(date_column, columns = ['Date'])
data=date_column.
 ↪join(df1[['UniqueCarrier','FlightNum','TailNum','Origin','Dest','Distance',
       'DepTime','ArrTime','DepDelay','ArrDelay','ActualElapsedTime','AirTime',
      'Cancelled','CancellationCode']])

# Desired columns dataset with sub-datasets
No_ArrTime=data[data['ArrTime'].isnull() & data['ArrTime'].isna() &␣
 ↪data['ArrDelay'].isnull() & data['ArrDelay'].isna()]
No_ArrDept=data[data['ArrDelay'].isnull() | data['ArrDelay'].isna()]

No_TailNum=data[data['TailNum'].isna()]

flights_cancelled=data[data['Cancelled']==1]

data=data[data['Cancelled']==0]

TD = data[data.TailNum.notnull()]
```

```
ATD = TD[TD.ArrTime.notnull()]
dataC = ATD[ATD.ArrDelay.notnull() | ATD.ArrDelay.notna() | ATD.AirTime.
 ↪notnull()]
dataC=dataC.drop_duplicates()
```

[4]:
```
dataCR = dataC.sample(n=200000) # Select 200.000 random rows of your dataset
dataCR[['DepTime','ArrTime','DepDelay','ArrDelay','ActualElapsedTime','AirTime']]=(dataCR[['De
 ↪convert_dtypes(infer_objects=False,␣
 ↪convert_integer=True,convert_string=False, convert_boolean=False,␣
 ↪convert_floating=False)
dataCR.head()
```

[4]:

	Date	UniqueCarrier	FlightNum	TailNum	Origin	Dest	Distance	\
1013914	2008-06-23	MQ	3980	N800AE	SDF	ORD	286	
1347410	2008-08-19	DL	205	N641DL	LAS	ATL	1747	
848465	2008-05-22	AA	553	N428AA	RSW	DFW	1017	
1535041	2008-10-18	WN	2032	N309SW	BHM	MSY	321	
56620	2008-01-09	OO	6434	N763SK	LAX	SAT	1210	

	DepTime	ArrTime	DepDelay	ArrDelay	ActualElapsedTime	AirTime	\
1013914	1452	1516	17	21	84	53	
1347410	1200	1916	7	11	256	212	
848465	1755	2001	50	61	186	169	
1535041	1742	1842	32	22	60	49	
56620	655	1131	40	27	156	137	

	Cancelled	CancellationCode
1013914	0	N
1347410	0	N
848465	0	N
1535041	0	N
56620	0	N

1.1.2 Exercici 2

Fes un informe complet del dataset:

1. Resumeix estadísticament el dataset i les columnes d'interès. Fes una anàlisi estadístic del que consideris rellevant.
2. Troba quantes dades faltants hi ha per columna.
3. Crea columnes noves (velocitat mitjana del vol, si ha arribat tard o no…).
4. Fes una taula de les aerolínies amb més endarreriments acumulats.
5. Quins són els vols més llargs? I els més endarrerits? Busca les rutes més llargues i les que acumulen més retards.
6. Aporta allò que consideris rellevant.

Apartat 1 Resumeix estadísticament el dataset i les columnes d'interès. Fes una anàlisi estadístic del que consideris rellevant.

Al llarg dels diferents apartats, no només es perfecciona i s'afegeixen columnes d'interés al dataset, així com la velocitat mitjana o si hi ha un vol atrassat i una classificació pertinent, sinó que també s'analitzen i s'extreu informació interessant respecte al *delay* dels vols i les aerolinies.

[5]:
```
def resest(df):

    print('\033[1m'+"Resum estadístic"+'\033[0m')
    ii=0

    for (columnName, columnData) in df.iteritems():
        array=columnData.to_numpy(dtype=float)
        suma=[]
        res=[]
        avg=[]
        suma=np.sum(array)
        avg=suma/len(array)

        ngran=max(array)
        npetit=min(array)
        res=ngran-npetit

        if columnName == 'Distance':
            print('\n\033[1m'+columnName+'\033[0m')
            print("\nTotal:",suma,'[m]')
            print("Diferència del més gran amb el més petit:",res,'[m]')
            print("Mitjana:",avg.round(2),'[m]')
        else:
            print('\n\033[1m'+columnName+'\033[0m')
            print("\nTotal:",suma,'[min]')
            print("Diferència del més gran amb el més petit:",res,'[min]')
            print("Mitjana:",avg.round(2),'[min]')
        if ii>0:
            rnp=np.corrcoef(parray,array)
            print('Coeficient de correlació entre '+columnName+' i␣
↪'+pcolumnName,rnp[1,0].round(2))
        parray=array
        pcolumnName=columnName
        ii +=1
    return suma,res,avg,rnp

#Resum estadístic del dataset

column_names=['DepDelay','ActualElapsedTime','Distance','AirTime','ArrDelay']
resest(dataCR[column_names])
print('\n\033[1m'+'Correlació entre columnes del dataset','\033[0m')
display(dataCR[column_names].corr(method='pearson'))
print('\n\033[1m'+'Resum general dataset'+'\033[0m')
```

```
dataCR.describe().round(2)
```

Resum estadístic

DepDelay

```
Total: 8621356.0 [min]
Diferència del més gran amb el més petit: 1704.0 [min]
Mitjana: 43.11 [min]
```

ActualElapsedTime

```
Total: 26693564.0 [min]
Diferència del més gran amb el més petit: 708.0 [min]
Mitjana: 133.47 [min]
Coeficient de correlació entre ActualElapsedTime i DepDelay 0.02
```

Distance

```
Total: 153349750.0 [m]
Diferència del més gran amb el més petit: 4938.0 [m]
Mitjana: 766.75 [m]
Coeficient de correlació entre Distance i ActualElapsedTime 0.95
```

AirTime

```
Total: 21692961.0 [min]
Diferència del més gran amb el més petit: 652.0 [min]
Mitjana: 108.46 [min]
Coeficient de correlació entre AirTime i Distance 0.98
```

ArrDelay

```
Total: 8438096.0 [min]
Diferència del més gran amb el més petit: 1774.0 [min]
Mitjana: 42.19 [min]
Coeficient de correlació entre ArrDelay i AirTime -0.0
```

Correlació entre columnes del dataset

	DepDelay	ActualElapsedTime	Distance	AirTime	ArrDelay
DepDelay	1.000000	0.017133	-0.009642	-0.004386	0.954403
ActualElapsedTime	0.017133	1.000000	0.953504	0.977288	0.065753
Distance	-0.009642	0.953504	1.000000	0.980118	-0.030794
AirTime	-0.004386	0.977288	0.980118	1.000000	-0.001087
ArrDelay	0.954403	0.065753	-0.030794	-0.001087	1.000000

Resum general dataset

[5]:

	FlightNum	Distance	DepTime	ArrTime	DepDelay	ArrDelay
count	200000.00	200000.00	200000.00	200000.00	200000.00	200000.00
mean	2177.80	766.75	1518.08	1610.13	43.11	42.19
std	1940.24	574.94	451.70	548.78	53.71	57.16
min	1.00	24.00	1.00	1.00	6.00	-67.00
25%	608.00	338.00	1203.00	1316.00	12.00	9.00
50%	1541.00	607.00	1545.00	1715.00	24.00	24.00
75%	3411.00	999.00	1901.00	2031.00	53.00	56.00
max	9740.00	4962.00	2400.00	2400.00	1710.00	1707.00

	ActualElapsedTime	AirTime	Cancelled
count	200000.00	200000.00	200000.0
mean	133.47	108.46	0.0
std	72.12	68.69	0.0
min	16.00	0.00	0.0
25%	80.00	58.00	0.0
50%	116.00	90.00	0.0
75%	165.00	137.00	0.0
max	724.00	652.00	0.0

Aquí tenim un resum de les estadístiques més rellevants del dataset així com les correlacions entre les columnes del dataset on si el valor es proper a 1 o -1, exlcloent l'1 de la diagonal, veiem una relació directa entre els valors d'ambdós columnes.

Apartat 2 Troba quantes dades faltants hi ha per columna.

[6]:
```
def missingdata(data,nameStr):
    columns = data.columns
    total_missing = data.isna().sum()
    percentage_missing = data.isna().sum() * 100 / len(data)
    total_null = data.isnull().sum()
    percentage_null = data.isnull().sum() * 100 / len(data)
    table_percentage_missing = pd.DataFrame({'%missing': percentage_missing})
    table_total_missing = pd.DataFrame({'total_missing': total_missing})
    table_percentage_null = pd.DataFrame({'%null': percentage_null})
    table_total_null = pd.DataFrame({'total_null': total_null})
    M=table_total_missing.join(table_percentage_missing)
    N=table_total_null.join(table_percentage_null)
    col_names=['total_missing','%missing','total_null','%null']
    print('\n\033[1m'+nameStr+'\033[0m')
    print('\n')
    print(tabulate(M.join(N), headers=col_names,tablefmt="github"))


#Dades faltants per columna i percentatge abans de treure els valors faltants␣
 ↪de TailNum i ArrTime
```

```
missingdata(df1, 'InitialData')


#missingdata(data, 'PreviousData')


#Dades faltants per columna i percentatge després de treure els valors faltants␣
 ↪de TailNum i ArrTime

missingdata(dataC, 'FilteredData')
```

InitialData

	total_missing	%missing	total_null	%null
Unnamed: 0	0	0	0	0
Year	0	0	0	0
Month	0	0	0	0
DayofMonth	0	0	0	0
DayOfWeek	0	0	0	0
DepTime	0	0	0	0
CRSDepTime	0	0	0	0
ArrTime	7110	0.367108	7110	0.367108
CRSArrTime	0	0	0	0
UniqueCarrier	0	0	0	0
FlightNum	0	0	0	0
TailNum	5	0.000258163	5	0.000258163
ActualElapsedTime	8387	0.433043	8387	0.433043
CRSElapsedTime	198	0.0102233	198	0.0102233

AirTime	8387	0.433043	8387	0.433043	
ArrDelay	8387	0.433043	8387	0.433043	
DepDelay	0	0	0	0	
Origin	0	0	0	0	
Dest	0	0	0	0	
Distance	0	0	0	0	
TaxiIn	7110	0.367108	7110	0.367108	
TaxiOut	455	0.0234929	455	0.0234929	
Cancelled	0	0	0	0	
CancellationCode	0	0	0	0	
Diverted	0	0	0	0	
CarrierDelay	689270	35.5889	689270	35.5889	
WeatherDelay	689270	35.5889	689270	35.5889	
NASDelay	689270	35.5889	689270	35.5889	
SecurityDelay	689270	35.5889	689270	35.5889	
LateAircraftDelay	689270	35.5889	689270	35.5889	

FilteredData

	total_missing	%missing	total_null	%null
Date	0	0	0	0
UniqueCarrier	0	0	0	0
FlightNum	0	0	0	0
TailNum	0	0	0	0
Origin	0	0	0	0
Dest	0	0	0	0
Distance	0	0	0	0
DepTime	0	0	0	0
ArrTime	0	0	0	0

```
| DepDelay          |                0 |          0 |             0 |        0 |
| ArrDelay          |                0 |          0 |             0 |        0 |
| ActualElapsedTime |                0 |          0 |             0 |        0 |
| AirTime           |                0 |          0 |             0 |        0 |
| Cancelled         |                0 |          0 |             0 |        0 |
| CancellationCode  |                0 |          0 |             0 |        0 |
```

Apartat 3 Crea columnes noves (velocitat mitjana del vol, si ha arribat tard o no...).

[7]:
```
#Creation of new columns

#Velocity applying the transformations to get kilometers and hours

AvgVelocity=((dataCR['Distance']*1.609344)/(dataCR['AirTime']/60))
dataCR['AvgVelocity']=AvgVelocity

#Check if infinite values arise and turn them to 0 (it appeared 1 value while␣
 ↪testing)
r = dataCR['AvgVelocity'].index[np.isinf(dataCR['AvgVelocity'])]
dataCR['AvgVelocity'][r]=0

#Time spent moving inside the airport in the plane

TaxiTime=(dataCR['ActualElapsedTime']-dataCR['AirTime']).
 ↪convert_dtypes(infer_objects=False)
dataCR['TaxiTime']=TaxiTime

#Classification of flights based on delayed
def tdelay(x):
    if x<=0:
        lt='0'
    elif x<=60:
        lt='L'
    else:
        lt='RL'
    return lt

TypeDelay=dataCR['ArrDelay'].apply(tdelay)
dataCR['TypeDelay']=TypeDelay

#Late value for each flight

def blate(x):
    if x>0:
        l=1
    else:
        l=0
```

```
    return l

Late=dataCR['ArrDelay'].apply(blate)
dataCR['Late']=Late
dataCR.head()
```

[7]:

	Date	UniqueCarrier	FlightNum	TailNum	Origin	Dest	Distance	\
1013914	2008-06-23	MQ	3980	N800AE	SDF	ORD	286	
1347410	2008-08-19	DL	205	N641DL	LAS	ATL	1747	
848465	2008-05-22	AA	553	N428AA	RSW	DFW	1017	
1535041	2008-10-18	WN	2032	N309SW	BHM	MSY	321	
56620	2008-01-09	OO	6434	N763SK	LAX	SAT	1210	

	DepTime	ArrTime	DepDelay	ArrDelay	ActualElapsedTime	AirTime	\
1013914	1452	1516	17	21	84	53	
1347410	1200	1916	7	11	256	212	
848465	1755	2001	50	61	186	169	
1535041	1742	1842	32	22	60	49	
56620	655	1131	40	27	156	137	

	Cancelled	CancellationCode	AvgVelocity	TaxiTime	TypeDelay	Late
1013914	0	N	521.063076	31	L	1
1347410	0	N	795.714331	44	L	1
848465	0	N	581.077934	17	RL	1
1535041	0	N	632.570723	11	L	1
56620	0	N	852.83485	19	L	1

Apartat 4 Fes una taula de les aerolínies amb més endarreriments acumulats.

[8]:
```
#Most delays able

dCR=dataCR
col_names=['ArrCum']
UCg=dCR.groupby(by='UniqueCarrier').sum()
Tflights=dCR.groupby(by='UniqueCarrier').count()

TT=Tflights['Late'].sort_values(ascending=False)
TotF = pd.DataFrame({'Number of flights': TT})
TotF.reset_index(inplace=True)


CDelay=UCg['ArrDelay'].sort_values(ascending=False)
DelayC = pd.DataFrame({'CarrierDelays [min]': CDelay})
DelayC.reset_index(inplace=True)
print(tabulate(DelayC,headers='keys',tablefmt='github',showindex='False'))
```

```
Lnum=UCg['Late'].sort_values(ascending=False)
DelayN = pd.DataFrame({'Number of flights delayed': Lnum})
DelayN.reset_index(inplace=True)
FDel=DelayN.merge(TotF, on='UniqueCarrier')
FDel['%Delayed']=((FDel['Number of flights delayed']*100)/FDel['Number of␣
 ↪flights']).round(2)

print('\n')
print(tabulate(FDel,headers='keys',tablefmt='github',showindex='False'))
```

UniqueCarrier	CarrierDelays [min]
WN	1175878
AA	917037
UA	682895
MQ	667890
OO	612183
XE	542806
DL	472870
CO	420380
EV	399405
YV	388702
US	371160
NW	370135
FL	322272
B6	302345
OH	274179
9E	258686
AS	145705
F9	85258
HA	26365
AQ	1945

UniqueCarrier	Number of flights delayed	Number of flights	%Delayed
WN	33583	38848	86.45
AA	17915	19819	90.39
MQ	13667	14794	92.38
UA	12674	14435	87.8
OO	12478	13446	

```
92.8  |
| DL             |                      10583 |               11894 |
88.98 |
| XE             |                       9858 |               10811 |
91.18 |
| CO             |                       8798 |               10525 |
83.59 |
| US             |                       8575 |               10109 |
84.83 |
| EV             |                       7746 |                8437 |
91.81 |
| NW             |                       7548 |                8196 |
92.09 |
| FL             |                       6781 |                7375 |
91.95 |
| YV             |                       6595 |                6963 |
94.71 |
| OH             |                       5109 |                5448 |
93.78 |
| B6             |                       4945 |                5636 |
87.74 |
| 9E             |                       4795 |                5254 |
91.26 |
| AS             |                       3546 |                4109 |
86.3  |
| F9             |                       2748 |                3010 |
91.3  |
| HA             |                        778 |                 806 |
96.53 |
| AQ             |                         74 |                  85 |
87.06 |
```

Apartat 5 Quins són els vols més llargs? I els més endarrerits? Busca les rutes més llargues i les que acumulen més retards.

[9]:
```
#Largest routes

Rg=dCR[['Origin','Dest','Distance']].groupby(by=['Origin','Dest']).mean()
LongR=Rg.sort_values(by='Distance',ascending=False)
LongR.reset_index(inplace=True)
LongR.index +=1
print('\n')
print(tabulate(LongR.head(10),headers='keys',tablefmt='github'))

#Most Delayed routes
Dg=dCR[['Origin','Dest','ArrDelay']].groupby(by=['Origin','Dest']).sum()
MD=Dg.sort_values(by='ArrDelay',ascending=False)
```

```
MD.reset_index(inplace=True)
MD.index +=1
print('\n')
print(tabulate(MD.
 ↪head(10),headers=['Origin','Dest','AccDelay'],tablefmt='github'))
```

```
|    | Origin   | Dest   |   Distance |
|----|----------|--------|------------|
|  1 | HNL      | EWR    |       4962 |
|  2 | EWR      | HNL    |       4962 |
|  3 | HNL      | ATL    |       4502 |
|  4 | ATL      | HNL    |       4502 |
|  5 | ORD      | HNL    |       4243 |
|  6 | HNL      | ORD    |       4243 |
|  7 | KOA      | ORD    |       4213 |
|  8 | ORD      | OGG    |       4184 |
|  9 | HNL      | MSP    |       3972 |
| 10 | MSP      | HNL    |       3972 |
```

```
|    | Origin   | Dest   |   AccDelay |
|----|----------|--------|------------|
|  1 | ORD      | LGA    |      25206 |
|  2 | LAX      | SFO    |      22913 |
|  3 | ORD      | EWR    |      21494 |
|  4 | ATL      | EWR    |      21437 |
|  5 | LGA      | ORD    |      20999 |
|  6 | ATL      | LGA    |      18928 |
|  7 | DFW      | ORD    |      18446 |
|  8 | SFO      | LAX    |      18115 |
|  9 | LGA      | ATL    |      17844 |
| 10 | EWR      | ATL    |      17750 |
```

Apartat 6 Aporta allò que consideris rellevant.

En els previs apartats ja s'ha anat analitzant prou la informació que considerava rellevant i per no agrupar-la aquí es deixa en els altres apartats i aquí s'afeigeix una comparació per veure els aeroports més transitats, més vols *departed i arrived*.

[10]:
```
Of=dCR.groupby(by='Origin').count()
Ofli=Of['Late'].sort_values(ascending=False)
DO = pd.DataFrame({'Most flights departed': Ofli})
DO.reset_index(inplace=True)
DO.index +=1

print('\n')
```

```
print(tabulate(DO.head(10),headers='keys',tablefmt='github'))

Af=dCR.groupby(by='Dest').count()
Afli=Af['Late'].sort_values(ascending=False)

DA = pd.DataFrame({'Most flights arrived': Afli})
DA.reset_index(inplace=True)
DA.index +=1

print('\n')
print(tabulate(DA.head(10),headers='keys',tablefmt='github'))

TOA=(Af['Late']+Of['Late']).convert_dtypes(infer_objects=False,␣
 ↪convert_integer=True)
TotOA= pd.DataFrame({'Total Flights': TOA.sort_values(ascending=False)})
TotOA.reset_index(inplace=True)
TotOA=TotOA.rename(columns={"index": "Airport"})
TotOA.index +=1
display(TotOA.head(10))


print('\n\033[1m'+'Resum general dataset'+'\033[0m')
dCR.describe().round(2)
```

	Origin	Most flights departed
1	ATL	13544
2	ORD	13083
3	DFW	10017
4	DEN	7679
5	LAX	6162
6	IAH	5998
7	PHX	5747
8	LAS	5512
9	EWR	5454
10	SFO	4527

	Dest	Most flights arrived
1	ORD	11011
2	ATL	11005
3	DFW	7289
4	DEN	6613
5	LAX	6195

```
|  6 | EWR     |                  5826 |
|  7 | LAS     |                  4989 |
|  8 | PHX     |                  4906 |
|  9 | SFO     |                  4895 |
| 10 | IAH     |                  4547 |
```

```
   Airport  Total Flights
1      ATL          24549
2      ORD          24094
3      DFW          17306
4      DEN          14292
5      LAX          12357
6      EWR          11280
7      PHX          10653
8      IAH          10545
9      LAS          10501
10     SFO           9422
```

Resum general dataset

[10]:
```
        FlightNum   Distance    DepTime    ArrTime   DepDelay   ArrDelay  \
count  200000.00  200000.00  200000.00  200000.00  200000.00  200000.00
mean     2177.80     766.75    1518.08    1610.13      43.11      42.19
std      1940.24     574.94     451.70     548.78      53.71      57.16
min         1.00      24.00       1.00       1.00       6.00     -67.00
25%       608.00     338.00    1203.00    1316.00      12.00       9.00
50%      1541.00     607.00    1545.00    1715.00      24.00      24.00
75%      3411.00     999.00    1901.00    2031.00      53.00      56.00
max      9740.00    4962.00    2400.00    2400.00    1710.00    1707.00

        ActualElapsedTime    AirTime  Cancelled  AvgVelocity   TaxiTime  \
count          200000.00  200000.00   200000.0    200000.00  200000.00
mean              133.47     108.46        0.0       638.98      25.00
std                72.12      68.69        0.0       125.92      15.39
min                16.00       0.00        0.0         0.00       2.00
25%                80.00      58.00        0.0       566.57      16.00
50%               116.00      90.00        0.0       650.17      21.00
75%               165.00     137.00        0.0       721.84      29.00
max               724.00     652.00        0.0     10718.23     386.00

             Late
count  200000.00
mean        0.89
std         0.31
min         0.00
25%         1.00
50%         1.00
```

```
75%          1.00
max          1.00
```

1.1.3 Exercici 3

Exporta el dataset net i amb les noves columnes a Excel.

[11]:
```
# saving the excel
data_Final=dataCR
data_Final.to_excel('Airlines_delay_Transformed.xlsx')
print('DataFrame is written to Excel File successfully.')
```

```
DataFrame is written to Excel File successfully.
```