# Big Data Processing Systems — 2020/21
Project nº 2

## Introduction

In just a few words, in this project you are asked to solve **a subset of the questions/exercises** you were asked in Project 1, but now with different technologies, namely: pySpark Dataframes, pySpark SQL, and Hive HQL. You may also implement exactly the same optional exercises you did in Porject 1.

The *pySpark dataframes* and *pySpark SQL* exercises are to be developed in the same docker container used for the first project. The *hive* exercises are to be developed in a different docker container. Please download and run it as follows:

1) download this docker image

```
docker pull prasanthj/docker-hive-on-tez
```

2) and then run it with

```
docker run -v path_to_local_folder:/root/work --name hive -it -P
prasanthj/docker-hive-on-tez /etc/hive-bootstrap.sh -bash
```

3) please note that "**path_to_local_folder**" in 2) MUST BE replaced with a path to a folder in your computer (Windows/Linux/macOS). This folder will be shared between your host computer and the docker image. This means you can use your favourite text editor to edit your HQL scripts, and then switch to the docker container to run/text them.

## The Data Set

Consider you want to process information about taxi rides in some city to better understand the behavior of the community and help taxi drivers maximize their profit.

To this end, you have a data set with information about taxi drivers, including information for pickup and drop-off location.

The data set has the following columns:

| Col# | Column Name | Type | Description |
|---|---|---|---|
| **1** | Trip ID | Plain Text | A unique identifier for the trip. |
| **2** | Taxi ID | Plain Text | A unique identifier for the taxi. |
| **3** | Trip Start Timestamp | Date & Time | When the trip started, rounded to the nearest 15 minutes. |
| **4** | Trip End Timestamp | Date & Time | When the trip ended, rounded to the nearest 15 minutes. |
| **5** | Trip Seconds | Number | Time of the trip in seconds. |
| **6** | Trip Miles | Number | Distance of the trip in miles. |
| **7** | Pickup Region ID | Plain Text | The City Region where the trip began. For privacy, this field is not shown for some trips. |
| **8** | Dropoff Region ID | Plain Text | The City Region where the trip ended. For privacy, this field is not shown for some trips. |
| **9** | Pickup Community Area | Number | The Community Area where the trip began. For privacy, this field is not shown for some trips. |
| **10** | Dropoff Community Area | Number | The Community Area where the trip ended. For privacy, this field is not shown for some trips. |
| **11** | Fare | Number | The fare for the trip. |

| | | | |
|---|---|---|---|
| **12** | Tips | Number | The tip for the trip. |
| **13** | Tolls | Number | The tolls for the trip. |
| **14** | Extras | Number | Extra charges for the trip. |
| **15** | Trip Total | Number | Total cost of the trip, the total of the previous columns. |
| **16** | Payment Type | Plain Text | Type of payment for the trip. |
| **17** | Company | Plain Text | The taxi company. |
| **18** | Pickup Centroid Latitude | Number | The latitude of the center of the pickup city region or the community area if the city region has been omitted. For privacy, this field is not shown for some trips. |
| **19** | Pickup Centroid Longitude | Number | The longitude of the center of the pickup city region or the community area if the city region has been omitted. For privacy, this field is not shown for some trips. |
| **20** | Pickup Centroid Location | Point | The location of the center of the pickup city region or the community area if the city region has been omitted. For privacy, this field is not shown for some trips. |
| **21** | Dropoff Centroid Latitude | Number | The latitude of the center of the dropoff city region or the community area if the city region has been omitted. For privacy, this field is not shown for some trips. |
| **22** | Dropoff Centroid Longitude | Number | The longitude of the center of the dropoff city region or the community area if the city region has been omitted. For privacy, this field is not shown for some trips. |
| **23** | Dropoff Centroid Location | Point | The location of the center of the dropoff city region or the community area if the city region has been omitted. For privacy, this field is not shown for some trips. |

Please note that numbers, when larger than 999, have a coma separating the thousands, e.g., "1,345.98". Take this into consideration when converting the string to a number.

## Downloading the Data Set

There are 3 versions of the data set, with different sizes:

- Taxi_Trips_151MB.csv (small — 151 Mbytes)
- Taxi_Trips_8GB.csv (medium — 8 Gbytes)
- Taxi_Trips_80GB.csv (large — 80 Gbytes)

They are available at the address:

https://tinyurl.com/yxj6m5yj

## pySpark Dataframes

You are asked to use *pySpark Dataframes* to create indexes for answering the following queries:

1. (2 points) To have a rough understanding of the fluctuation in the demand for taxis along the years, create and index reporting

   *How many trips were started in each year present in the data set?*

Output is expected to have two columns: *(year, #trips)*. In the example below the values are not real and you may get different ones.

| | |
|---|---|
| *2013* | *54745* |
| *2014* | *71652* |
| ••• | |
| *2019* | *43690* |
| *2020* | *6654* |

2. (3 points) To have a rough understanding of the fluctuation in the demand for taxis per hour, create and index reporting

   *For each of the 24 hours of the day, how many taxi trips there were, what was their average trip miles and trip total cost?*
   *Non-integer values should be printed with two decimal places.*

   Output is expected to have four columns (time_slot, #trips, avg_trip_miles, avg_trip_total). In the example below the values are not real and you may get different ones.

| | | | |
|---|---|---|---|
| *01 AM* | *11165* | *2.46* | *13.74* |
| *01 PM* | *20179* | *3.38* | *15.79* |
| *02 AM* | *8832* | *2.35* | *12.71* |
| *02 PM* | *20039* | *3.56* | *16.10* |
| *03 AM* | *6594* | *2.48* | *12.96* |
| ••• | | | |
| *11 PM* | *16303* | *3.07* | *15.00* |
| *12 AM* | *13543* | *2.83* | *14.45* |
| *12 PM* | *19874* | *3.38* | *16.04* |

3. (4 points) To have a rough estimation of the most popular locations for picking up clients at each moment, create and index reporting

   *For each of the 24 hours of the day, which are the (up to) 5 most popular routes (pairs pickup/dropoff regions) according to the the total number of taxi trips? Also report and the average fare (total trip cost).*
   *Non-integer values should be printed with two decimal places.*

   Output is expected to have four columns (pickup_location, #trips, avg_trip_miles, avg_trip_total). In the example below the values are not real and you may get different ones.

| | | |
|---|---|---|
| *04 PM 17031839100 17031839100* | *478* | *6.30* |
| *05 PM 17031839100 17031839100* | *466* | *6.66* |
| *11 AM 17031839100 17031839100* | *447* | *6.98* |
| ••• | | |
| *05 PM 17031839100 17031320100* | *292* | *7.20* |
| *07 PM 17031839100 17031839100* | *285* | *6.49* |

4. (1 to 3 points) An **optional 4th index** that will answer a non-trivial and interesting question over the given data set.

## pySpark SQL and Hive HQL Exercises

You are asked to use both *pySpark* SQL and *hive HQL* to create indexes for answering the following queries:

1. (2 points) To have a rough understanding of the fluctuation in the demand for taxis per month of the year, create and index reporting

   *What is the accumulated number of taxi trips per month?*

   Output is expected to have two columns: *(month_number, #total_trips)*. In the example below the values are not real and you may get different ones.

| | |
|---|---|
| 03 | 39260 |
| 06 | 37016 |
| ••• | |
| 01 | 30357 |
| 11 | 30017 |
| 12 | 29252 |

2. (3 points) To have a rough understanding of the taxi service, create and index reporting

   *For each pickup region, report the list of unique dropoff regions?*

   Output is expected to have two columns: *(pickup_region_ID, list_of_dropoff_region_ID)*. In the example below the values are not real and you should get different ones (the output is nicely wrapped in this text to facilitate reading).

| | |
|---|---|
| 17031030400 | [17031040402, 17031030400, 17031243500] |
| 17031030400 | [17031040402, 17031030400, 17031243500] |
| 17031830900 | [17031080201, 17031980000, 17031320600, 17031081700, 17031030101, 17031240200, 17031242100, 17031063100, 17031241500, 17031081500] |
| 17031070700 | [17031063301, 17031241400, 17031061500] |
| 17031061200 | [17031062100, 17031062800, 17031061800, 17031031300, 17031032100, 17031061200, 17031040402, 17031063000, 17031081600, 17031242300, 17031070300, 17031831000, 17031060900, 17031831900, 17031062700] |
| 17031210601 | [17031063302, 17031061902] |
| 17031243200 | [17031071000, 17031071500, 17031831100] |
| 17031060200 | [17031281900, 17031081800] |
| ••• | |
| 17031030603 | [17031030603] |
| 17031060300 | [17031832100, 17031060500, 17031031400, 17031832500] |
| 17031410900 | [17031980000, 17031410900] |
| 17031390700 | [17031080100, 17031841100] |
| 17031242500 | [17031242500, 17031242000] |
| 17031031900 | [17031031900, 17031070300] |
| 17031243000 | [17031240500, 17031061500] |

3. (4 points) To have a rough understanding of the total price of each taxi ride (based in the pickup and drop off regions), create and index reporting

   *What is the expected charge/cost of a taxi ride, given the pickup region ID, the weekday (0=Monday, 6=Sunday) and time in format "hour AM/PM"?*

   Output is expected to have two columns: *(month_number, avg_total_trip_cost)*. In the example below the values are not real and you should get different ones.

| | |
|---|---|
| 17031221300_5_12AM | 8.40 |
| 17031980000_2_07PM | 67.40 |
| 17031980000_3_06PM | 52.37 |
| 17031081201_4_05AM | 1.20 |
| 17031280100_0_02PM | 2.85 |
| 17031243400_5_09PM | 3.00 |
| 17031839100_0_08AM | 5.87 |

| | |
|---|---|
| *17031081401_2_10AM* | *4.80* |
| *17031081500_0_04PM* | *4.20* |
| *17031081500_6_01PM* | *5.26* |
| *17031081600_3_05PM* | *8.53* |
| *17031081403_3_01PM* | *4.20* |
| *17031281900_3_12PM* | *2.60* |
| ••• | |

4. (1 to 3 points) An **optional 4th index** (different from the one proposed for map-reduce) that will answer a non-trivial and interesting question over the given data set.

## Project Development

Each project team must have two members.

Please submit (two) Jupiter Notebooks for the *pySpark Dataframes* and *pySpark SQL* exercises.

Please submit 3 (or 4) text files with the *hive HQL* queries.

Please use the small data samples to develop and validate your solution to the proposed exercises.

## Project Submission

Please follow the instructions below to submit your project.

1. Prepare one Jupiter Notebook with the solution for the *pySpark Dataframes* exercises with name “`pySpark-dataframes.ipynb`”.
2. Prepare one Jupiter Notebook with the solution for the *pySpark SQL* exercises with name “`pySpark-SQL.ipynb`”.
3. Prepare 3 or 4 text files with the queries for using *hive HQL*, named “`hive-1.hql`”, “`hive-2.hql`”, etc.
4. Prepare a project report in PDF format with name “`report.pdf`”, covering the exercises in the three technologies, with the look and feel of a research paper, with a maximum of 4 pages (optional work, acknowledgments and references may go on a 5th page). It is mandatory to follow the IEEE template for Computer Society Journals (`https://goo.gl/Xtjdh4`) using either Word or LaTeX. Please remember to show, for each exercise, a small sample of the produced index and relate the corresponding execution times with the ones you obtained in the first project (assuming you will use the same computer for both projects).

Submission instructions:

1. Create a folder named **`Gnn_AAAAA_BBBBB_hw2`**, where

| | |
|---|---|
| **`Gnn`** | —> group number, e.g., G04 |
| **`AAAAA`** | —> student 1 number, e.g., 45454 |
| **`BBBBB`** | —> student 2 number, e.g., 54321 |

   (the numbers `AAAAA` and `BBBBB` must be in increasing order).

2. Copy your Jupiter Notebook for the *pySpark Dataframes* exercises (“`pySpark-dataframes.ipynb`”) to the above folder (“`Gnn_AAAAA_BBBBB_hw2`”).
3. Copy your Jupiter Notebook for the *pySpark SQL* exercises (“`pySpark-SQL.ipynb`”) to the above folder (“`Gnn_AAAAA_BBBBB_hw2`”).
4. Copy the *hive HQL* queries “`hive-1.hql`”, “`hive-2.hql`”, etc, to the above folder (“`Gnn_AAAAA_BBBBB_hw2`”).
5. Copy your project report (“`report.pdf`”) to the above folder (“`Gnn_AAAAA_BBBBB_hw2`”).

6. Zip the folder and its contents into a file named "**`Gnn_AAAAA_BBBBB_hw2.zip`**".
7. Submit your ZIP file in the form at the address:
   https://forms.gle/X8qUjMha5Tn1P1T99
   no later than **Monday, December 7, 2020 @ 23:59**.