Big Data Management
Assignment 1

Description

In this assignment your task is to prepare the batch layer (off-line processing pipeline) of the lambda architecture that will enable us to perform some analytics on a dataset. You are required to use **Apache Spark's DataFrame** to perform the required computation.

Dataset

Transport Infrastructure Ireland (TII) (`https://www.tii.ie`) operates and maintains a network of traffic counters on the motorway, national primary and secondary road networks in Ireland. These traffic counters capture data on different parameters. There are currently around 400 of these counters active across the network. For an interactive view of the data they capture, go to the TII Traffic Counter Data Website: (`https://www.nratrafficdata.ie`). On this website, green dots display the individual traffic counter locations around the country. Summary traffic data information can be obtained from each site by clicking on the green dot. Upon clicking a dot, a pop-up window will appear that summarises the traffic data and provides a link to more detailed data. You can click on the **list of sites** button to view description of each counter.

The traffic counter data set is a valuable source of information on vehicle movements across the national road network and is made available publicly in its raw form, in order to provide researchers, public bodies, engineering companies, as well as the general public, with the opportunity to analyse and query the data independently for their own specific purposes. The first row of each file contains headers which describe each field. However, the meaning of some of these may not be apparent to consumers. The following explains some of the less obvious column headers:

cosit: The unique identifier for the traffic counter device. In conjunction with the site's dataset, this can be used to determine the location and route of the counter, used to record the vehicle movement.

lane: The Id of the lane in which the movement was recorded, which is specific to each counter.

straddlelane: If a value is present, this indicates that the vehicle may have been changing lanes as it passed over the counter.

class/classname: This indicates the category of vehicle that was recorded e.g. car, bus, etc.

length: The approximate length of the vehicle recorded.

headway: The approximate distance between the front of the recorded vehicle and the vehicle behind.

gap: The approximate distance between the rear of the vehicle and the front of the vehicle behind.

weight: This is available on (Weigh-in-Motion) WIM sites only and indicates the approximate weight of the vehicle.

temperature: If available, this indicates the approximate surface temperature of the road at the location of the device.

numberofaxles: This is available on WIM sites only and indicates the number of axles detected for the vehicle.

axleweights: This is available on WIM sites only and expresses as an array of real numbers, the weight over each axel in order.

axlespacing: This is available on WIM sites only and expresses as an array of real numbers, the distance between each of the axles.

Downloading the Data

The data is updated and published daily, one day in arrears. The traffic counters on average generate over 4 million records for each day. The aggregated data from all traffic counters is a very large multi-terabyte data set. Without doubt this is a big data management problem. However, for this assignment, we will only be using data for the 31st January, 2020. Download the (31st January 2020) data using the following link:

https://data.tii.ie/Datasets/TrafficCountData/2020/01/31/per-vehicle-records-2020-01-31.csv

To view the content of the downloaded file on your terminal you can use the less command:

```
less per-vehicle-records-2020-01-31.csv

// To exit 'less' press q
```

Setup

In order to complete this work and the future assignments you need to have a working VM. The VM must have the Hadoop installed in Pseudo-Distributed Operation (if you 're storing the data in HDFS). It must have the Apache Spark setup and Jupyter Notebook linked with PySpark. If your VM has everything ready - go to the next section. Otherwise follow the guides provided on Moodle on how to setup Ubuntu VM, Hadoop, and Apache Spark.

Spark DataFrame

Spark SQL is a Spark module for structured data processing. Spark SQL module has a number of classes which allow you to perform computation on the data. Among those, DataFrame is an important class of the Spark SQL module. A DataFrame is a distributed collection of data grouped into named columns. DataFrame is a higher level abstraction on top of RDDs that allows you to perform operations in most efficient ways (when compared to RDDs) as well as use a query language (e.g., SQL) to manipulate the data. A DataFrame is equivalent to a relational table in Spark SQL. This assignment requires you to learn about the DataFrame class and the functions it provides. You can find a list of functions with examples at the following link: `https://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.DataFrame`.

Loading Data in Spark DataFrame

We can use **spark.read.csv** function which loads a CSV file and returns the result as a DataFrame. This function will go through the input once to determine the input schema if inferSchema is enabled. To avoid going through the entire data once, disable inferSchema option or specify the schema explicitly using schema. For more details see: (`https://spark.apache.org/docs/latest/api/python/pyspark.sql.html`). For example, in the line below we are using the inferSchema option. You can also try specifying schema explicitly using the schema option.

```
vehicle_counter_DF = spark.read.csv('/home/bdm/per-vehicle-records-2020-01-31.csv',
                                     inferSchema = True, header = True)
```

- You can check the type of `vehicle_counter_DF`:

```
type(vehicle_counter_DF)
# output: pyspark.sql.dataframe.DataFrame
```

- Count the number of lines/records in `vehicle_counter_DF`:

```
vehicle_counter_DF.count()
# output: 4740861
```

- Take a look at the first 5 lines in `vehicle_counter_DF`:

```
vehicle_counter_DF.show(5)
# output: 5 lines from the vehicle_counter_DF
```

- As each line has comma-separated values, you can split it by comma:

```
vehicle_counter_DF.printSchema
# output: the schema of the vehicle_counter_DF dataframe
```

Questions

Your task is to prepare code (using the DataFrame class) that could run in the batch layer, and will help performing the following analyses:

1. Calculate the usage of Irish road network in terms of percentage grouped by vehicle category.
2. Calculate the highest and lowest hourly flows on M50 - show the hours and total number of vehicle counts.
3. Calculate the evening and morning rush hours on M50 - show the hours and the total counts.
4. Calculate average speed between each junction on M50 (e.g., junction 1 - junction2, junction 2 - junction 3, etc.).
5. Calculate the top 10 locations with highest number of counts of HGVs (class). Map the COSITs with their names given on the map.

Submission

- Create a Python notebook - name it `assignment1.ipynb`. The notebook should be exported as iPython Notebook with *.ipynb extension. Make sure you have run the entire code and then exported the notebook. If the code in your notebook does not run, it will result in 20% penalty.
- Take one screenshot of your solution to each question (show code + its output) and put it in a document, generate a pdf of this document.
- Zip both files (pdf and .ipynb) together and submit your solution on Moodle. There is 10% penalty for incorrect submission.
- Do not submit work that's not your own and do not let others copy work that is your own. Both Copyier and Copyee will get ZERO marks.