**Application Big Data**

**Étapes 1et 2**

**Préparé et présenté par:**

Carlos, CAVALCANTE, 1795417

Kelogue THÉRASMÉ, 1795487

**À la professeure:**

Nesrine Zermili

**Dans le cadre du cours:**

Exploitation d'une plateforme de données massives

420-T16-BB gr.00226

Hiver 2018

Montréal, 29 Mars 2018

## Le mandat

**Application Big Data**

**Étape 1**

## Objectifs :

* Définir une application en Big Data selon un contexte économique et l’analyse des besoins.

## Travail attendu :

Sur la base de vos recherches et expertises d’un domaine applicatif, vous devez :

* Choisir un domaine d’application ou secteur économique.
* Identifier les cas d’utilisation possible pour un projet Big Data
* Identifier les sources de données pour aider à définir quelles technologies et capacités de Big Data sont nécessaires pour résoudre la problématique commerciale.
* Décider quelles sont les données à inclure et ce qu’il faut laisser de côté. Identifier uniquement les données stratégiques qui aboutiront à des informations significatives.

## 

## Plan du document

[Mandat 1](#_Toc21941)

[Plan du document 3](#_Toc17483)

[1. Domaine d’application  4](#_Toc13390)

[2. Objectif 4](#_Toc11903)

[3. Questions d’analyse ou problèmes à résoudre 4](#_Toc19379)

[4. Cas d’utilisation d’objets connectés 4](#_Toc20847)

[5. Données nécessaires 5](#_Toc26150)

[6. Les sources de données mobilisées 5](#_Toc15468)

[7. Données pertinentes 7](#_Toc10693)

[8. Techniques d’analyse 8](#_Toc18668)

[a) Collisions ou accidents 8](#_Toc20178)

[b) Crimes 8](#_Toc2109)

[9. Mise en œuvre, scripts et résultats  8](#_Toc23316)

[9.1 Les collisions avec Pig 8](#_Toc1017)

[9.2 Les crimes avec Hive et MapReduce 13](#_Toc2305)

[Les contraventions avec Hive et Pig 17](#_Toc3827)

## Domaine d’application

Ce projet s’inscrit dans la dynamique de développement d’application visant le transport intelligent.

## Objectif

Le travail consiste à exploiter la base de données ouvertes de la Ville de New-York pour concevoir une application aidant les citoyens et les citoyennes à prévenir les risques liés au déplacement urbain. Nous parlons dans le cadre de ce travail des risques pour les piétons et les automobilistes.

## Questions d’analyse ou problèmes à résoudre

Dans cette perspective, nous formulons deux questions d’analyse de départ :

* Quel est le risque de recevoir un billet de contravention considérant la localisation et les caractéristiques du véhicule ?
* Quel est le risque d’avoir un accident, d’être victime d’un crime selon la position géographique dans la ville?

Parmi les problèmes réels que cette application peut résoudre est d’aider:

* les conducteurs et les piétons à éviter des accidents en les alertant lorsqu’ils s’approchent d’une zone dangereuse, c’est-à-dire oû l’on a déjà enregistré un nombre relativement élevé d’accidents.
* Les conducteurs à éviter de se faire voler ou d’être victimes de certaines crimes en les alertant à la vigilance lorsqu’ils se déplacent ou immobilisent leur véhicule dans certains endroits.

Un transport intelligent, c’est moins d’accidents liés au déplacement et moins de crimes et de billets de contravention.

## Cas d’utilisation d’objets connectés

Un cas d’utilisation possible est calculer en temps réel le niveau du risque et d’alerter la personne sur la prudence nécessaire. Ce projet peut s’implémenter soit sous la forme d’une application mobile pour téléphone intelligente ou d’un objet connecté doté de capacité de géolocalisation, d’un microprocesseur et de dispositif d’alerte sonore ou visible.

|  |  |
| --- | --- |
| Option 1 | Option 2 |
| * Voiture dotée de GPS (capteur) * Microprocesseur * Écran d’affichage et émetteur de bip (actionneur). | * Un téléphone intelligent |

## Données nécessaires

* Trois thématiques stratégiques:

Nous avons donc colligé des données relatives aux trois thématiques suivantes:

* + Les contraventions,
  + Les crimes,
  + Et les accidents impliquant un véhicule ou plusieurs.
* Une portée géographique urbaine:

Le projet exploite les données ouvertes de la Ville de New-York (NYC).

* Une portée temporelle large et diverse

Ce projet prend en considération des données

* + Historiques - à partir de la base de données ouvertes, nous avons colligé des données de 2005 à date en lien à notre problématique dont les crimes, les contraventions et les accidents
  + En temps réel – la position géographique courante de l’utilisateur utilisant l’application.
  + Non temporelles, plutôt attributaires comme par exemple les caractéristiques des véhicules, indépendantes du temps.
  + Temps réel : positionnement géographique du véhicule.

## Les sources de données mobilisées

Le portail des données ouvertes de la ville de New York met à disposition du public trois datasets pertinents pour ce projet:

* **Dataset sur les crimes entre 01/01/2006 et 12/31/2015**

This dataset includes all valid felony, misdemeanor, and violation crimes reported to the New York City Police Department (NYPD) from 2006 to the end of last year (2015). For additional details, please see the attached data dictionary in the ‘About’ section.

<https://data.cityofnewyork.us/Public-Safety/2014-to-2015-historic-data/2799-jedt>

* **NYC Parking Tickets**

Rows of Parking Ticket Data, Aug 2013-June 2017

There are four files, covering Aug 2013-June 2017. The files are roughly organized by fiscal year (July 1 - June 30) with the exception of the initial dataset. The initial dataset also lacks 8 columns that are included in the other three datasets (although be warned that these additional data columns are used sparingly). See the dataset descriptions for exact details. Columns include information about the vehicle ticketed, the ticket issued, location, and time.

<https://www.kaggle.com/new-york-city/nyc-parking-tickets/downloads/nyc-parking-tickets.zip/>

* **Collisions**

This is a breakdown of every collision in NYC by location and injury. This data is collected because the NYC Council passed Local Law #11 in 2011. This data is manually run every month and reviewed by the TrafficStat Unit before being posted on the NYPD website. Each record represents a collision in NYC by city, borough, precinct and cross street. This data can be used by the public to see how dangerous/safe intersections are in NYC. The information is presented in pdf and excel format to allow the casual user to just view the information in the easy to read pdf format or use the excel files to do a more in-depth analysis.

<https://data.cityofnewyork.us/resource/qiz3-axqb.geojson>.

<https://data.cityofnewyork.us/api/views/h9gi-nx95/files/b5fd8e71-ca48-4e96-bf63-1b8a7c4cc47b?download=true&filename=Collision_DataDictionary.xlsx>.

## Données pertinentes

Ce tableau présente les données pertinentes retenues pour chacune des thématiques

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Liste de variables retenues | | | |  |
| Thème | **Dimension** | **Variable** | **Type** | **Source de données** |
| Collisions | Géographie | Zip Code | catégorique | Collision dataset  2012 - … |
| Borough | catégorique |
| Temps | Période (à classer -- am, pm, soir, nuit) | catégorique |
| Période (à classer en -- pointe, hors pointe) | catégorique |
| Année | numérique, continue |
| Véhicule | type de véhicule (taxi, truck, passenger vehicle, bus, etc) | catégorique | vehicle type code 1 |
| Impacts des collisions | # de blessés | numérique, discrète |  |
| # de morts | numérique, discrète |
| # de piétons blessés | numérique, discrète |
| # de piétons morts | numérique, discrète |
| Crimes | Géographie | Borough  Zip Code | Catégorique  (Zip code -- par géoref. long-lat) | Dataset sur les crimes 2006 – 2015 |
| Temps | Période (à classer en am, pm, soir, nuit) | catégorique | Hour/exact time |
| Nature de la crime | Description | catégorique | KY\_CD, OFNS DESC |
| Catégorie (felony =3, misdemeanor =2 , violation = 1) | ordinale | Law\_cat\_CD |
| Contraventions | Géographie | Borough | catégorique | Parkings tickets dataset |
| Street name | catégorique |
| Pas de long – lat |  |
| Temps | Période | catégorique |
| Caractéristiques du véhicule | Véhicule Body | catégorique |
| Marque | catégorique |
| Année | numérique, continue |
| Motifs | Violation code | catégorique |

## Techniques d’analyse

Les données sont chargées dans Hadoop puis ont été manipulées avec pig, hive, mapreduce et python en vue d’y extraire des statistiques et déterminer le volume de crimes, d’accidents et de billets de contraventions selon la localisation et la période de la journée.

L’objectif étant d’analyser les données par rapport à la localisation géographique, les statistiques sont en effet calculées en fonction des deux indicateurs géographiques, le disctrict (borough) et le zipcode. Ainsi nous avons manipulé les données pour synthétiser les données et décrire les trois thématiques.

## Collisions ou accidents

* + Les collisions par district (tableau 1.1)
  + Les collisions par zipcode (tableau 1.2)
  + Les collisions selon le zipcode, groupé par district (tableau 1.3).
  + Les collisions selon la période de la journée (tableau 1.4)

## Crimes

* + Les crimes par district (tableau 2.1)
  + Les crimes selon la période (tableau 2.2)
  + Les crimes selon la période, groupé par district (tableau 2.3).

## Mise en œuvre, scripts et résultats

### 9.1 Les collisions avec Pig

Techniquement pour y parvenir, on suit la démarche décrite ci-dessous :

* 1. Préalablement nous avons préparé les fichiers de données en les nettoyant et les mettant au format texte. Nous avons noté l’ordre des champs avant de lancer Cloudera.
  2. Cloudera offre l’éditeur de commandes et nous y avons téléchargé les données dans HDFS.
  3. Uen fois que les données sont dans le système HDFS, nous avons fait usage de Pig, un outil Hadoop, permettant le chargement des données, l’extraction et l’analyse descriptive. À l’invite $ : pig
  4. Chargé des données des fichiers HDFS dans pig pour manipulation et extraction d’information. Nous avons extrait les données du fichier et les mis dans un conteneur (collision5000) et aussi indiquer la structure de données en exécutant la commande suivante. Nous avons extrait les bags de données suivantes :

Date de la collision annee, mois, jour: time

Periode de la journée

Borough

Zip code

latitude

longitude

blessés

morts

piétons blessés

pietons morts

cyclistes blessés

cyclistes morts

chauffeurs blessés

chauffeurs morts

cause principale

code unique

type vehicle

**Commande Pig pour charger les données:**

grunt> collision5000= LOAD '/user/cloudera/tpfinal/collisions5000.txt' USING PigStorage('\t') AS (date:chararray, annee:int, mois:int, jour:int, time:chararray, periode:chararray, quartier:chararray, zipcode:chararray, lat:chararray, longi:chararray, blesses:int,morts:int, pietonsbles:int, pietonsmorts:int, cyclistbles:int, cyclistmorts:int, chauffeursbles:int, chauffeursmorts:int, cause1:chararray, key1:chararray, vehicle:chararray);

**Commandes de manipulation des données:**

* Grouper les données par quartier (district) dans un conteneur nommé quartierdata:

grunt> quartierdata = GROUP collision5000 by quartier;

* Visualiser la structure de quartierdata

grunt> describe quartierdata;  
quartierdata: {group: chararray,collision5000: {(date: chararray,annee: int,mois: int,jour: int,time: chararray,periode: chararray,quartier: chararray,zipcode: chararray,lat: chararray,longi: chararray,blesses: int,morts: int,pietonsbles: int,pietonsmorts: int,cyclistbles: int,cyclistmorts: int,chauffeursbles: int,chauffeursmorts: int,cause1: chararray,key1: chararray,vehicle: chararray)}}

* Générer le tableau du nombre de collisions par district (group)

grunt> quartierCOUNT = FOREACH quartierdata GENERATE group as quartier, COUNT(collision5000.key1);  
Dump quartierCOUNT;

Résultats :

(BRONX, 763)  
(QUEENS, 1407)  
(BROOKLYN, 1545)  
(MANHATTAN, 1094)  
(SATEN ISLAND, 191)

* Créer un tableau de synthèse des collisions par quartier  
  Nous avons repris la même commande que la précédente et y ajouté des fonctions d’aggrégation des bags pertinents, dont le nombre de blessés et de morts.

grunt> quartierCOUNT = FOREACH quartierdata GENERATE group as quartier, COUNT(collision5000.key1), SUM(collision5000.blesses), SUM(collision5000.morts), SUM(collision5000.pietonsbles), SUM(collision5000.pietonsmorts); dump quartierCOUNT ;

**Tableau 1.1: sommaire descriptif des collisions, blessés et morts en 2017 selon le district**

(BRONX, 763, 169, 1, 57, 0)  
(QUEENS, 1407, 317, 1, 74, 1)  
(BROOKLYN, 1545, 456, 2, 112, 1)  
(MANHATTAN, 1094, 183, 0, 80, 0)  
(STATEN ISLAND, 191, 42, 0, 7, 0)  
District Collisions Blessés Morts Blessés morts   
 Piétons Piétons

Les informations exposées dans ce tableau peuvent servir à mesurer le niveau du risque de collisions /d’accidents dans chacun des districts de la Ville de New York. Cette démarche peut être implémentée à diverses échelles spatiales, par exemple par zip code ou carrefour. Ou aussi selon la période de la journée.

* Grouper les données par zip code dans un conteneur nommé zipdata:

grunt> zipdata = GROUP collision5000 by zipcode;

* Visualiser la structure de zipdata

grunt> describe zipdata;  
zipdata: {group: chararray,collision5000: {(date: chararray,annee: int,mois: int,jour: int,time: chararray,periode: chararray,quartier: chararray,zipcode: chararray,lat: chararray,longi: chararray,blesses: int,morts: int,pietonsbles: int,pietonsmorts: int,cyclistbles: int,cyclistmorts: int,chauffeursbles: int,chauffeursmorts: int,cause1: chararray,key1: chararray,vehicle: chararray)}}

* Générer le tableau de synthèse par zip code

zipdata\_stats = FOREACH zipdata GENERATE group as location, COUNT(collision5000.key1) as total\_colli, SUM(collision5000.morts) as total\_morts, SUM(collision5000.blesses) as total\_bless, SUM(collision5000.pietonsmorts) as total\_pietonsmorts, SUM(collision5000.pietonsbles) as total\_pietonsbless;

* Ordonner selon le nombre de collisions et afficher le tableau

zipdata\_stats\_ord = order zipdata\_stats by total\_colli DESC;  
grunt> dump zipdata\_stats\_ord;

Le résultat est un classement des différents zip codes de la ville de New York selon le nombre de collisions enregistrées pendant la période étudiée. Il identifie donc les zones géographiques les plus risquées ou moins risquées en fonction du nombre d’accidents de véhicules. On peut les classer, de même, selon la gravité des accidents en fonction du nombre de morts et de blessés. Les deux dernières colonnes fournissent les effectifs de piétons parmi les victimes.

**Tableau 1.2 sommaire descriptif des collisions, blessés et morts en 2017 par zip code**

District Collisions Blessés Morts P-blessés P-morts  
(11101, 98, 0, 18, 0, 6)  
(11207, 87, 0, 20, 0, 5)  
(11385, 79, 0, 22, 0, 7)  
(11236, 71, 0, 35, 0, 4)  
(10019, 69, 0, 11, 0, 5)  
(11203, 64, 0, 26, 0, 1)  
(11234, 64, 0, 14, 0, 1)  
(10016, 63, 0, 13, 0, 8)  
(11212, 60, 0, 25, 0, 9)

…..

…..

(10069, 1, 0, 0, 0, 0)  
(10118, 1, 0, 0, 0, 0)  
(10154, 1, 0, 0, 0, 0)  
(10172, 1, 0, 0, 0, 0)  
(11005, 1, 0, 1, 0, 0)  
(10000, 1, 0, 0, 0, 0)  
(11040, 1, 0, 0, 0, 0)  
(11360, 1, 0, 0, 0, 0)  
(10020, 1, 0, 0, 0, 0)

* Il est intéressant de savoir dans quel quartier retrouve-t-on les zipcodes oû les déplacements sont parmi les plus dangereux. Pour cela, nous avons regroupé les données du conteneur en fonction de deux colonnes: quartiers/districs et zipcode, coome suit:

zipquartier = GROUP collision5000 by (quartier, zipcode);

* Cette commande est ensuite exécutée pour calculer un tableau de synthèse des données par quartier et zipcode:

zipquartier\_stats=FOREACH zipquartier GENERATE group as location, COUNT(collision5000.key1) as total\_colli, SUM(collision5000.morts) as total\_morts, SUM(collision5000.blesses) as total\_bless, SUM(collision5000.pietonsmorts) as total\_pietonsmorts, SUM(collision5000.pietonsbles) as total\_pietonsbless ;

* Cette commande permet de classer les zipcodes selon le nombre de collisions:

zipquartier\_stats\_ord = ORDER zipquartier\_stats by total\_colli DESC;

**Tableau 1.3 Classement des zip codes selon le nombre de collisions en 2017**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| zipcode | collisions | morts | blessés | piétons morts | piétons blessés |
| 10467 | 58 | 0 | 19 | 0 | 7 |
| 11212 | 60 | 0 | 25 | 0 | 9 |
| 10016 | 63 | 0 | 13 | 0 | 8 |
| 11234 | 64 | 0 | 14 | 0 | 1 |
| 11203 | 64 | 0 | 26 | 0 | 1 |
| 10019 | 69 | 0 | 11 | 0 | 5 |
| 11236 | 71 | 0 | 35 | 0 | 4 |
| 11385 | 79 | 0 | 22 | 0 | 7 |
| 11207 | 87 | 0 | 20 | 0 | 5 |
| 11101 | 98 | 0 | 18 | 0 | 6 |

**Sortie de pig**

Disctrict zipcode collisionsmorts blessésp-morts p-blessés

((BRONX, 10467), 58, 0, 19, 0, 7)

((BROOKLYN, 11212), 60, 0, 25, 0, 9)  
((MANHATTAN,10016), 63, 0, 13, 0, 8)  
((BROOKLYN,11234), 64, 0, 14, 0, 1)  
((BROOKLYN,11203), 64, 0, 26, 0, 1)  
((MANHATTAN,10019), 69, 0, 11, 0, 5)  
((BROOKLYN,11236), 71, 0, 35, 0, 4)  
((QUEENS,11385), 79, 0, 22, 0, 7)  
((BROOKLYN,11207), 87, 0, 20, 0, 5)  
((QUEENS,11101), 98, 0, 18, 0, 6)

...

((MANHATTAN,10000), 1, 0, 0, 0, 0)  
((MANHATTAN,10020), 1, 0, 0, 0, 0)  
((QUEENS,11360), 1, 0, 0, 0, 0)  
((QUEENS,11040), 1, 0, 0 0 0)  
((MANHATTAN,10172), 1, 0, 0, 0, 0)  
((MANHATTAN,10154), 1, 0, 0, 0, 0)  
((QUEENS,11005), 1, 0, 1, 0, 0)   
((MANHATTAN,10118), 1, 0, 0, 0, 0)  
((MANHATTAN,10069), 1, 0, 0, 0, 0)  
((QUEENS,11208), 1, 0, 0, 0, 0)

* Incorporer la dimension temporelle dans l’analyse des collisions en:
  + Regroupant les données selon la période de la journée:

grunt> periodata = GROUP collision5000 by periode;

* + comptabilisant les collisions pour chaqune des périodes:

grunt> periodCOUNT = FOREACH periodata GENERATE group as periode, COUNT(collision5000.key1), SUM(collision5000.blesses), SUM(collision5000.morts), SUM(collision5000.pietonsbles), SUM(collision5000.pietonsmorts);

* + visualisant le résultat du tableau généré

grunt> dump periodCOUNT;

**Tableau 1.4 Sommaire descriptif des collisions, blessés et morts en 2017 selon la période**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Période | Collisions | Blessés | Morts | P-blessés | P-morts |
| (0h00-6h00 | 442 | 140 | 0 | 29 | 0) |
| (6h00-9h00 | 595 | 143 | 0 | 46 | 0) |
| (9h00-15h00 | 1823 | 324 | 1 | 109 | 0) |
| (15h00-18h00 | 1043 | 239 | 1 | 76 | 0) |
| (18h00-24h00 | 1097 | 321 | 2 | 70 | 2) |

Ce tableau permet de faire ressortir les plages horaires les plus à risque en regard au nombre d’accidents et de blessés enregistrés à l’heure.

**Tableau 1.5 Collisions et blessés à l’heure selon la période de la journée en 2017**

|  |  |  |  |
| --- | --- | --- | --- |
| Plage horaire | Collision | Blessés | Piétons blessés |
| 0h00-6h00 | 74 | 23 | 5 |
| 6h00-9h00 | 198 | 48 | 15 |
| 9h00-15h00 | 304 | 54 | 18 |
| 15h00-18h00 | 348 | 80 | 25 |
| 18h00-24h00 | 183 | 54 | 12 |

### 9.2 Les crimes avec Hive et MapReduce

* **Outils de manipulation:**

Les données relatives aux crimes sont plutôt analysées avec Hive et MapReduce Python.

* **Création, extraction et chargement des données:**

D’abord nous avons créé une table nommée crime et y versé les données extraites du fichier local nommé crimes\_nyc\_txt par requêtes HIVE.

CREATE TABLE crime (

CMPLNT\_NUM INT, //numero unique

CMPLNT\_FR\_DT ARRAY<DATE>, //date

CMPLNT\_FR\_TM STRING, //heure

KY\_CD STRING, //description

LAW\_CAT\_CD STRING, //catégorie de crime

BORO\_NM STRING //district ou borough //localisation

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY “\t>”

STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH '/home/cloudera/crimes\_nyc.txt' OVERWRITE INTO TABLE crime;

* Tableau de synthèse sur les crimes par district

Ensuite, nous avons créé un tableau de synthèse présentant le nombre de crimes par quartier classé par ordre d’importance.

SELECT boro\_nm, count(boro\_nm) as crime\_borough

FROM crime GROUP BY boro\_nm ORDER BY crime\_borough DESC;

**Tableau 2.1 Nombres de crimes enregistrés par district**

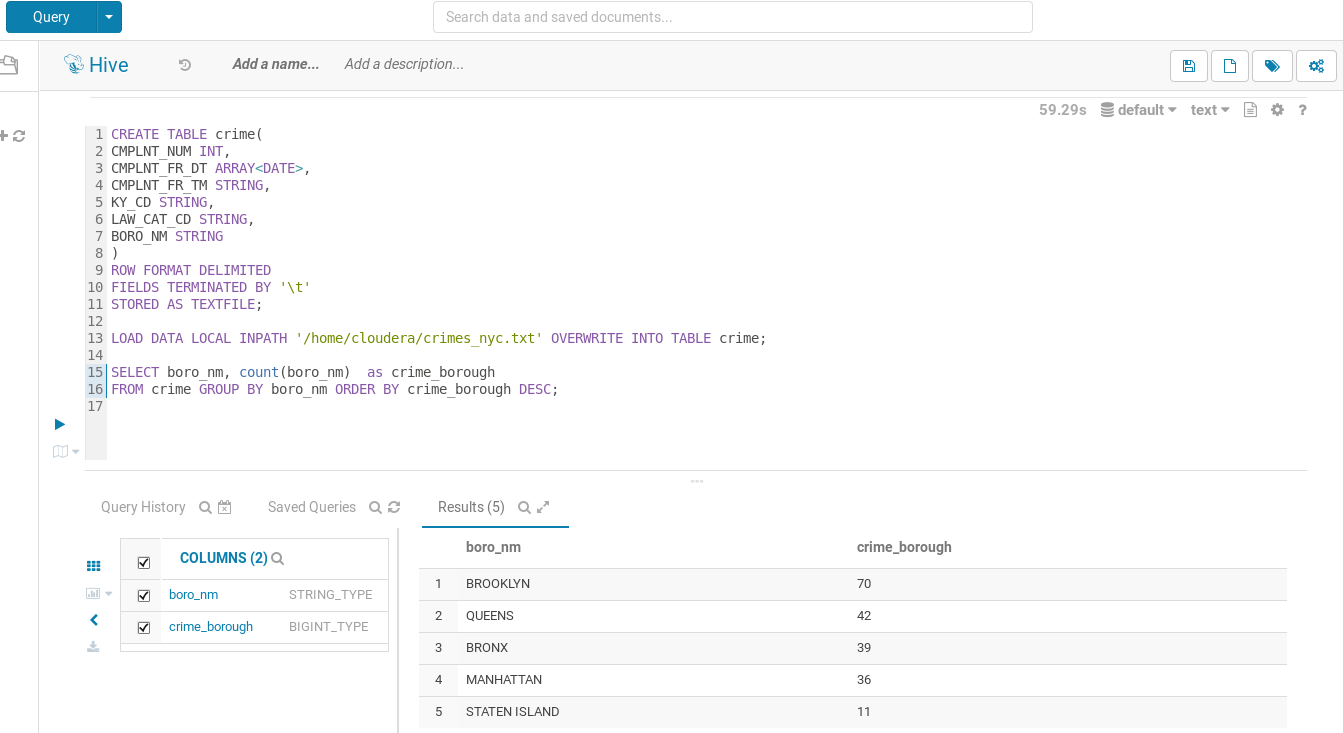
1 BROOKLYN 70

2 QUEENS 42

3 BRONX 39

4 MANHATTAN 36

5 STATEN ISLAND 11



* Nombre de crimes selon la période de la journée

En utilisant une requête HIVE, nous avons obtenu la liste de crimes entre 20 :00 et 22 :00.

SELECT \* FROM crime WHERE CMPLNT\_FR\_TM BETWEEN "20:00:00" AND "22:00:00";

**Tableau 2.2**



À partir de Mapreduce et Python, nous avons pu obtenir les mêmes résultats en exécutant le script suivant qui fournit le nombre de crimes par catégorie:

from mrjob.job import MRJob

class MRCrimesByBorough(MRJob):

def mapper(self, key, line):

(CMPLNT\_NUM, CMPLNT\_FR\_DT, CMPLNT\_FR\_TM, KY\_CD, LAW\_CAT\_CD, BORO\_NM) = line.split('\t')

yield LAW\_CAT\_CD, 1

def reducer(self, LAW\_CAT\_CD, occurences):

yield LAW\_CAT\_CD, sum(occurences)

if \_\_name\_\_ == '\_\_main\_\_':

MRCrimesByBorough.run()

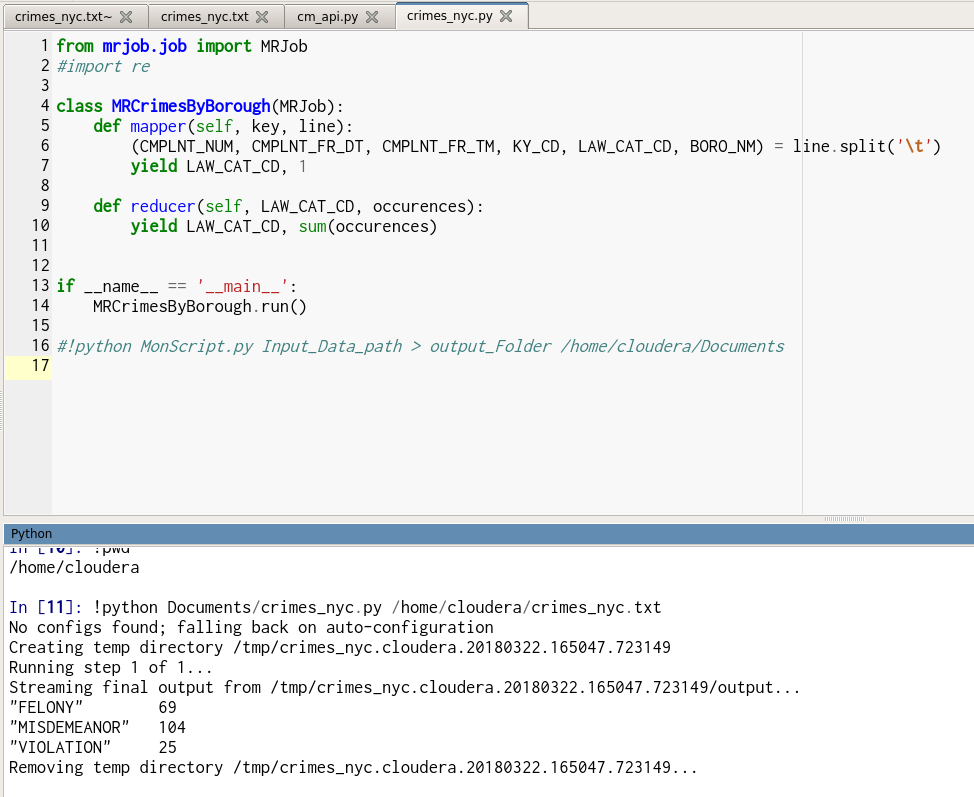


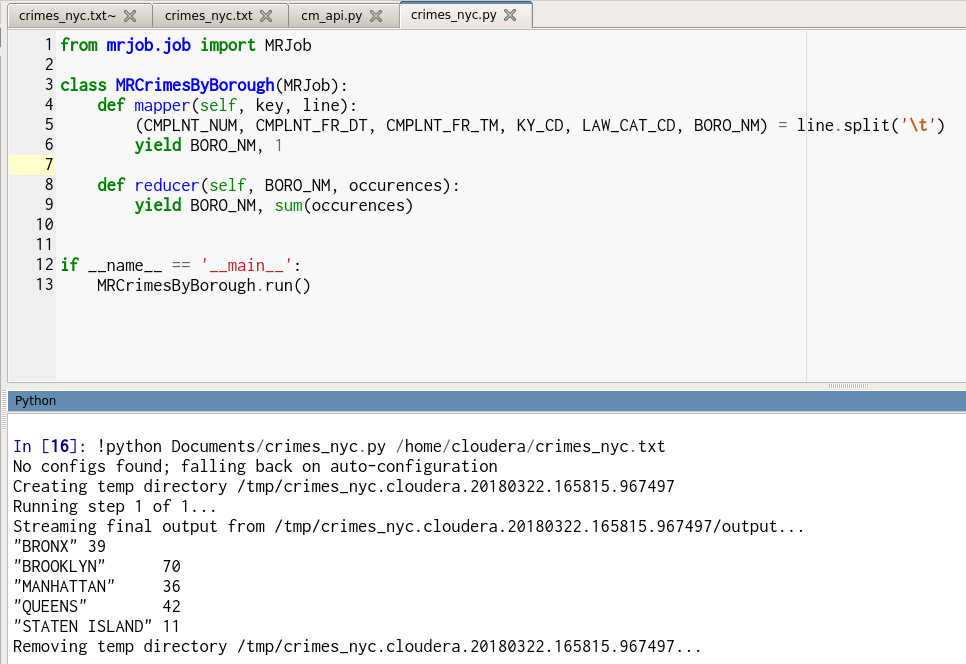
Table 2.3: Occurences  de crimes par categorie

"FELONY" 69

"MISDEMEANOR" 104

"VIOLATION" 25

Ce tableau montre les types de crimes selon leur fréquence. Mais avec la commande suivante nous avons recalculé, à partir de MapReduce et Python , le nombre de crimes enregistrés par disctricts:



Les résultats sont les mêmes que ceux obtenus avec HiveQL présentés au tableau 2.1

### **9.3 Les contraventions avec Hive et Pig**

Les mêmes scripts que précedemment juste en modifiant les noms des colonnes sachant que la structure des données sont identiques.

## Conclusion

Les manipulations opérées à partir des scripts exécutés à l’aide de Pig, Hive, MapReduce et Python montrent la capacité d’exploiter de larges datasets pour resoudre des problèmes réels. Nous avons pu à partir des trois datasets analysés localiser les zones géographiques selon l’occurrence des collisions, des crimes ou des contraventions survenus au cours des dernières années.

Cependant il est important de noter que le travail de manipulation et d’analyse des datasets a été limité à cause de la faible capacité de notre infrastructure (un seule machine). Si on avait accès à un véritable cluster, on pourrait analyser à l’échelle géographique élémentaire, le point géographique à partir de la longitude et la latitude. Ce qui aurait permis d’avoir une meilleure précision de l’information.

Par ailleurs, nous avons pu constater la capacité de Pig et Hive à croiser et à joindre plusieurs bags pour arriver à des tableaux croisant plusieurs paramètres.

Ces scripts pourrait servir à traiter les datasets thématiques et alimentés en information régulièrement des applications web mises au service des citoyens lors de leur déplacement à travers la ville.

Côté pédagogique, ce travail nous a permis d’investir du temps dans l’apprentissage de ces outils incorporés dans la plateforme Cloudera Hadoop, allant de la préparation de l’environnement de travail, en passant par le chargement et la manipulation des données jusqu’à l’extraction d’information à l’aide de requêtes Pig, Hive, ou Python exécutées avec MapReduce.

Nous avions eu peur au début avec le crash des machines virtuelles et la nouveauté de ces approches big data, mais finalement avec moins de données et les suggestions de l’enseignante, nous arrivons finalement à comprendre le fonctionnent de ces puissants outils appropriés aux données non structurées, semi structurées ou structurées.