Memoria Práctica BICIMAD - Programación Paralela

Gabriel Alba Serrano
Pablo Fernández del Amo
Joaquín Velarde Noval

1. Definición del problema planteado

Una empresa del sector del transporte está actualmente ofertando un servicio de bicicletas de uso compartido teniendo preferencia por lanzar su campaña en el Distrito del Retiro por proximidad a sus oficinas. Su objetivo es maximizar sus beneficios, para ello se está planteando una reorganización de los puntos de estacionamiento de sus bicicletas, y además, una nueva campaña de marketing para atraer a antiguos usuarios del servicio de bicicletas de uso compartido, ya que las tendencias de este servicio cambiaron después de la pandemia de COVID-19. La empresa pretende que la campaña empiece a funcionar en marzo del próximo año, y acabada la campaña se espera que los niveles de popularidad de la marca ya sean más altos.

2. Idea de la solución del problema

Como la empresa que quiere realizar la campaña no se fundó hasta 2021 y necesita datos pre-pandemia, vamos a utilizar los de la empresa de BiciMAD (datos que son públicos). Resolveremos el problema usando pyspark, obteniendo cuál es el perfil de un usuario medio, y cuáles son las estaciones de BiciMAD más concurridas. Además, en caso de que la empresa quiera acceder a los datos de una sola zona, por tener esta más nivel adquisitivo de media, se podrá hacer uso de un filtro opcional de zona, además del filtro opcional de estación del año (invierno, primavera, otoño, verano), obteniendo así únicamente los datos de la estación que se indique.

3. Diseño de la solución

En primer lugar traducimos los archivos *json* mediante la función *line_info*, complementada con la *date_converter* para descifrar las fechas, ya que éstas vienen en un formato poco amable, difícil de tratar. Declaramos algunas variables globales que el usuario puede cambiar si lo desea, en función de si quiere obtener los datos de una época del año o de un distrito de Madrid, y así sacar conclusiones en función de esos datos. En caso de ser utilizadas, se aplicarán los filtros *filter_by_district* y

filter_by_season al RDD antes de proceder al análisis.

A continuación, nos enfocamos en ver las estaciones de bicicletas más concurridas según el día de la semana, la edad de los usuarios y el tipo de usuario. Para ello utilizamos las funciones *spot_more_ends_per_day*, *spot_more_starts_per_day*, *spot_more_starts_per_day*, etc. Todas se basan en el mismo esquema, el paradigma de programación *MapReduce*, nuestra implementación consiste en aplicar la función *filter* junto con la función *map* y la función *countByKey()* de Python, para posteriormente comparar los elementos del RDD resultante para ver cual es el máximo.

Una vez hemos hecho esto, vemos cuál es el uso de bicicletas para cada rango de edad y por cada tipo de usuario. Para ello observamos el número de veces que cada usuario utiliza el servicio de bicicletas y cuál es su tiempo medio de uso. Esto es relativamente sencillo usando las funciones *countByKey()* y *groupByKey()*. Consideramos que el tiempo medio de uso puede ser relevante puesto que cuanto más tiempo se usa una bicicleta, la empresa obtiene un mayor beneficio.
En caso de que la empresa tenga interés en realizar un análisis anual, hemos implementado una función adicional que devuelve el número de viajes por mes.

4. Análisis de resultados

Para visualizar las estaciones de parada de la empresa BiciMAD tenemos los siguientes planos:

- Plano de las estaciones de BiciMAD
- Plano detallado BiciMAD

Hemos analizado los datos de 2018 y 2017 completos y los del mes de marzo de 2019. Usaremos los datos de marzo para extraer la mayoría de nuestras conclusiones, puesto que es el mes en el que nuestro cliente quiere empezar su campaña, y nos fijaremos en los de 2018 y 2017 para ratificar las tendencias que hemos observado. Además hemos añadido algunas gráficas para visualizar mejor los resultados en las que hemos tenido en cuenta diferentes grupos de edad y una clasificación de los usuarios.
Los grupos de edad son:

- Grupo 0: Indeterminado
- Grupo 1: Niños y adolescentes (0-16 años)
- Grupo 2: Jóvenes sin la mayoría de edad (17-18 años)
- Grupo 3: Jóvenes con mayoría de edad (19-26 años)
- Grupo 4: Adultos de edad menos avanzada (27-40 años)
- Grupo 5: Adultos de edad más avanzada (41-65 años)
- Grupo 6: Jubilados (a partir de los 66 años)

Tipos de usuarios:

- Tipo I: Abonados (tienen un abono anual)
- Tipo II: Ususarios ocasionales (utilizan el servicio de bicicletas esporádicamente)
- Tipo III: Trabajadores de la empresa

Algunos datos relevantes que hemos obtenido son:

4.1. Marzo 2019 - Todos los distritos

Las estaciones más concurridas por día de la semana son Lavapiés (43), Jaime el Conquistador (175) y Paseo de la Esperanza (163), teniendo esta última predominancia en los días más dedicados al ocio. También se observa un considerable aumento del uso de las bicicletas los domingos (de unas 700 en cada una de estas estaciones a alrededor de 1000). Estos podrían ser buenos sitios para poner nuestros anuncios.

Se observa en la Figura 1 que la gran mayoría de los usuarios gozan de una suscripción anual, y que solo un pequeño porcentaje hace viajes casuales o de empresa, pero que el tiempo medio de uso de estos usuarios es el menor. Esto, sin embargo, no es de gran relevancia debido al elevado número de clientes que están suscritos (casi el 93 %). Esto ha de ser tenido en cuenta a la hora de poner el anuncio. La estación más concurrida en total por estos usuarios es Paseo de la Esperanza (163).

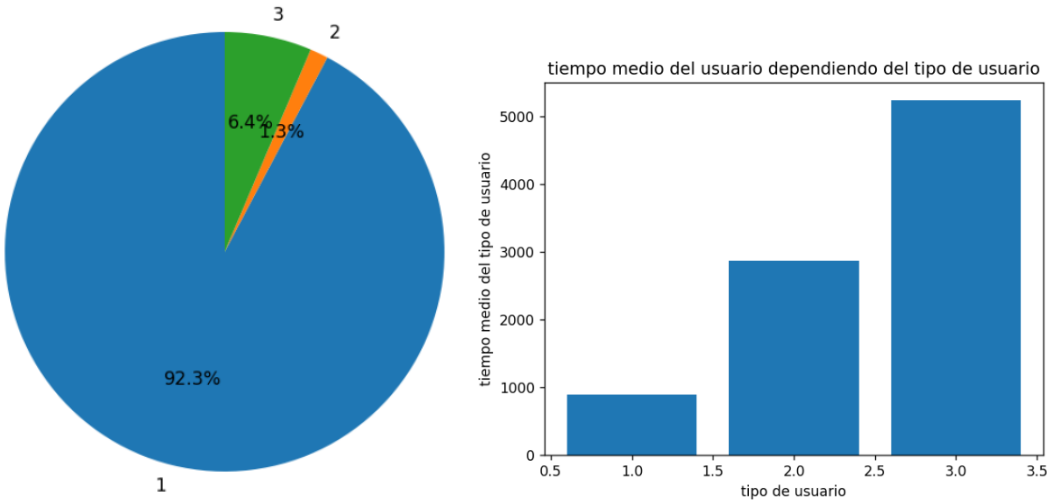

Figura 1: Porcentaje de viajes por tipo de usuario y tiempo medio del usuario dependiendo del tipo de usuario (ocasional, trabajador de la empresa ya bonado, respectivamente)

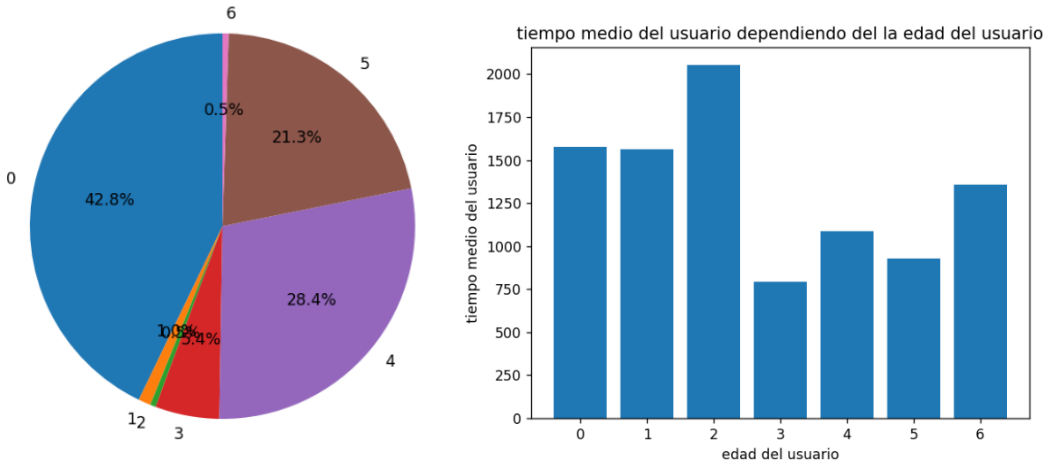

Figura 2: Porcentaje de uso de las biciclestas según el grupo de edad y tiempo medio del usuario dependiendo de la edad del usuario.

La fiabilidad de nuestro análisis sobre los rangos de edad de los usuarios puede ser puesta en tela de juicio, puesto que en el 43 % de los casos no se conoce la edad del usuario. De todas maneras, en

la Figura 2 podemos observar que dentro del 57 % restante, los grupos que más usan las bicicletas son el 4 y el 5 con amplia diferencia, esto es, personas entre 27 y 40 años (28 %) y entre 40 y 63 (21 %). Así, nuestro anuncio probablemente deberá estar enfocado a gente bien entrada en la etapa adulta de su vida. Como es de esperar, los más jóvenes son los que, cuando cogen la bicicleta, viajes más largos hacen. La estación más concurrida por el grupo 4 vuelve a ser la 163, y las más concurridas por el grupo 5 son la 64 (Antonio Maura) y la 90 (Goya). Los jóvenes frecuentan mucho la estación 75 (Menéndez Pelayo).

4.2. Marzo de 2019 - Distrito Retiro

Hemos decidido aplicar el filtro, en nuestro caso, a Retiro, una zona amplia y foco de turistas, pero se podría aplicar a cualquier distrito indicando en el código implementado, los números de estación de ese distrito. En este distrito las estaciones más concurridas son la 64 (Antonio Maura), la 90 (Goya) y la 83 (Pío Baroja). Cabe destacar que los viajes que se inician en este distrito, por lo que parece, no suelen, en su mayoría, salir del distrito. Los datos de edad y tipo de usuario siguen la línea de los globales de Madrid, con los grupos 4 y 5 de edad dominando, y siendo la inmensa mayoría de viajes hechos por usuarios con suscripción anual.

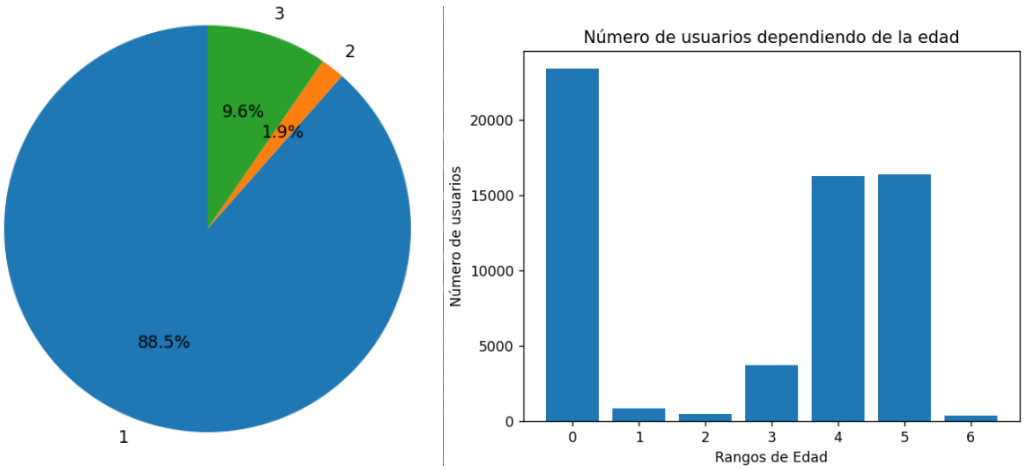

Figura 3: Porcentaje de viajes por tipo de usario y número de usarios dependiendo de la edad (distrito Retiro).

4.3. Año 2018 - Distrito Retiro

Las conclusiones que extrajimos de los datos de marzo de 2019 en cuanto al perfil de los usuarios quedan confirmadas en este análisis anual, que da resultados muy parecidos. En cuanto a las estaciones más concurridas, como es de esperar los datos también son similares, aunque aparecen algunas estaciones nuevas a tener en cuenta como la 76 (Avda Mediterráneo).

Los datos realmente interesantes de este año, al menos potencialmente, para nuestra empresa, son los viajes por mes, que incrementan sustancialmente en otoño, como podemos observar en la Figura 4.

En los histogramas de la Figura 5 observamos que el perfil más común de cliente del servicio de bicicletas es el Tipo I (abonados) y que los usuarios Tipo II (trabajadores de la empresa) aquellos con un tiempo medio más alto por viaje, seguidos de cerca por los abonados. Además el usuario Tipo II (clientes ocasionales) es el que menos utiliza el servivio de bicis. Por lo tanto la publicidad

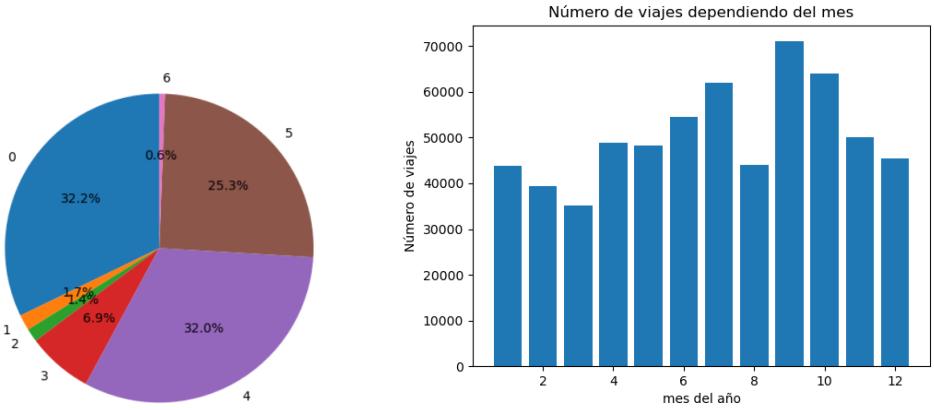

Figura 4: Porcentaje de tiempo medio de viaje según el rango de edad y Número de viajes realizados cada mes (distrito Retiro).

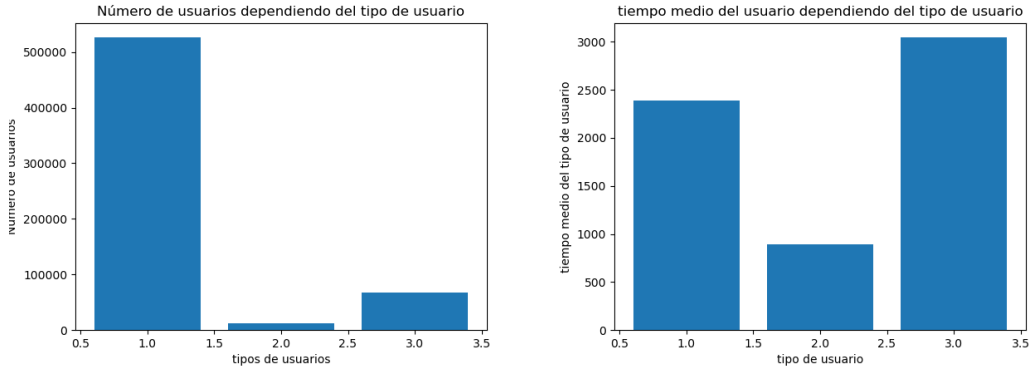

Figura 5: Número de usuarios y tiempo medio dependiendo del tipo de usuario en el año 2018.

4.4. Año 2017 - Distrito Retiro

Hemos analizado los datos del año 2017 desde que se tienen registros (desde el mes de abril). Se observa que el número de viajes en este año es algo menor que en 2018, lo cual es una muy buena señal, pues significa que el uso de las bicicletas tenía una tendencia ascendente antes de la pandemia. Además, se observa un aumento en 2018 respecto de este año en la cantidad de abonados anuales, lo que refuerza nuestra construcción del perfil de usuario. El porcentaje de usuarios según la edad es muy similar a los resultados obtenidos en 2018.
Además, el número de viajes respecto a cada mes, sigue una tendencia similar a la de 2018, teniendo una bajada en el mes de agosoto, que se interpreta como un desplazamiento de los residentes de Madrid a localidades costeras, por el periodo vacacional, teniendo mejores números los meses adayacentes, como consecuencia de una climatología agradable para el uso del servivio de bicletas.

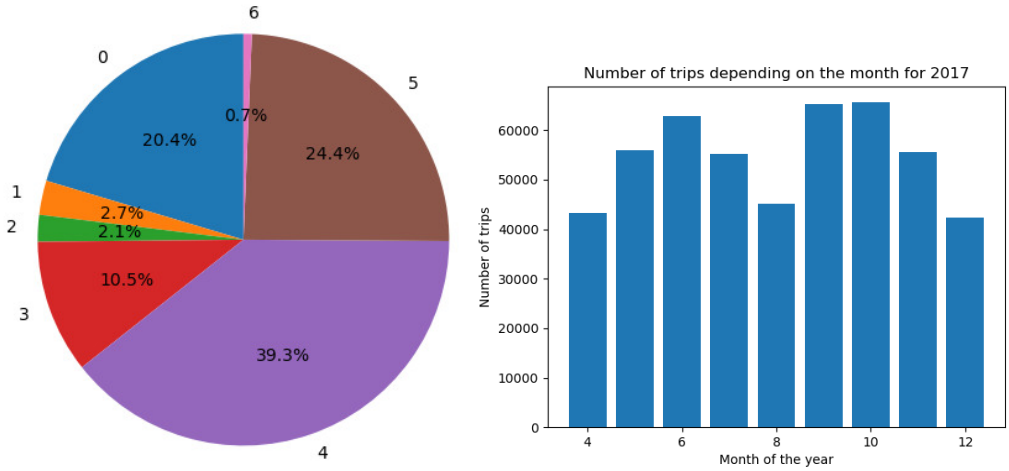

Figura 6: Porcentajes de usuarios según el rango de edad y Evolución temporal del número de viajes.

5. Conclusión

Una estrategia de marketing eficaz consistiría en colocar anuncios publicitarios cerca de las estaciones 175 y 163 y reforzar el número de bicicletas allí, ya que dichas estaciones tenían una fuerte demanda de este servicio antes de la pandemia, y así poder recuperar antiguos clientes. Esos anuncios deberían tener en cuenta que los usuarios más comunes suelen ser personas de mediana edad, siendo la suscripción anual, la opción más atractiva para este tipo de usuarios, para ello la empresa podría ofrecer algún tipo de oferta o plan de financiación para la suscripción anual.

Además, cabe destacar que la duración media de viaje de los jóvenes es significativamente más elevada que para otros grupos, especialmente con más afluencia en la zona circundante a la estación 75. Los anuncios en dicha estación podrían enfocarse en este tipo de clientes.

Finalmente de debe tener en cuenta que en primavera, y especialmente en marzo, el uso de las bicicletas suele ser menor, así que se podría plantear la posibilidad de pasar la campaña a septiembre, el mes con más demada antes de la pandemia, y prolongarla a lo largo del otoño.

6. Extra: Evolución del servicio de bicicletas durante el primer semestre de 2019 - Distrito Retiro

En caso de que la empresa no quiera trasladar la campaña al mes de septiembre, como hemos recomendado, vamos a realizar un estudio del primer semestre de 2019, para corroborar las hipótesis que hemos mantenido los años 2018 y 2017.

En la Figura 7 observamos que el rango de edad con mayor número de clientes es el Grupo 0, el grupo de edad indeterminado, seguido por el Grupo 4 (personas entre 27 y 40 años) con un 26.7 %, siendo este el grupo de edad definido con más usuarios al igual que en los años 2018 y 2017. Análogamente, el tipo de usuario con mayor frecuencia de uso del servicio continúa siendo el Tipo 0 (abonados), corroborando de esta manera las tendencias obtenidas durante los años anteriores.

En la Figura 8, se puede apreciar que los meses con mayor número de viajes son mayo y junio, y no marzo, confirmando la conclusión que hemos sacado en el apartado 6, es decir, el mes de marzo no es el más apropiado para implementar la campaña, obteniendo más beneficios comenzándola en el mes de mayo, coincidiendo con el fin de la primavera.

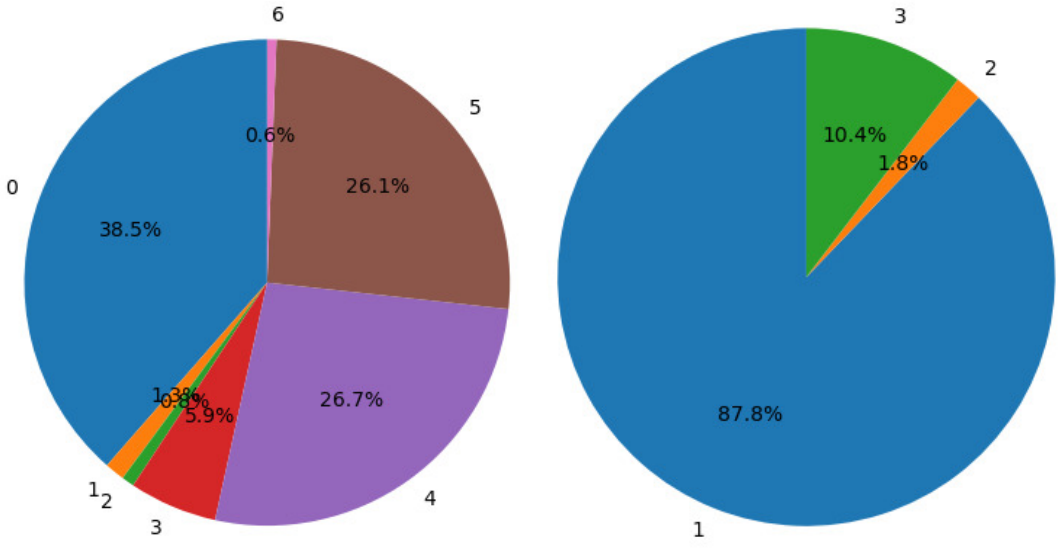

Figura 7: Porcentajes de usuarios según el rango de edad y según el tipo de usario, respectivamente (durante el primer semestre de 2019).

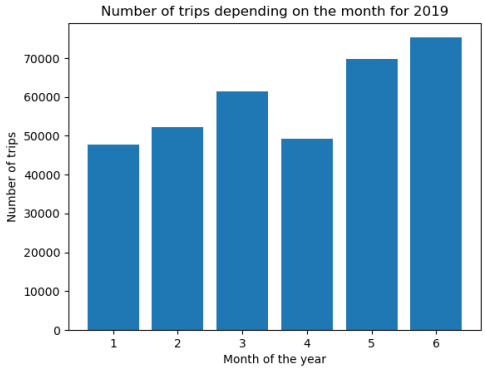

Figura 8: Evolución temporal del número de viajes durante el primer semestre de 2019.

7. Anexo con la implementación del programa en Python

```
1
2  import json
3  from pprint import pprint
4  from pyspark import SparkContext, SparkConf
5  import datetime
6  import sys
7  import os
8  import matplotlib.pyplot as plt
9
10 # Set station, district, and filters (also can be done with sys.argv)
11 YEAR = 2017
12 STATION = 'Spring'
13 DISTRICT = [i for i in range(64,92)]  # Retiro district
14 WANT_TO_FILTER_DISTRICT = True
15 WANT_TO_FILTER_STATION = False
16 YEARLY_ANALYSIS = False
17
18 # Function to extract information from each line
19 def line_info(line):
```

```
    data = json.loads(line)
    ageRange1 = data['ageRange']
    id1 = data['user_day_code']
    start = data['idunplug_station']
    end = data['idplug_station']
    day = data['unplug_hourTime']['$date']
    duration = data["travel_time"]
    user_type = data['user_type']
    return (ageRange1, id1, start, end, date_converter(day), duration, user_type)

# Date conversion
def date_converter(date):
    """Converts the date received from the BiciMad files.
    Returns a tuple with the date in tuple format.
    The tuple contains the weekday, the day of the month, the month, and the year."""

    month = int(date[5:7])
    day = int(date[8:10])
    week_day = datetime.datetime(YEAR, month, day).weekday()
    return (week_day, day, month, YEAR)

def day_converter(week_day):
    """Given a weekday from the datetime module, i.e., a number from 0 to 6,
    converts the number to human-readable format."""

    days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', '
    Sunday']
    return days[week_day]

def season(date_tuple):
    """Given a tuple returned by date_converter, it returns the
    season for that date."""

    (name, day, month, year) = date_tuple
    if month in (4, 5) or (month == 3 and day >=20) or (month == 6 and day <=20):
        return 'Spring'
    elif month in (7, 8) or (month == 6 and day >=21) or (month == 9 and day <=21):
        return 'Summer'
    elif month in (10, 11) or (month == 9 and day >=22) or (month == 12 and day <=20):
        return 'Fall'
    else:
        return 'Winter'

# Functions related to filtering the information prior to analysis

def filter_by_season():
    return WANT_TO_FILTER_STATION

def filter_by_district():
    return WANT_TO_FILTER_DISTRICT

def is_in_district(x):
    return x in DISTRICT

def filter_district(rdd):
    return rdd.filter(lambda x: is_in_district(int(x[2])))

def filter_season(rdd):
    return rdd.filter(lambda x : season(x[4]) == STATION)

# Auxiliary filters for the main functions

# Age filters

def filter_by_age_start(rdd, age):
    """Given an age, returns an RDD where only the trips made by users of that
    age are included, with the start as key. Other functions in this section
    are analogous."""

    return rdd.filter(lambda x : x[0] == age).map(lambda x: (x[2], (x[1], x[3], x[4])
    ))
```

```
89
90 def filter_by_age_end(rdd, age):
91     # Returns the end as key for records of a given age
92     rdd_age = rdd.filter(lambda x : x[0] == age).map(lambda x: (x[3], (x[1], x[2], x
       [4])))
93     return rdd_age
94
95 # Functions to filter by user type
96 def filter_type_start(rdd, type):
97     rdd_type = rdd.filter(lambda x : x[6] == type).map(lambda x: (x[2], (x[1], x[3], x
       [4])))
98     return rdd_type
99
100 def filter_type_end(rdd, type):
101     rdd_type = rdd.filter(lambda x : x[6] == type).map(lambda x: (x[3], (x[1], x[2], x
       [4])))
102     return rdd_type
103
104 # Functions to filter by day of the week
105 def filter_day_start(rdd, day):
106     rdd_day = rdd.filter(lambda x : x[4][0] == day).map(lambda x: (x[2], (x[1], x[3],
       x[4])))
107     return rdd_day
108
109 def filter_day_end(rdd, day):
110     rdd_day = rdd.filter(lambda x : x[4][0] == day).map(lambda x: (x[3], (x[1], x[2],
       x[4])))
111     return rdd_day
112
113 # Function to return the station with the most departures for each day of the week
114 def spot_more_starts_per_day(rdd):
115     rdd_day = [0] * 7
116     for i in range(7):
117         rdd_day[i] = filter_day_end(rdd, i).countByKey()
118         max_station = max(rdd_day[i], key = rdd_day[i].get)
119         print(f'The station with the most departures on {day_converter(i)} is {
       max_station} with {rdd_day[i][max_station]} departures.')
120
121 # Function to return the station with the most arrivals for each day of the week
122 def spot_more_ends_per_day(rdd):
123     rdd_day = [0] * 7
124     for i in range(7):
125         rdd_day[i] = filter_day_start(rdd, i).countByKey()
126         max_station = max(rdd_day[i], key = rdd_day[i].get)
127         print(f'The station with the most arrivals on {day_converter(i)} is {
       max_station} with {rdd_day[i][max_station]} arrivals.')
128
129 # Function to return the station with the most departures for each user type
130 def spot_more_starts_per_type(rdd):
131     rdd_type = [0] * 4
132     for i in range(4):
133         rdd_type[i] = filter_type_start(rdd, i).countByKey()
134         if len(list(rdd_type[i]))>1:
135             max_station = max(rdd_type[i], key = rdd_type[i].get)
136             print(f'The station with the most departures for user type {i} is {
       max_station} with {rdd_type[i][max_station]} departures.')
137         else:
138             print(f'No departures for user type {i}.')
139
140 # Function to return the station with the most arrivals for each user type
141 def spot_more_ends_per_type(rdd):
142     rdd_type = [0] * 4
143     for i in range(4):
144         rdd_type[i] = filter_type_start(rdd, i).countByKey()
145         if len(list(rdd_type[i]))>1:
146             max_est = max(rdd_type[i], key = rdd_type[i].get)
147             print(f'La estaci n en la que llegan m s bicicletas para el tipo de
       usuario {i} es {max_est} y han salido {rdd_type[i][max_est]} bicis')
148         else:
149             print(f'para el tipo de usuario {i} no llegan bicicletas')
150
```

```
151
152 # Similar to the previous but for each age range
153 def spot_more_starts_per_age(rdd):
154     rdd_age = [0] * 7
155     for i in range(7):
156         rdd_age[i] = filter_by_age_start(rdd, i).countByKey()
157         if len(list(rdd_age[i]))>1:
158             max_est = max(rdd_age[i], key = rdd_age[i].get)
159             print(f'The station from which most bikes leave for the age group {i} is {
    max_est} and {rdd_age[i][max_est]} bikes have left')
160         else:
161             print(f'for the age range {i}, no bikes leave')
162
163 def spot_more_ends_per_age(rdd):
164     rdd_age = [0] * 7
165     for i in range(7):
166         rdd_age[i] = filter_by_age_end(rdd, i).countByKey()
167         if len(list(rdd_age[i]))>1:
168             max_est = max(rdd_age[i], key = rdd_age[i].get)
169             print(f'The station where most bikes arrive for the age group {i} is {
    max_est} and {rdd_age[i][max_est]} bikes have arrived')
170         else:
171             print(f'for the age range {i}, no bikes arrive')
172
173 # Function that prints how many trips are made by age
174 def trips_per_age(rdd):
175     rdd_counted_ages = rdd.countByKey()
176     print('The age groups and the bicycles they use are:', '\n')
177     for i in rdd_counted_ages:
178         print(f'The age group {i} used {rdd_counted_ages[i]} bicycles')
179
180     # BAR CHART
181     axes = [[0,1,2,3,4,5,6],[rdd_counted_ages[i] for i in range(7)]]
182     plt.bar(axes[0],axes[1])
183     plt.ylabel('Number of users')
184     plt.xlabel('Age Ranges')
185     plt.title('Number of users depending on age')
186     plt.savefig('tripsagebar',format='png')
187
188     # PIE CHART
189     labels = axes[0]
190     sizes = axes[1]
191     fig1, ax1 = plt.subplots()
192     ax1.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
193     ax1.axis('equal')
194     plt.savefig('tripsagepie',format='png')
195
196 # Function that prints the average time of each trip by age.
197 def time_per_age(rdd_base):
198     rdd_duration= rdd_base.map(lambda x:(x[0],x[5])).groupByKey().map(lambda x : (sum(
    list(x[1]))/len(list(x[1])))).collect()
199     print('The average time that bicycles are used according to the age range is:')
200     for i in range(len(list(rdd_duration))):
201         print(f'The age group {i} used bicycles for an average of {rdd_duration[i]/60}
     minutes')
202
203     # BAR CHART
204     axes = [[0,1,2,3,4,5,6],list(rdd_duration)]
205     plt.bar(axes[0],axes[1])
206     plt.ylabel('average user time')
207     plt.xlabel('user age')
208     plt.title('average user time depending on the user age')
209     plt.savefig('timeage',format='png')
210
211 # Function that prints how many trips are made by user type
212 def trips_per_type(rdd):
213     rdd_counted_type = rdd.map(lambda x: (x[6],(x[0:5]))).countByKey()
214     print('The types of users and the bicycles they use are:', '\n')
215     for i in rdd_counted_type:
216         print(f'the user type {i} used {rdd_counted_type[i]} bicycles')
217
```

```
    # BAR CHART
    axes = [[1,2,3],[rdd_counted_type[i] for i in range(1,4)]]
    plt.bar(axes[0],axes[1])
    plt.ylabel('Number of users')
    plt.xlabel('types of users')
    plt.title('Number of users depending on user type')
    plt.savefig('tripstypesbar',format='png')

    # PIE CHART
    labels = axes[0]
    sizes = axes[1]
    fig1, ax1 = plt.subplots()
    ax1.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
    ax1.axis('equal')
    plt.savefig('tripstypepie',format='png')

# Function that prints the average time of each trip by user type
def time_per_type(rdd_base):
    rdd_duration= rdd_base.map(lambda x:(x[6],x[5])).groupByKey().map(lambda x : (sum(list(x[1]))/len(list(x[1])))).collect()
    print('The average time that bicycles are used according to the user type is:')
    for i in range(len(list(rdd_duration))):
        print(f'The user type {i+1} used bicycles for an average of {rdd_duration[i]/60} minutes')

    # BAR CHART
    axes = [[1,2,3],[rdd_duration[i] for i in range(3)]]
    plt.bar(axes[0],axes[1])
    plt.ylabel('average time of the user type')
    plt.xlabel('user type')
    plt.title('average user time depending on the user type')
    plt.savefig('durationtype',format='png')

# Function that prints the number of trips made per month, in case of annual analysis
def trips_per_month(rdd):
    rdd_trips=rdd.map(lambda x: (x[4][2], (x[1], x[3], x[4]))).countByKey()
    for i in range(1,13):
        print(f'in the month {i} of the year, {rdd_trips[i]} bicycles were used')
    axes = [[i for i in range(1, 13)],[rdd_trips[i] for i in range(1, 13)]]
    plt.bar(axes[0],axes[1])
    plt.ylabel('Number of trips')
    plt.xlabel('Month of the year')
    plt.title('Number of trips depending on the month')
    plt.savefig('tripsyear',format='png')


def main():

    # Initialize Spark and load the JSON files
    conf = SparkConf().setAppName("Routes")

    # Create SparkContext with our configuration
    with SparkContext(conf = conf) as sc:
        sc.setLogLevel("ERROR")  # Set log level to 'ERROR' to avoid cluttering output with log messages

        directory = os.path.abspath('bicimad_data')  # Get the absolute path of the data directory
        rdd_base = sc.emptyRDD()  # Initialize an empty RDD
        i=0  # Counter for files processed
        # Iterate over each file in the directory
        for filename in os.listdir(directory):
            i+=1  # Increment the file counter
            # Process only JSON files
            if filename.endswith(".json"):
                print(f"Processing file: {filename}")  # Inform about the file being processed
                file_rdd = sc.textFile(os.path.join(directory, filename))  # Load the JSON file into an RDD
                rdd_base = rdd_base.union(file_rdd)  # Merge the loaded RDD with the base RDD
```

```
            else:
                pass  # If the file is not a JSON file, ignore it

        # If we processed 12 files, enable yearly analysis
        if i==12:
            YEARLY_ANALYSIS=True

        # Transform the loaded data
        rdd = rdd_base.map(line_info)

        # Apply the district filter, if requested
        if filter_by_district():
            rdd = filter_district(rdd)  # Apply the district filter
            print('\n', 'Analyzing district data', '\n')
        else:
            print('\n', 'Analyzing Madrid data', '\n')  # Inform about the full data analysis

        # Apply the season filter, if requested
        if filter_by_season():
            print('The data will be filtered by season')
            rdd = filter_season(rdd)  # Apply the season filter
        else:
            print('\n', 'The data will not be filtered by season')  # Inform about not applying season filter

        # Start the analysis
        print('\n', 'Statistics of bike usage by day', '\n')
        spot_more_starts_per_day(rdd)  # Determine the spots with the most starts per day
        print('\n')
        spot_more_ends_per_day(rdd)  # Determine the spots with the most ends per day

        print('\n', 'Statistics of bike usage by user type', '\n')
        print('The types of users are:', '\n')
        print('0: User type could not be determined', '\n')
        print('1: Annual user (possessor of an annual pass)', '\n')
        print('2: Occasional user', '\n')
        print('3: Company worker', '\n')

        spot_more_starts_per_type(rdd)  # Determine the spots with the most starts per user type
        print('\n')
        spot_more_ends_per_type(rdd)  # Determine the spots with the most ends per user type

        print('\n', 'Statistics of bike usage by age groups', '\n')
        print('The age groups are:', '\n')
        print('0: User age group could not be determined' , '\n')
        print ('1: User is between 0 and 16 years old' , '\n')
        print ('2: User is between 17 and 18 years old', '\n')
        print ('3: User is between 19 and 26 years old' , '\n')
        print ('4: User is between 27 and 40 years old' , '\n')
        print('5: User is between 41 and 65 years old' , '\n')
        print('6: User is 66 years old or more' , '\n')

        spot_more_starts_per_age(rdd)  # Determine the spots with the most starts per age group
        print('\n')
        spot_more_ends_per_age(rdd)  # Determine the spots with the most ends per age group

        print('\n', 'Usage according to age', '\n')
        trips_per_age(rdd)  # Determine the number of trips per age group
        time_per_age(rdd)  # Determine the duration of trips per age group

        print('\n', 'Statistics of bike usage by user type', '\n')
        trips_per_type(rdd)  # Determine the number of trips per user type
        time_per_type(rdd)  # Determine the duration of trips per user type

        # If yearly analysis is enabled, perform it
```

```
        if YEARLY_ANALYSIS:
            print('Yearly user evolution')
            trips_per_month(rdd)  # Determine the number of trips per month
    sc.stop()  # Stop the SparkContext to free resources

if __name__ == "__main__":

    # Interact with the user for changing initial filters, if arguments provided.
    if len(sys.argv) > 2:
        if isinstance(sys.argv[1], bool):
            if sys.argv[1]:
                WANT_TO_FILTER_STATION=True
            else:
                WANT_TO_FILTER_STATION=False

        if isinstance(sys.argv[2], str):
            STATION=sys.argv[2]

        if isinstance(sys.argv[3], bool):
            if sys.argv[3]:
                WANT_TO_FILTER_DISTRICT=True
            else:
                WANT_TO_FILTER_DISTRICT=False

        if isinstance(sys.argv[4], list):
            DISTRICT=sys.argv[4]

    main()  # Call the main function
```