Actividad 1

November 20, 2022

[1]:
```
from pyspark.sql import SparkSession

sparkSession = SparkSession.builder.appName("actividad_1").getOrCreate()
```

[3]:
```
sparkContext = sparkSession.sparkContext

cars = sparkContext.textFile("../data/cars.csv")

header_line = cars.first()
print(header_line)
list(enumerate(header_line.split(';')))
```

```
Car;MPG;Cylinders;Displacement;Horsepower;Weight;Acceleration;Model;Origin
```

[3]:
```
[(0, 'Car'),
 (1, 'MPG'),
 (2, 'Cylinders'),
 (3, 'Displacement'),
 (4, 'Horsepower'),
 (5, 'Weight'),
 (6, 'Acceleration'),
 (7, 'Model'),
 (8, 'Origin')]
```

[4]:
```
#punto 1 : Mostrar las primeras 5 filas
cars = cars.filter(lambda line: line != header_line)

cars.take(5)
```

[4]:
```
['Chevrolet Chevelle Malibu;18.0;8;307.0;130.0;3504.;12.0;70;US',
 'Buick Skylark 320;15.0;8;350.0;165.0;3693.;11.5;70;US',
 'Plymouth Satellite;18.0;8;318.0;150.0;3436.;11.0;70;US',
 'AMC Rebel SST;16.0;8;304.0;150.0;3433.;12.0;70;US',
 'Ford Torino;17.0;8;302.0;140.0;3449.;10.5;70;US']
```

[5]:
```
cars_tuples_rdd = cars.map(lambda line: line.split(';'))

print(cars_tuples_rdd.take(3))
```

```
[['Chevrolet Chevelle Malibu', '18.0', '8', '307.0', '130.0', '3504.', '12.0',
'70', 'US'], ['Buick Skylark 320', '15.0', '8', '350.0', '165.0', '3693.',
'11.5', '70', 'US'], ['Plymouth Satellite', '18.0', '8', '318.0', '150.0',
'3436.', '11.0', '70', 'US']]
```

[43]:
```
cars_tuples_rdd.cache()



cars_tuples_format_rdd = cars_tuples_rdd.map(
    lambda fields_list : (fields_list[0], #CARS
                          int(fields_list[2]), #CYLINDERS
                          float(fields_list[4]), #HORSEPOWER
                          float(fields_list[5]), #WEIGHT
                          float(fields_list[6]), #ACCELERATION
                          fields_list[8]  #ORIGIN
                         )
)
cars_tuples_format_rdd.take(5)
```

[43]:
```
[('Chevrolet Chevelle Malibu', 8, 130.0, 3504.0, 12.0, 'US'),
 ('Buick Skylark 320', 8, 165.0, 3693.0, 11.5, 'US'),
 ('Plymouth Satellite', 8, 150.0, 3436.0, 11.0, 'US'),
 ('AMC Rebel SST', 8, 150.0, 3433.0, 12.0, 'US'),
 ('Ford Torino', 8, 140.0, 3449.0, 10.5, 'US')]
```

[50]:
```
#PUNTO 2 Muestra las columnas 'Cars' y 'Cylinders' de todos los vehículos que␣
 ↪sean de 'Europa'.
cars_eu = cars_tuples_format_rdd.filter(lambda x :(x[5].find("Europe")) != -1).
 ↪map(
    lambda x: (x[0], #CARS
               x[1], #CYLINDERS
               x[5] #ORIGIN
              )
)
cars_eu.collect()
```

[50]:
```
[('Citroen DS-21 Pallas', 4, 'Europe'),
 ('Volkswagen 1131 Deluxe Sedan', 4, 'Europe'),
 ('Peugeot 504', 4, 'Europe'),
 ('Audi 100 LS', 4, 'Europe'),
 ('Saab 99e', 4, 'Europe'),
 ('BMW 2002', 4, 'Europe'),
 ('Volkswagen Super Beetle 117', 4, 'Europe'),
 ('Opel 1900', 4, 'Europe'),
 ('Peugeot 304', 4, 'Europe'),
 ('Fiat 124B', 4, 'Europe'),
```

```
('Volkswagen Model 111', 4, 'Europe'),
('Volkswagen Type 3', 4, 'Europe'),
('Volvo 145e (sw)', 4, 'Europe'),
('Volkswagen 411 (sw)', 4, 'Europe'),
('Peugeot 504 (sw)', 4, 'Europe'),
('Renault 12 (sw)', 4, 'Europe'),
('Volkswagen Super Beetle', 4, 'Europe'),
('Fiat 124 Sport Coupe', 4, 'Europe'),
('Fiat 128', 4, 'Europe'),
('Opel Manta', 4, 'Europe'),
('Audi 100LS', 4, 'Europe'),
('Volvo 144ea', 4, 'Europe'),
('Saab 99le', 4, 'Europe'),
('Audi Fox', 4, 'Europe'),
('Volkswagen Dasher', 4, 'Europe'),
('Opel Manta', 4, 'Europe'),
('Fiat 128', 4, 'Europe'),
('Fiat 124 TC', 4, 'Europe'),
('Fiat x1.9', 4, 'Europe'),
('Volkswagen Dasher', 4, 'Europe'),
('Volkswagen Rabbit', 4, 'Europe'),
('Audi 100LS', 4, 'Europe'),
('Peugeot 504', 4, 'Europe'),
('Volvo 244DL', 4, 'Europe'),
('Saab 99LE', 4, 'Europe'),
('Fiat 131', 4, 'Europe'),
('Opel 1900', 4, 'Europe'),
('Renault 12tl', 4, 'Europe'),
('Volkswagen Rabbit', 4, 'Europe'),
('Volkswagen Rabbit', 4, 'Europe'),
('Volvo 245', 4, 'Europe'),
('Peugeot 504', 4, 'Europe'),
('Mercedes-Benz 280s', 6, 'Europe'),
('Renault 5 GTL', 4, 'Europe'),
('Volkswagen Rabbit Custom', 4, 'Europe'),
('Volkswagen Dasher', 4, 'Europe'),
('BMW 320i', 4, 'Europe'),
('Volkswagen Rabbit Custom Diesel', 4, 'Europe'),
('Audi 5000', 5, 'Europe'),
('Volvo 264gl', 6, 'Europe'),
('Saab 99gle', 4, 'Europe'),
('Peugeot 604sl', 6, 'Europe'),
('Volkswagen Scirocco', 4, 'Europe'),
('Volkswagen Rabbit Custom', 4, 'Europe'),
('Mercedes Benz 300d', 5, 'Europe'),
('Peugeot 504', 4, 'Europe'),
('Fiat Strada Custom', 4, 'Europe'),
```

```
 ('Volkswagen Rabbit', 4, 'Europe'),
 ('Audi 4000', 4, 'Europe'),
 ('Volkswagen Rabbit C (Diesel)', 4, 'Europe'),
 ('Volkswagen Dasher (diesel)', 4, 'Europe'),
 ('Audi 5000s (diesel)', 5, 'Europe'),
 ('Mercedes-Benz 240d', 4, 'Europe'),
 ('Renault Lecar Deluxe', 4, 'Europe'),
 ('Volkswagen Rabbit', 4, 'Europe'),
 ('Triumph TR7 Coupe', 4, 'Europe'),
 ('Volkswagen Jetta', 4, 'Europe'),
 ('Renault 18i', 4, 'Europe'),
 ('Peugeot 505s Turbo Diesel', 4, 'Europe'),
 ('Saab 900s', 4, 'Europe'),
 ('Volvo Diesel', 6, 'Europe'),
 ('Volkswagen Rabbit l', 4, 'Europe'),
 ('Volkswagen Pickup', 4, 'Europe')]
```

[69]:
```
cars_total = cars_tuples_format_rdd.map( lambda x : x[5])\
                                   .countByValue()

cars_total
```

[69]:
```
defaultdict(int, {'US': 254, 'Europe': 73, 'Japan': 79})
```

[78]:
```
cars_us = cars_tuples_format_rdd.filter(lambda x :(x[5].find("US")) != -1).map(
    lambda x: (x[2], #HORSEPOWER
               x[3], #WEIGHT
               x[4], #ACCELERATION
               x[5] #ORIGIN
           )
)

cars_japan = cars_tuples_format_rdd.filter(lambda x :(x[5].find("Japan")) !=␣
 ↪-1).map(
    lambda x: (x[2], #HORSEPOWER
               x[3], #WEIGHT
               x[4], #ACCELERATION
               x[5] #ORIGIN
           )
)

cars_europe = cars_tuples_format_rdd.filter(lambda x :(x[5].find("Europe")) !=␣
 ↪-1).map(
    lambda x: (x[2], #HORSEPOWER
               x[3], #WEIGHT
               x[4], #ACCELERATION
               x[5] #ORIGIN
```

```
            )
)

print(cars_europe.map( lambda x : x[3]).countByValue())
print(cars_japan.map( lambda x : x[3]).countByValue())
print(cars_us.map( lambda x : x[3]).countByValue())
```

```
defaultdict(<class 'int'>, {'Europe': 73})
defaultdict(<class 'int'>, {'Japan': 79})
defaultdict(<class 'int'>, {'US': 254})
```

[182]:
```
# PUNTO 3 Obtener la media de 'Horsepower', 'Weight' y 'Acceleration' por␣
 ↪'Origen'.
europe_horse= cars_europe.map(
    lambda x : (x[0])
    ).mean()
europe_weight= cars_europe.map(
    lambda x : (x[1])
    ).mean()
europe_acceleration= cars_europe.map(
    lambda x : (x[2])
    ).mean()

us_horse= cars_us.map(
    lambda x : (x[0])
    ).mean()
us_weight= cars_us.map(
    lambda x : (x[1])
    ).mean()
us_acceleration= cars_us.map(
    lambda x : (x[2])
    ).mean()

japan_horse= cars_japan.map(
    lambda x : (x[0])
    ).mean()
japan_weight= cars_japan.map(
    lambda x : (x[1])
    ).mean()
japan_acceleration= cars_japan.map(
    lambda x : (x[2])
    ).mean()

print("La media de caballos de EUROPA es : " , round(europe_horse,2))
print("La media de peso de EUROPA es : ",  round(europe_weight,2))
print("La media de aceleración de EUROPA es : " , round(europe_acceleration,2))
```

```
print("---------")
print("La media de caballos de US es : " , round(us_horse,2))
print("La media de peso de US es : ",  round(us_weight,2))
print("La media de aceleración de US es : " , round(us_acceleration,2))
print("---------")
print("La media de caballos de Japon es : " , round(japan_horse,2))
print("La media de peso de Japon es : ",  round(japan_weight,2))
print("La media de aceleración de Japon es : " , round(japan_acceleration,2))
```

```
La media de caballos de EUROPA es :  78.78
La media de peso de EUROPA es :  2431.49
La media de aceleración de EUROPA es :  16.82
---------
La media de caballos de US es :  118.01
La media de peso de US es :  3372.7
La media de aceleración de US es :  14.94
---------
La media de caballos de Japon es :  79.84
La media de peso de Japon es :  2221.23
La media de aceleración de Japon es :  16.17
```

[197]:
```
cars_ratio = cars_tuples_format_rdd.map(
    lambda x: (round((x[2] and x[3] /x[2]),2), # peso / caballos , evalua la␣
 ↪división por si hay un 0, entonces devolverá 0 (si no haces este control, da␣
 ↪error)
                x[1]
    )
)
cars_ratio.take(5)
```

[197]:
```
[(26.95, 8), (22.38, 8), (22.91, 8), (22.89, 8), (24.64, 8)]
```

[198]:
```
print(cars_ratio.map( lambda x : x[1]).countByValue())
```

```
defaultdict(<class 'int'>, {8: 108, 4: 207, 6: 84, 3: 4, 5: 3})
```

[214]:
```
# PUNTO 4 Calcular el ratio entre potencia y peso y, a continuación, sacar la␣
 ↪media por cantidad de cilindros .

cars_ratio_3 = cars_ratio.filter(lambda x :(x[1] ==3)).map(lambda i : (i[0])).
 ↪mean()
cars_ratio_4 = cars_ratio.filter(lambda x :(x[1] ==4)).map(lambda i : (i[0])).
 ↪mean()
cars_ratio_5 = cars_ratio.filter(lambda x :(x[1] ==5)).map(lambda i : (i[0])).
 ↪mean()
cars_ratio_8 = cars_ratio.filter(lambda x :(x[1] ==8)).map(lambda i : (i[0])).
 ↪mean()
```

```
print("La media del ratio peso-potencia con 3 cilindros es: ",␣
 ↪round(cars_ratio_3,2))
print("La media del ratio peso-potencia con 4 cilindros es: ",␣
 ↪round(cars_ratio_4,2))
print("La media del ratio peso-potencia con 5 cilindros es: ",␣
 ↪round(cars_ratio_5,2))
print("La media del ratio peso-potencia con 8 cilindros es: ",␣
 ↪round(cars_ratio_8,2))
```

```
La media del ratio peso-potencia con 3 cilindros es:  24.14
La media del ratio peso-potencia con 4 cilindros es:  29.25
La media del ratio peso-potencia con 5 cilindros es:  39.12
La media del ratio peso-potencia con 8 cilindros es:  26.45
```

[ ]: