

S02 T05: Exploración de los datos

```
In [32]: import pandas as pd
import numpy as np
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
```

- Ejercicio 1

Descarga el data set Airlines Delay: Airline on-time statistics and delay y cargalo en un pandas Dataframe.

Explora los datos que contiene, y queda únicamente con las columnas que consideres relevantes.

```
In [2]: df = pd.read_csv("DelayedFlights.csv")
df
```

```
Out[2]:
```

	Unnamed: 0	Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarrier
0	0	2008	1	3	4	2003.0	1955	2211.0	2225	
1	1	2008	1	3	4	754.0	735	1002.0	1000	
2	2	2008	1	3	4	628.0	620	804.0	750	
3	4	2008	1	3	4	1829.0	1755	1959.0	1925	
4	5	2008	1	3	4	1940.0	1915	2121.0	2110	
...
1936753	7009710	2008	12	13	6	1250.0	1220	1617.0	1552	
1936754	7009717	2008	12	13	6	657.0	600	904.0	749	
1936755	7009718	2008	12	13	6	1007.0	847	1149.0	1010	
1936756	7009726	2008	12	13	6	1251.0	1240	1446.0	1437	
1936757	7009727	2008	12	13	6	1110.0	1103	1413.0	1418	

1936758 rows × 30 columns

- **En esta primer parte vamos a analizar un poco las columnas y los resultados para averiguar que sirve y que no**

```
In [3]: df.columns # vamos a averiguar que columnas hay
```

```
Out[3]: Index(['Unnamed: 0', 'Year', 'Month', 'DayofMonth', 'DayOfWeek', 'DepTime',
            'CRSDepTime', 'ArrTime', 'CRSArrTime', 'UniqueCarrier', 'FlightNum',
            'TailNum', 'ActualElapsedTime', 'CRSElapsedTime', 'AirTime', 'ArrDelay',
            'DepDelay', 'Origin', 'Dest', 'Distance', 'TaxiIn', 'TaxiOut',
            'Cancelled', 'CancellationCode', 'Diverted', 'CarrierDelay',
            'WeatherDelay', 'NASDelay', 'SecurityDelay', 'LateAircraftDelay'],
            dtype='object')
```

```
In [4]: # también vamos a averiguar cuantas columnas y filas hay
df.shape
```

Out[4]: (1936758, 30)

```
In [6]: #vemos que hay casi unos 2 millones de entradas
# vamos a hacer limpieza de duplicados en las filas
df.drop_duplicates( inplace = True)
df.shape
```

Out[6]: (1936758, 30)

```
In [7]: #podemos ver que no hay duplicados.
# vamos a simplemente las columnas de tiempo deptime y arrtime, para saber si se refiere
#o a la hora en formato hh:mm
df2Dep = df [ "DepTime"]
df2Arr = df [ "ArrTime"]
print (df2Dep.max())
print (df2Dep.min())
print (df2Arr.max())
print (df2Arr.min())
```

2400.0
1.0
2400.0
1.0

In []:

Podemos ver que DepTime i Arrtime son las horas y minutos hh:mm pasados en formato hhmm, sin el ":"

```
In [8]: # analizamos un poco las variables relacionadas con el tiempo ( min, segundos)
df3 = df [['DepTime',
          'CRSDepTime', 'ArrTime', 'CRSArrTime', 'ActualElapsedTime', 'CRSElapsedTime', 'AirT
          'DepDelay', 'CarrierDelay',
          'WeatherDelay', 'NASDelay', 'SecurityDelay', 'LateAircraftDelay']]
df3
```

Out[8]:

	DepTime	CRSDepTime	ArrTime	CRSArrTime	ActualElapsedTime	CRSElapsedTime	AirTime	ArrDelay	De
0	2003.0	1955	2211.0	2225	128.0	150.0	116.0	-14.0	
1	754.0	735	1002.0	1000	128.0	145.0	113.0	2.0	
2	628.0	620	804.0	750	96.0	90.0	76.0	14.0	
3	1829.0	1755	1959.0	1925	90.0	90.0	77.0	34.0	
4	1940.0	1915	2121.0	2110	101.0	115.0	87.0	11.0	
...	
1936753	1250.0	1220	1617.0	1552	147.0	152.0	120.0	25.0	
1936754	657.0	600	904.0	749	127.0	109.0	78.0	75.0	
1936755	1007.0	847	1149.0	1010	162.0	143.0	122.0	99.0	
1936756	1251.0	1240	1446.0	1437	115.0	117.0	89.0	9.0	
1936757	1110.0	1103	1413.0	1418	123.0	135.0	104.0	-5.0	

1936758 rows × 14 columns

Podemos ver DepDelay y ArrDelay **recogen la diferencia ente el tiempo esperado y el tiempo real** de llegada o salida, así como **elapsed time** nos da información del tiempo transcurrido, por lo que las primeras cuatro columnas nos informan de los mismo que las cinco siguientes.

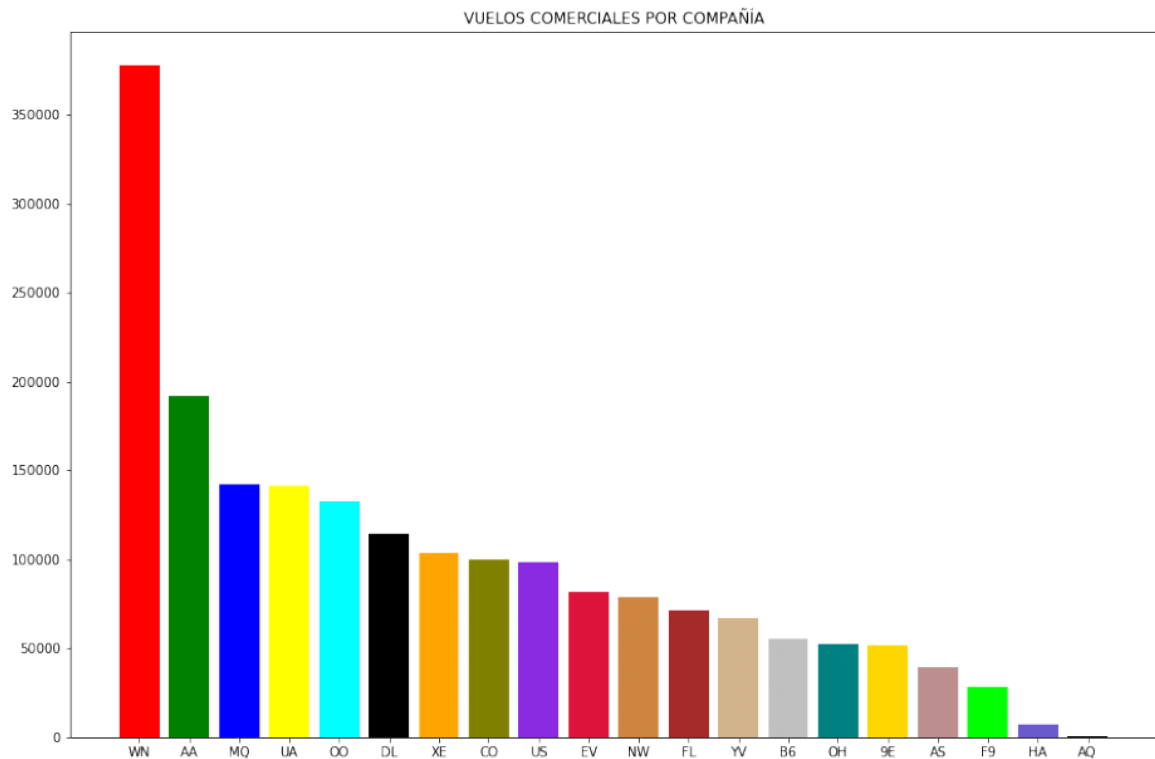
```
In [9]: # construimos un primer data frame con el primer filtrado, dejando fuera las primeras cuatro
# así como las columnas que tienen que ver con la compañía y el número de serie del avión

df4 = df [[ 'Year', 'Month', 'DayofMonth', 'DayOfWeek', 'UniqueCarrier',
            'ActualElapsedTime', 'CRSElapsedTime', 'AirTime', 'ArrDelay',
            'DepDelay', 'Origin', 'Dest', 'Distance', 'TaxiIn', 'TaxiOut',
            'Cancelled', 'CancellationCode', 'Diverted', 'CarrierDelay',
            'WeatherDelay', 'NASDelay', 'SecurityDelay', 'LateAircraftDelay']]

# analicemos un poco la variable " UniqueCarrier" para ver que nos aporta.
df4 [[ "UniqueCarrier"]]. value_counts()
```

```
Out[9]: UniqueCarrier
WN          377602
AA          191865
MQ          141920
UA          141426
OO          132433
DL          114238
XE          103663
CO          100195
US           98425
EV           81877
NW           79108
FL           71284
YV           67063
B6           55315
OH           52657
9E           51885
AS           39293
F9           28269
HA            7490
AQ             750
dtype: int64
```

```
In [34]: img1 = mpimg.imread("fig1.png")
plt.figure(figsize=(20, 20))
plt.imshow(img1)
plt.title('Imagen1')
plt.axis('off')
plt.show()
```



```
In [10]: df4 [[ "Cancelled"]]. value_counts()# miramos también los vuelos cancelados
```

```
Out[10]: Cancelled
0          1936125
1             633
dtype: int64
```

```
In [11]: # observamos unos 600 vuelos sobre dos millones
df4 [[ "CancellationCode"]]. value_counts() # así como miramos sus códigos de cancelación
```

```
Out[11]: CancellationCode
N          1936125
B             307
A             246
C              80
dtype: int64
```

```
In [12]: # los términos A, B y C no nos aportan mucho, así que eliminamos esta columna, pero si de
# para saber si hay compañías más lentas que otras.
# miramos los vuelos desviados
df4 [[ "Diverted"]]. value_counts()
```

```
Out[12]: Diverted
0          1929004
1             7754
dtype: int64
```

```
In [13]: # dejamos estas columnas

df5 = df [[ 'Year', 'Month', 'DayofMonth', 'DayOfWeek', 'UniqueCarrier',
            'ActualElapsedTime', 'CRSElapsedTime', 'AirTime', 'ArrDelay',
```

```
'DepDelay', 'Origin', 'Dest', 'Distance',  
'Cancelled', 'Diverted', 'CarrierDelay',  
'WeatherDelay', 'NASDelay', 'SecurityDelay', 'LateAircraftDelay']]  
df5
```

Out[13]:

	Year	Month	DayofMonth	DayOfWeek	UniqueCarrier	ActualElapsedTime	CRSElapsedTime	AirTime	Arrl
0	2008	1	3	4	WN	128.0	150.0	116.0	
1	2008	1	3	4	WN	128.0	145.0	113.0	
2	2008	1	3	4	WN	96.0	90.0	76.0	
3	2008	1	3	4	WN	90.0	90.0	77.0	
4	2008	1	3	4	WN	101.0	115.0	87.0	
...	
1936753	2008	12	13	6	DL	147.0	152.0	120.0	
1936754	2008	12	13	6	DL	127.0	109.0	78.0	
1936755	2008	12	13	6	DL	162.0	143.0	122.0	
1936756	2008	12	13	6	DL	115.0	117.0	89.0	
1936757	2008	12	13	6	DL	123.0	135.0	104.0	

1936758 rows × 20 columns

- **Ejercicio 2:Haga un informe completo de la fecha set:.**
- Resume estadísticamente las columnas de interés
- Encuentra cuántos datos faltantes hay por columna
- Crea columnas nuevas (velocidad media del vuelo, si ha llegado tarde o no...)
- Tabla de las aerolíneas con mayores retrasos acumulados
- ¿Cuáles son los vuelos más largos? ¿Y los más atrasados?
- Etc.
- Análisis de los valores nulo

```
In [14]: # vamos a ver primero los datos faltantes por columnas  
df5.isnull().sum()
```

Out[14]:

Year	0
Month	0
DayofMonth	0
DayOfWeek	0
UniqueCarrier	0
ActualElapsedTime	8387
CRSElapsedTime	198
AirTime	8387
ArrDelay	8387
DepDelay	0
Origin	0

```
Dest 0
Distance 0
Cancelled 0
Diverted 0
CarrierDelay 689270
WeatherDelay 689270
NASDelay 689270
SecurityDelay 689270
LateAircraftDelay 689270
dtype: int64
```

```
In [15]: # podemos ver que 'ActualElapsedTime', 'AirTime', 'ArrDelay, tienen 8387 datos faltantes
# vamos a hacer un filtro para ver si se observa alguna relación entre ellos, antes de as:
# un valor a estos nulos. Es fácil ver que los 8387 vuelos sin tiempo registrados
# coinciden con los vuelos derivados y cancelados. 7754 +633
# así que vamos a ver los vuelos cancelados
df5 [df5["Cancelled"]==1 ]

#también se ve que las últimas 5 columnas tiene más valores vacíos ya que son la causa de:
# y registran el tiempo asignado a esa causa, por lo cual es más fácil que contengan valores
```

Out[15]:

	Year	Month	DayofMonth	DayOfWeek	UniqueCarrier	ActualElapsedTime	CRSElapsedTime	AirTime	Arr
1542406	2008	10	27	1	WN	NaN	60.0	NaN	
1546593	2008	10	25	6	XE	NaN	107.0	NaN	
1547161	2008	10	22	3	XE	NaN	72.0	NaN	
1547178	2008	10	22	3	XE	NaN	63.0	NaN	
1548271	2008	10	15	3	XE	NaN	72.0	NaN	
...	
1934590	2008	12	7	7	DL	NaN	130.0	NaN	
1935491	2008	12	10	3	DL	NaN	125.0	NaN	
1935651	2008	12	10	3	DL	NaN	123.0	NaN	
1935876	2008	12	11	4	DL	NaN	144.0	NaN	
1936470	2008	12	12	5	DL	NaN	64.0	NaN	

633 rows × 20 columns

```
In [16]: # vamos a intentar encontrar una causa a los 198 nulos CRSElapsedTime
df5[df5.CRSElapsedTime.isnull()]
```

Out[16]:

	Year	Month	DayofMonth	DayOfWeek	UniqueCarrier	ActualElapsedTime	CRSElapsedTime	AirTime	Arr
138532	2008	1	31	4	9E	NaN	NaN	NaN	
138574	2008	1	10	4	9E	NaN	NaN	NaN	
138697	2008	1	7	1	9E	NaN	NaN	NaN	
138786	2008	1	5	6	9E	NaN	NaN	NaN	
138946	2008	1	17	4	9E	NaN	NaN	NaN	
...	
1501260	2008	9	14	7	9E	NaN	NaN	NaN	

	Year	Month	DayofMonth	DayOfWeek	UniqueCarrier	ActualElapsedTime	CRSElapsedTime	AirTime	Arrl
1501426	2008	9	28	7	9E	NaN	NaN	NaN	
1501463	2008	9	26	5	9E	NaN	NaN	NaN	
1501786	2008	9	11	4	9E	NaN	NaN	NaN	
1502025	2008	9	28	7	9E	NaN	NaN	NaN	

198 rows × 20 columns

```
In [17]: df5[["Diverted"]][df5.CRSElapsedTime.isnull()].value_counts()
# vemos que todos los nulos son de vuelos derivados por lo que tampoco los vamos a cambiar
# o algún método numérico
```

```
Out[17]: Diverted
1          198
dtype: int64
```

- Resumen estadístico de las columnas de interés

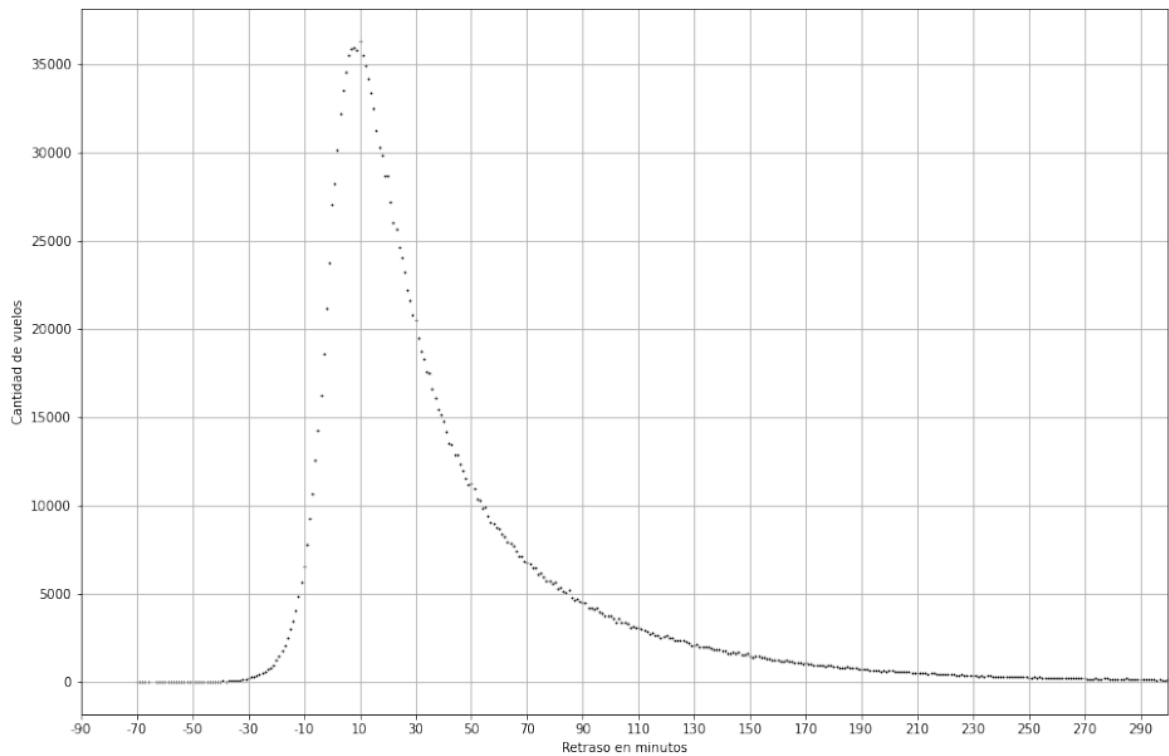
```
In [19]: df6 = df[['ActualElapsedTime', 'AirTime', 'ArrDelay',
                  'Distance']]

df7 = df6.describe()
df7
```

```
Out[19]:
```

	ActualElapsedTime	AirTime	ArrDelay	Distance
count	1.928371e+06	1.928371e+06	1.928371e+06	1.936758e+06
mean	1.333059e+02	1.082771e+02	4.219988e+01	7.656862e+02
std	7.206007e+01	6.864261e+01	5.678472e+01	5.744797e+02
min	1.400000e+01	0.000000e+00	-1.090000e+02	1.100000e+01
25%	8.000000e+01	5.800000e+01	9.000000e+00	3.380000e+02
50%	1.160000e+02	9.000000e+01	2.400000e+01	6.060000e+02
75%	1.650000e+02	1.370000e+02	5.600000e+01	9.980000e+02
max	1.114000e+03	1.091000e+03	2.461000e+03	4.962000e+03

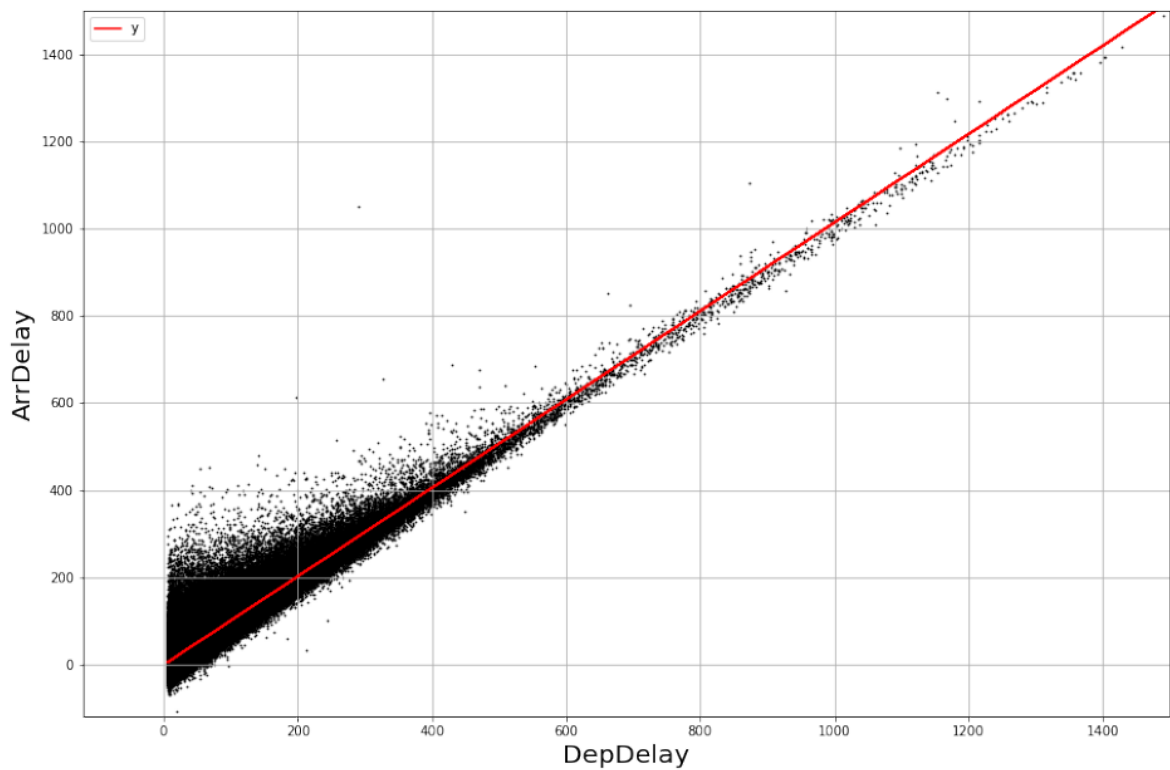
```
In [35]: img2 = mpimg.imread("fig2.png")
plt.figure(figsize=(20, 20))
plt.imshow(img2)
plt.title('Imagen2')
plt.axis('off')
plt.show()
```



Observamos que la media de un vuelo, desde que sale, despegue y hasta que aterriza, se encuentra alrededor de las 2 horas y 13 minutos. Por otro lado la media del retraso en cada vuelo, o sea, el retraso hasta llegar a destino se encuentra en los 42 minutos, siendo la media de tiempo de la que dura el viaje en el aire es de 108 minutos. También podemos ver que hay una relación casi lineal entre el tiempo de retraso en el despegue, con el retraso de llegada

In [38]:

```
img4 = mpimg.imread("fig4.png")
plt.figure(figsize=(20, 20))
plt.imshow(img4)
plt.title('Imagen4')
plt.axis('off')
plt.show()
```

- Tabla de las aerolíneas con mayores retrasos acumulados

In [20]:

```
df8 = df5[["UniqueCarrier", "ArrDelay"]]
comp_delay = df8.groupby('UniqueCarrier')['ArrDelay'].sum().sort_values()

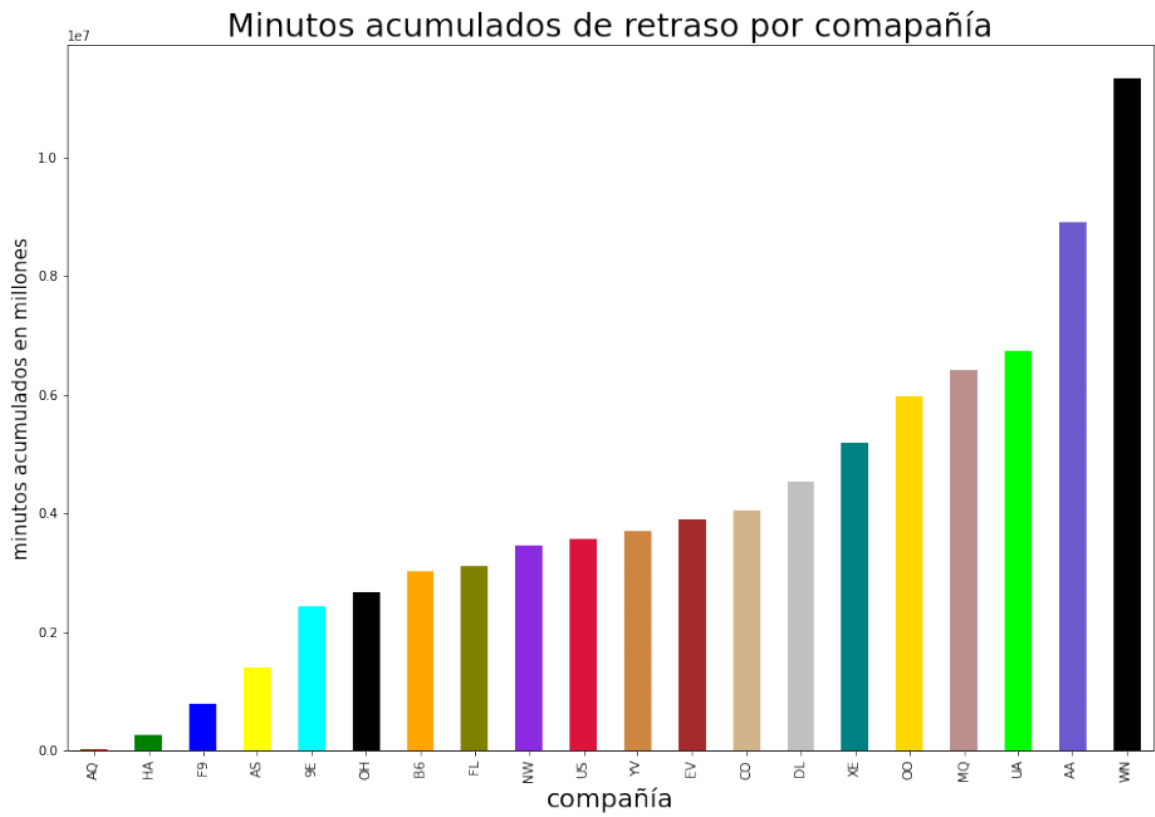
# podremos ver el retraso de las compañías
for index, value in comp_delay.items():
    if value > 5000000:
        print(index, value)
```

```
XE 5176042.0
OO 5978936.0
MQ 6396704.0
UA 6733013.0
AA 8889066.0
WN 11319092.0
```

In [49]:

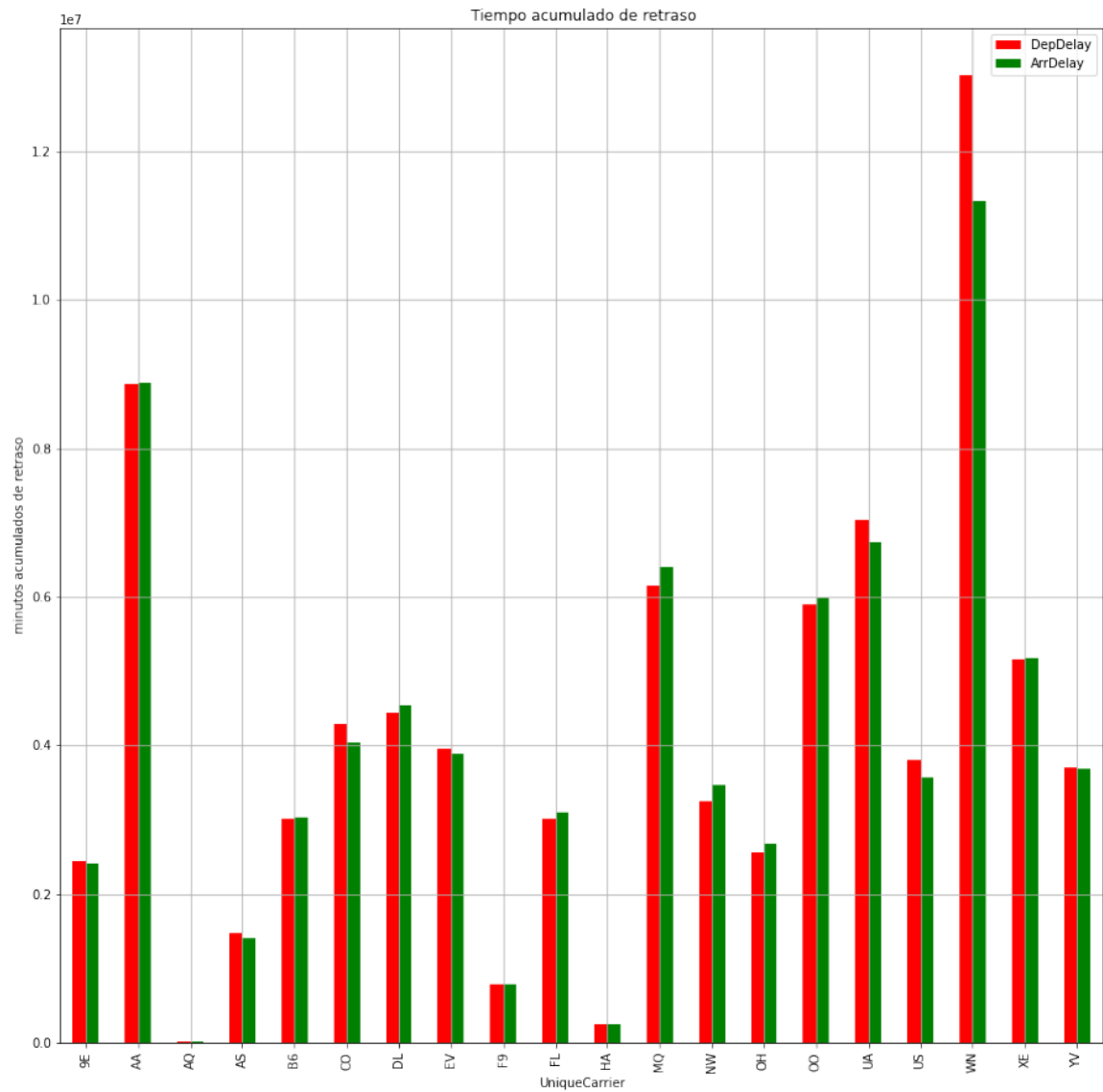
```
img3 = mpimg.imread("fig3.png")

plt.figure(figsize=(20,20))
plt.imshow(img3)
plt.title('Imagen3')
plt.axis('off')
plt.show()
```

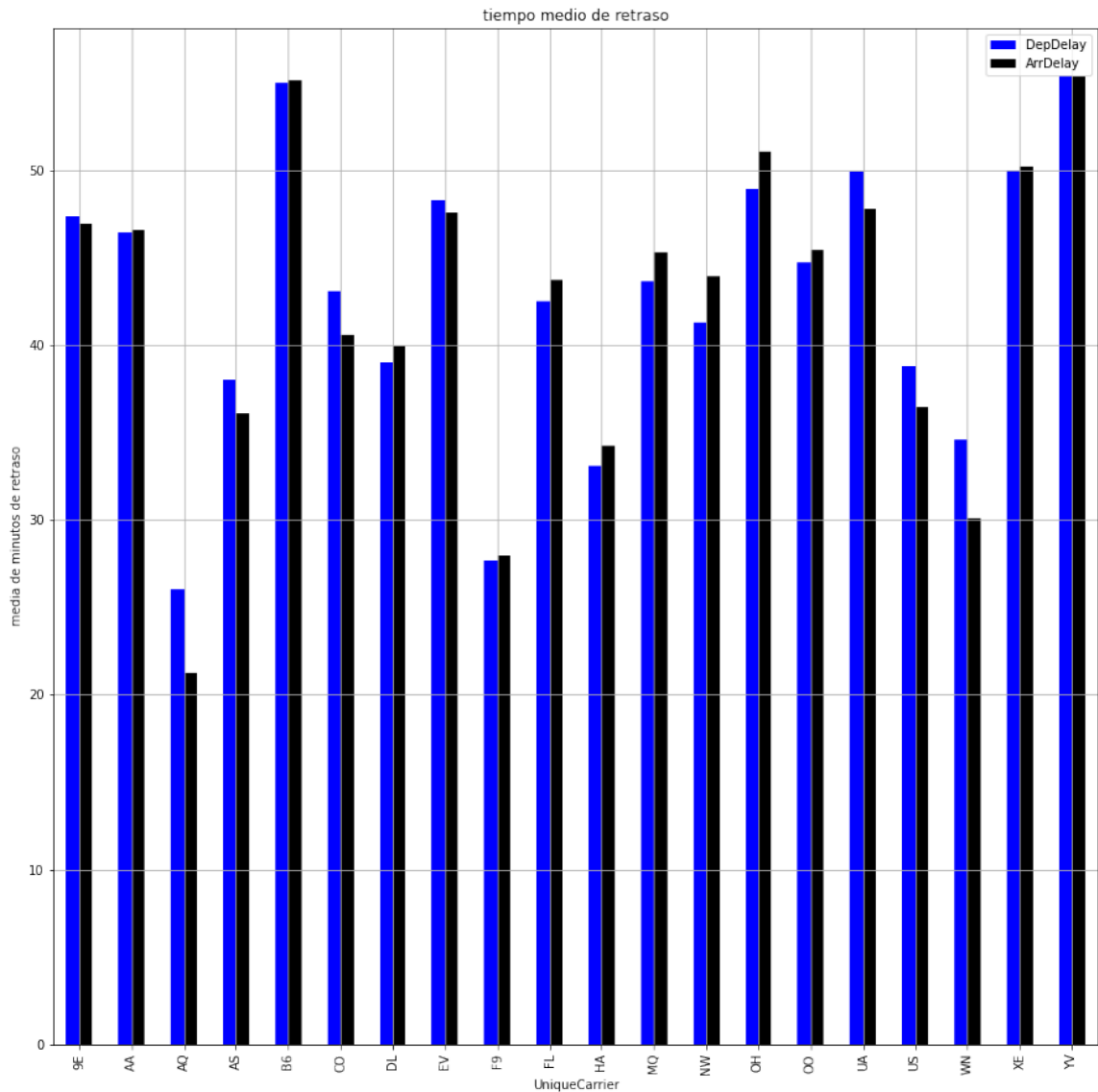


In [50]:

```
img5 = mpimg.imread("fig5.png")
plt.figure(figsize=(20,20))
plt.imshow(img5)
plt.title('Imagen5')
plt.axis('off')
plt.show()
```



```
In [51]: img6 = mpimg.imread("fig6.png")
plt.figure(figsize=(20,20))
plt.imshow(img6)
plt.title('Imagen6')
plt.axis('off')
plt.show()
```



- Crea columnas nuevas (velocidad media del vuelo, si ha llegado tarde o no...)

In [21]:

```
# fragmentamos el DF en 3 columnas,

time=df5["AirTime"]# sea el tiempo del viaje
dist= df5["Distance"]# la distancia recorrida
retraso = df5["ArrDelay"]# si ha llegado tarde al aeropuerto
vel = dist/ time # cálculo de la columna velocidad
vel = pd.Series([round(y,2) for y in vel])

vlista = vel.tolist()# pasamos la Serie a Lista
# vamos a insertar la velocidad
df9 = df5
df9.insert(13,"Velocidad",vlista)
df9
```

Out[21]:

	Year	Month	DayofMonth	DayOfWeek	UniqueCarrier	ActualElapsedTime	CRSElapsedTime	AirTime	Arri
0	2008	1	3	4	WN	128.0	150.0	116.0	
1	2008	1	3	4	WN	128.0	145.0	113.0	
2	2008	1	3	4	WN	96.0	90.0	76.0	
3	2008	1	3	4	WN	90.0	90.0	77.0	
4	2008	1	3	4	WN	101.0	115.0	87.0	
...
1936753	2008	12	13	6	DL	147.0	152.0	120.0	
1936754	2008	12	13	6	DL	127.0	109.0	78.0	
1936755	2008	12	13	6	DL	162.0	143.0	122.0	
1936756	2008	12	13	6	DL	115.0	117.0	89.0	
1936757	2008	12	13	6	DL	123.0	135.0	104.0	

1936758 rows × 21 columns

In [23]:

```
# vamos a convertir la serie en una mascara array, para encontrar los valores con retraso
# para luego concertirla en una serie y adjuntarla al DF

retarr = retraso.to_numpy()
def fun ( mat):# buscamos los valores ciertos o falso
    mask = (mat > 0)
    return mask
vect =np. vectorize ( fun)# vectoriza el cálculo en una sola operación
mascara = np.array (vect (retarr))
np.unique(mascara, return_counts=True)
```

Out[23]: (array([False, True]), array([213343, 1723415], dtype=int64))

In [24]:

```
mask_DF= pd.Series(mascara.copy())

mask_DF= mask_DF.map({True: 'yes', False: 'no'})
mask_DF=mask_DF.to_frame(name= "Delay")
mask_DF
```

Out[24]:

	Delay
0	no
1	yes
2	yes
3	yes
4	yes
...	...
1936753	yes
1936754	yes
1936755	yes

Delay	
1936756	yes
1936757	no

1936758 rows × 1 columns

```
In [25]: df10=pd.concat([df9,mask_DF], axis=1)
df10
```

```
Out[25]:
```

	Year	Month	DayofMonth	DayOfWeek	UniqueCarrier	ActualElapsedTime	CRSElapsedTime	AirTime	ArrD
0	2008	1	3	4	WN	128.0	150.0	116.0	
1	2008	1	3	4	WN	128.0	145.0	113.0	
2	2008	1	3	4	WN	96.0	90.0	76.0	
3	2008	1	3	4	WN	90.0	90.0	77.0	
4	2008	1	3	4	WN	101.0	115.0	87.0	
...
1936753	2008	12	13	6	DL	147.0	152.0	120.0	
1936754	2008	12	13	6	DL	127.0	109.0	78.0	
1936755	2008	12	13	6	DL	162.0	143.0	122.0	
1936756	2008	12	13	6	DL	115.0	117.0	89.0	
1936757	2008	12	13	6	DL	123.0	135.0	104.0	

1936758 rows × 22 columns

- ¿Cuáles son los vuelos más largos? ¿Y los más atrasados?

```
In [26]: # el vuelo más atrasado
maxdelay=df10["ArrDelay"].max()
df10[ df10 ["ArrDelay"]== maxdelay]
```

```
Out[26]:
```

	Year	Month	DayofMonth	DayOfWeek	UniqueCarrier	ActualElapsedTime	CRSElapsedTime	AirTime	ArrD
322516	2008	2	3	7	NW	459.0	455.0	437.0	24

1 rows × 22 columns

```
In [27]: # el vuelo más largo
maxtime=df10["AirTime"].max()
df10[ df10 ["AirTime"]== maxtime]
```

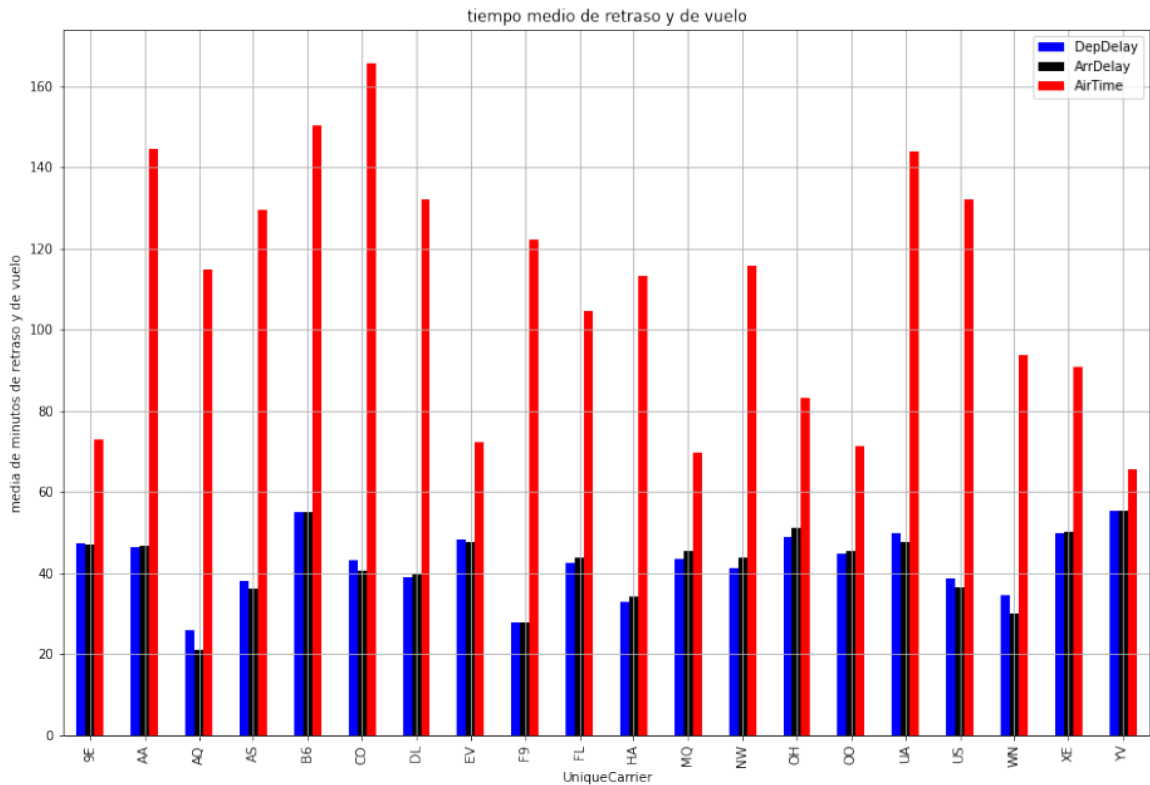
```
Out[27]:
```

	Year	Month	DayofMonth	DayOfWeek	UniqueCarrier	ActualElapsedTime	CRSElapsedTime	AirTime	ArrD
1488690	2008	9	9	2	HA	1114.0	355.0	1091.0	1

1 rows × 22 columns

```
In [37]: img7 = mpimg.imread("fig7.png")
plt.figure(figsize=(20, 20))
plt.imshow(img7)
plt.title('Imagen7')
plt.axis('off')
plt.show()
```

Imagen7



```
In [28]: #Limpiar el Data Set exportamos el df10 a excel

# la información me parece bastante relevante en muchas columnas, pero el archivo es demas
#para pasarlo a excell, así que vamos a reducir por fechas(todo vuelos del 2008)

df10["Month"].value_counts()
```

```
Out[28]: 12    203385
6      200914
3      200842
2      189534
1      183527
7      182945
8      162648
4      155264
5      153072
11     105563
10     103525
9       95539
Name: Month, dtype: int64
```

```
In [29]: # vamos a limpiar a todavía más el DF en columnas
df10.columns
```

```
Out[29]: Index(['Year', 'Month', 'DayOfMonth', 'DayOfWeek', 'UniqueCarrier',
```

```
'ActualElapsedTime', 'CRSElapsedTime', 'AirTime', 'ArrDelay',  
'DepDelay', 'Origin', 'Dest', 'Distance', 'Velocidad', 'Cancelled',  
'Diverted', 'CarrierDelay', 'WeatherDelay', 'NASDelay', 'SecurityDelay',  
'LateAircraftDelay', 'Delay'],  
dtype='object')
```

In [30]:

```
# opto limpiar por meses  
df11 = df10[['Month', 'UniqueCarrier',  
            'ActualElapsedTime', 'CRSElapsedTime', 'AirTime', 'ArrDelay',  
            'DepDelay', 'Origin', 'Dest', 'Distance', 'Velocidad', 'Cancelled',  
            'Diverted', 'Delay']]  
df11 = df11[df11["Month"]>8]  
df11.to_excel("Delay.xlsx")
```

In []: