Projet II (2023-24)

Technologies Big Data. M1 MIDS/Informatique

Vlady Ravelomanana and Stéphane Boucheron

2024-05-29

Due date : May, the 29th, 2024

NY Shared Bikes

This homework is about New York Shared Bike systems (Citibikes). The shared bikes system has been running for more than 10 years.

Citibikes has released a detailed historical dataset covering over 10 years from 2013 up to now. Taken as a whole, the detailed trip-level data is more than just a vast list of rides pick-up and dock-off coordinates: it's a story of a bike comuters in the big Apple (NYC).

> Where do Citi Bikers ride? When do they ride? How far do they go? Which stations are most popular? What days of the week are most rides taken on?
>
> How seasonal is the bike traffic? How does it vary though time and space?
>
> Where does the crowd hang out on Sunday afternoon?
>
> Who bikes during winter?
>
> What time do investment bankers ride to work? How has electrical assistance changed the landscape for bike shaing systems? The dataset addresses all of these questions and many more.

Note

Information can be gathered at the following URLs:

- https://bikeshare-research.org
- https://citibikenyc.com

- Real time feed
- NYC Bike feeds data
- Station status feed
- Bike routes

Tip

The dataset is available from on Amazon S3 (Amazon's cloud storage service).

- https://citibikenyc.com/system-data
- https://api.citybik.es/citi-bike-nyc.json
- https://s3.amazonaws.com/tripdata/index.html

Important

For this homework **we will let you decide on the tools to use** (expected: Spark/Dask) and to **find out information all by yourself**.

Download the zip archives

Download (programatically) the trip data for years 2014 to 2023. Your downloading process should be reproducible.
There are several CSV files for each calendar month, or for each calendar year.
Each yearly file is moderately large, from a few hundreeds of megabytes up to a gigabyte.
The full dataset is moderately large if it has to be handled on a laptop (compressed yearly CSV files make 8 gigabytes).
Recall that whatever the framework you use (Pandas, Dask or Spark), `CSV` files prove hard to handle.

After downloading the files (this takes time, but this is routine), a first step will consist in converting the csv files into a more Spark friendly format such as `parquet` or `orc`.

Saving into one of those formats require decisions about bucketing, partitioning and so on. Such decisions influence performance. It is your call.

Saving the data into a big data format (parquet/ORC)

Parquet (and ORC) allow a *limited amount of schema evolution*. You should take advantage of this. This requires a substantial amount of work:

- renaming columns so as to obtain a consistent naming scheme
- taking care of inconsistencies in timestamp formatting (not always ISO compliant)

- taking care of varying ways of identifying stations (different station ID formats are used throughout the ten years).
- taking of the fact that some columns disappear at some point while others appear.

At the end of the process, you should have one partitionned parquet or ORC file.

Organize the data into a star schema

From the trip data, you should build a *dimension table* gathering information concerning stations (name, id(s), latitude, longitude, ...) and an *event table* gathering information on trips with references to the dimension table (the start/end station ids).

Tip

Hint. Don't forget to ask `Spark/Dask` to use a substantial share of the memory and ressources from your computer.
Hint. Don't forget that you should specify a partitioning column and a number of partitions when creating the parquet files.

Important

When working on this, ask you and answer to the following questions:

1. What is the `StorageLevel` of the dataframe after reading the csv files?
2. What is the number of partitions of the dataframe?
3. Is it possible to tune this number at loading time?
4. Why would we want to modify the number of partitions when creating the parquet files?

Investigate (at least) one year of data before covid and one year after covid

Compute at least the following and produce relevant plots:

1. Break down the trip distance distribution for each day of week
2. Count the number of trips for each pickup/dropoff location couple
3. Compute trip distance distribution for gender (when available)
4. Compute trip distance distribution for age ranges (`15-24`,`25-44`, `45-54`, `55-64`, `65+`) (when available)

5. Compute trip distance distribution for different kind of bikes (when available)

Tip

If you intend to build appealing graphics, you may stick to `matplotlib + seaborn`, you can use also `plotly` or `altair`, which are used to build interactive graphics, but you can use whatever you want.

Assessing seasonalities and looking at time series

Compute and plot the following time series indexed by day of the week and hour of day:

1. The number of pickups/docks
2. The average distance
3. The average trip duration
4. The average number of ongoing trips

Monitor job execution

1. Use the `explain` method or have a look at the Spark UI to analyze the job. You should be able to assess
   - Parsed Logical Plan
   - Analyzed Logical Plan
   - Optimized Logical Plan
   - Physical Plan
2. Do the Analyzed Logical Plan and Optimized Logical Plan differ? Spot the differences if any. How would a RDBMS proceed with such a query?
3. How does the physical plan differ from the Optimized Logical Plan? What are the keywords you would not expects in a RDBMS? What is their meaning?
4. Inspect the stages on Spark UI. How many *stages* are necessary to complete the Spark job? What are the roles of `HashAggregate` and `Exchange hashpartitioning`?
5. Does the physical plan perform `shuffle` operations? If yes how many?
6. What are tasks with respect to stages (in Spark language)? How many tasks are your stages made of?

Dive into spatial information problems

> **i** Geographic information
>
> Pick a year
> For this, you will need to find tools to display maps and to build choropeth maps. We let you look and find relevant tools to do this.
>
> 1. Build a heatmap indexed by couple of stations where color is a function of number of trips from one station to another
>
> 2. Build an interactive heatmap with a slider allowing the user to select an `hour of day` and where the color is a function of number of trips from one station to another

You **might** need the following stuff in order to work with GPS coordinates and to plot things easily.

```
!pip install geojson geopandas plotly geopy
```

```
!pip install ipyleaflet
```