UQÀM | Faculté des sciences
Université du Québec à Montréal

INF8810 – **Traitement et analyse des données massives**

En groupe de 4 personnes

Projet de Session #1 :

ANALYSE DE DONNÉES AVEC MAPREDURE ET SPARK SQL

Session d'Automne 2022

Table des matières

I. Acquisition et Préparation des données

1. Informations sur la source de données

Dataset : Retards et annulations de vols en 2015

Lien : Retards et annulations de vols 2015 | Kaggle

Type : CSV

Description :

Sur quelle compagnie aérienne devriez-vous voler pour éviter des retards importants ?

« Retards et annulations de vol en 2015 » est une source de données recueillies et publiées par le « Bureau of Transportation Statistics du BTS » pour suivre le rendement à temps des vols intérieurs exploités par les grands transporteurs aériens.

Le data set contient des informations des activités de 14 compagnies aériennes effectuant des vols sur un ensemble de 323 destinations différentes. Ces informations sont les détails sur les différents vols, comme la date, la durée, le retard, l'état (annulé ou non), les circonstances (dérouté ou non) du vol et tant d'autres détails.

La source de données est composée de trois (3) fichiers de type Excel, ce sont :

- **airlines.csv**
  Les informations sur les compagnies aériennes.

Attribut	Type	Description
IATA_CODE	Texte	Code de la compagnie
AIRLINE	Texte	Titre de la compagnie

- **airports.csv**
  Les informations sur les aéroports qui représentent les départs et/ou destinations

Attribut	Type	Description
IATA_CODE	Texte	Code l'aéroport
AIRPORT	Texte	Titre de l'aéroport
CITY	Texte	La ville dans laquelle se situe l'aéroport
STATE	Texte	Le code de l'État Américain où se situe l'aéroport
COUNTRY	Texte	Le pays où se situe l'aéroport
LATITUDE	Nombre	La latitude
LONGITUDE	Nombre	La longitude

- **flights.csv**
  Les informations sur le vol proprement dit, de la programmation du départ du vol à son arrivée ou à son annulation.

Attribut	Type	Description
YEAR	Nombre	Année pendant laquelle le vol a été effectué
MONTH	Nombre	Mois durant lequel le vol a été effectué (1 – 12)
DAY	Nombre	Jour durant lequel le vol a été effectué (1 – 31)
DAY_OF_WEEK	Nombre	N° Jour durant lequel le vol a été effectué (1 – 31)
AIRLINE	Texte	La compagnie aérienne
FLIGHT_NUMBER	Nombre	Le numéro du vol
TAIL_NUMBER	Texte	Le numéro de l'avion de la compagnie
ORIGIN_AIRPORT	Texte	Code de l'aéroport de départ

DESTINATION_AIRPORT	Texte	Code de l'aéroport d'arrivée
SCHEDULED_DEPARTURE	Nombre	N° d'ordre de départ
DEPARTURE_TIME	Nombre	Heure du départ
DEPARTURE_DELAY	Nombre	Délais du départ
TAXI_OUT	Nombre	Heure du décollage
WHEELS_OFF	Nombre	Heure à laquelle les roues sont rentrées
SCHEDULED_TIME	Nombre	Temps du vol prévu
ELAPSED_TIME	Nombre	Durée effectuée par le vol
AIR_TIME		
DISTANCE	Nombre	Distance entre les deux aéroports (départ – arrivée)
WHEELS_ON	Nombre	Heure à laquelle les roues sont rentrées
TAXI_IN	Nombre	Heure de l'atterrissage
SCHEDULED_ARRIVAL	Nombre	Temps d'arrivée prévu
ARRIVAL_TIME	Nombre	Heure d'arrivée
ARRIVAL_DELAY	Nombre	Délais de l'arrivée
DIVERTED	Nombre	Vol dérouté (1 si oui et 0 si non)
CANCELLED	Nombre	Vol annulé (1 si oui et 0 si non)
CANCELLATION_REASON	Texte	Raison d'annulation du vol (A - Airline/Carrier; B - Weather; C - National Air System; D - Security)
AIR_SYSTEM_DELAY	Nombre	Retard dû à un réglage de type système
SECURITY_DELAY	Nombre	Retard dû à un réglage de type sécurité
AIRLINE_DELAY	Nombre	Retard causé par le vol
LATE_AIRCRAFT_DELAY	Nombre	Retard de l'avion
WEATHER_DELAY	Nombre	Retard causé par le climat

2. Préparation des données : réalisé avec python sur googlecolab

Informations de base différents fichiers

- **airlines.csv**

Échantillon des lignes (5 premières lignes)

	IATA_CODE	AIRLINE
0	UA	United Air Lines Inc.
1	AA	American Airlines Inc.
2	US	US Airways Inc.
3	F9	Frontier Airlines Inc.
4	B6	JetBlue Airways

À l'aide de la fonction **info()** de pandas, nous pouvons obtenir le nombre de lignes, de colonnes, ainsi qu'un vu d'ensembles ressortant **nombre de ligne non-null** et le **type** de chaque colonne.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   IATA_CODE   14 non-null     object
 1   AIRLINE     14 non-null     object
dtypes: object(2)
memory usage: 352.0+ bytes
None
```

- **airports.csv**

Échantillon des lignes (5 premières lignes)

	IATA_CODE	AIRPORT	CITY	STATE	COUNTRY	LATITUDE	LONGITUDE
0	ABE	Lehigh Valley International Airport	Allentown	PA	USA	40.65236	-75.44040
1	ABI	Abilene Regional Airport	Abilene	TX	USA	32.41132	-99.68190
2	ABQ	Albuquerque International Sunport	Albuquerque	NM	USA	35.04022	-106.60919
3	ABR	Aberdeen Regional Airport	Aberdeen	SD	USA	45.44906	-98.42183
4	ABY	Southwest Georgia Regional Airport	Albany	GA	USA	31.53552	-84.19447

À l'aide de la fonction **info()** de pandas, nous pouvons obtenir le nombre de lignes, de colonnes, ainsi qu'un vu d'ensembles ressortant **nombre de ligne non-null** et le **type** de chaque colonne.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 322 entries, 0 to 321
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   IATA_CODE   322 non-null    object 
 1   AIRPORT     322 non-null    object 
 2   CITY        322 non-null    object 
 3   STATE       322 non-null    object 
 4   COUNTRY     322 non-null    object 
 5   LATITUDE    319 non-null    float64
 6   LONGITUDE   319 non-null    float64
dtypes: float64(2), object(5)
memory usage: 17.7+ KB
None
```

- **flights.csv**

Vue partielle d'un échantillon des lignes (5 premières lignes)

	YEAR	MONTH	DAY	DAY_OF_WEEK	AIRLINE	FLIGHT_NUMBER	TAIL_NUMBER	ORIGIN_AIRPORT	DESTINATION_AIRPORT	SCHEDULED_DEPARTURE	...	ARRIVAL_TIME
0	2015	1	1	4	AS	98	N407AS	ANC	SEA	5	...	408.0
1	2015	1	1	4	AA	2336	N3KUAA	LAX	PBI	10	...	741.0
2	2015	1	1	4	US	840	N171US	SFO	CLT	20	...	811.0
3	2015	1	1	4	AA	258	N3HYAA	LAX	MIA	20	...	756.0
4	2015	1	1	4	AS	135	N527AS	SEA	ANC	25	...	259.0

5 rows × 31 columns

À l'aide de la fonction **info()** de pandas, nous pouvons obtenir le nombre de lignes, de colonnes, ainsi qu'un vu d'ensembles ressortant le **type** de chaque colonne.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5819079 entries, 0 to 5819078
Data columns (total 31 columns):
 #   Column               Dtype
---  ------               -----
 0   YEAR                 int64
 1   MONTH                int64
 2   DAY                  int64
 3   DAY_OF_WEEK          int64
 4   AIRLINE              object
 5   FLIGHT_NUMBER        int64
 6   TAIL_NUMBER          object
 7   ORIGIN_AIRPORT       object
 8   DESTINATION_AIRPORT  object
 9   SCHEDULED_DEPARTURE  int64
 10  DEPARTURE_TIME       float64
 11  DEPARTURE_DELAY      float64
 12  TAXI_OUT             float64
 13  WHEELS_OFF           float64
 14  SCHEDULED_TIME       float64
 15  ELAPSED_TIME         float64
 16  AIR_TIME             float64
 17  DISTANCE             int64
 18  WHEELS_ON            float64
 19  TAXI_IN              float64
 20  SCHEDULED_ARRIVAL    int64
 21  ARRIVAL_TIME         float64
 22  ARRIVAL_DELAY        float64
 23  DIVERTED             int64
 24  CANCELLED            int64
 25  CANCELLATION_REASON  object
 26  AIR_SYSTEM_DELAY     float64
 27  SECURITY_DELAY       float64
 28  AIRLINE_DELAY        float64
 29  LATE_AIRCRAFT_DELAY  float64
 30  WEATHER_DELAY        float64
dtypes: float64(16), int64(10), object(5)
memory usage: 1.3+ GB
None
```

Étant données les observations obtenues, nous nous attarderons sur le nettoyage du dataset flights qui est celui qui dispose l'essentiel du bruit dans les données.

Taux de valeurs manquantes des colonnes disposant au moins une valeur manquante

```
[21] def missing_data_percent(df):
       missing_data= 100 * df.isna().sum() / len(df)
       missing_data= missing_data[missing_data > 0].sort_values(ascending=False)
       return missing_data
```

```
missing_data = missing_data_percent(df_flight)
missing_data
```

```
CANCELLATION_REASON    98.455357
WEATHER_DELAY          81.724960
LATE_AIRCRAFT_DELAY    81.724960
AIRLINE_DELAY          81.724960
SECURITY_DELAY         81.724960
AIR_SYSTEM_DELAY       81.724960
ELAPSED_TIME            1.805629
AIR_TIME                1.805629
ARRIVAL_DELAY           1.805629
ARRIVAL_TIME            1.589822
TAXI_IN                 1.589822
WHEELS_ON               1.589822
WHEELS_OFF              1.530259
TAXI_OUT                1.530259
DEPARTURE_TIME          1.480526
DEPARTURE_DELAY         1.480526
TAIL_NUMBER             0.252978
SCHEDULED_TIME          0.000103
dtype: float64
```

Dans ce fichier plusieurs valeurs sont nulles, cela ne veut pourtant pas dire que la qualité des données est mauvaise. Nous remarquons que celles-ci n'ont pas été renseignées lorsque certains évènements n'ont pas eu lieu durant le vol. C'est par exemple pour les colonnes: AIR_SYSTEM_DELAY, SECURITY_DELAY, AIRLINE_DELAY, LATE_AIRCRAFT_DELAY et WEATHER_DELAY, CANCELLATION_REASON.

Par contre les valeurs manquantes de la colonne TAIL_NUMBER sont dues à une erreur. Dans le cadre de notre travail, ces colonnes n'entrent pas en jeu et nous allons tout simplement les supprimer.

Réduction de la dimension du dataset pour ne garder que les colonnes qui nous intéressent :

```
[29] df_flight1 = df_flight.drop(['DAY_OF_WEEK', 'TAIL_NUMBER', 'SCHEDULED_DEPARTURE', 'DEPARTURE_TIME', 'DEPARTURE_DELAY', 'TAXI_OUT',
                                  'WHEELS_OFF', 'SCHEDULED_TIME', 'ELAPSED_TIME', 'AIR_TIME', 'DISTANCE',  'WHEELS_ON', 'TAXI_IN',
                                 'ARRIVAL_DELAY', 'DIVERTED', 'CANCELLED'], axis=1)
```

```
df_flight1.head()
```

	YEAR	MONTH	DAY	AIRLINE	FLIGHT_NUMBER	ORIGIN_AIRPORT	DESTINATION_AIRPORT	SCHEDULED_ARRIVAL	ARRIVAL_TIME
0	2015	1	1	AS	98	ANC	SEA	430	408.0
1	2015	1	1	AA	2336	LAX	PBI	750	741.0
2	2015	1	1	US	840	SFO	CLT	806	811.0
3	2015	1	1	AA	258	LAX	MIA	805	756.0
4	2015	1	1	AS	135	SEA	ANC	320	259.0

Examinons et traitons les valeurs manquantes résiduelles

```
df_flight.info(show_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5819079 entries, 0 to 5819078
Data columns (total 9 columns):
 #   Column               Non-Null Count    Dtype
---  ------               --------------    -----
 0   YEAR                 5819079 non-null  int64
 1   MONTH                5819079 non-null  int64
 2   DAY                  5819079 non-null  int64
 3   AIRLINE              5819079 non-null  object
 4   FLIGHT_NUMBER        5819079 non-null  int64
 5   ORIGIN_AIRPORT       5819079 non-null  object
 6   DESTINATION_AIRPORT  5819079 non-null  object
 7   SCHEDULED_ARRIVAL    5819079 non-null  int64
 8   ARRIVAL_TIME         5726566 non-null  float64
dtypes: float64(1), int64(5), object(3)
memory usage: 399.6+ MB
```

```
[36] missing_data = missing_data_percent(df_flight)
     missing_data
```

```
ARRIVAL_TIME    1.589822
dtype: float64
```

Étant donné que la seule colonne ayant les valeurs manquantes **ARRIVAL_TIME** avec un taux de **1,59%** sur **5819079** lignes de données, nous avons décidé de supprimer ces lignes de données manquantes.

```
[39] df_flight.dropna (inplace = True)
```

Jointure des datasets « flights » et « airlines »

Pour des raisons liées à notre première question sélectionnée pour le MapReduce, nous avons décidé de joindre ces deux fichiers. Le dataset **« airports »** sera éventuellement utilisé dans le cadre de la question sur spark

```
df_flight = df_flight.rename(columns={'AIRLINE':'AIRLINE_CODE'})
result= pd.merge(df_flight, df_airline, how="inner", left_on="AIRLINE_CODE", right_on= "IATA_CODE")
result.drop(['AIRLINE_CODE'], axis=1, inplace = True)
result.head()
```

	YEAR	MONTH	DAY	FLIGHT_NUMBER	ORIGIN_AIRPORT	DESTINATION_AIRPORT	SCHEDULED_ARRIVAL	ARRIVAL_TIME	IATA_CODE	AIRLINE
0	2015	1	1	98	ANC	SEA	430	408.0	AS	Alaska Airlines Inc.
1	2015	1	1	135	SEA	ANC	320	259.0	AS	Alaska Airlines Inc.
2	2015	1	1	108	ANC	SEA	509	455.0	AS	Alaska Airlines Inc.
3	2015	1	1	122	ANC	PDX	525	507.0	AS	Alaska Airlines Inc.
4	2015	1	1	130	FAI	SEA	548	545.0	AS	Alaska Airlines Inc.

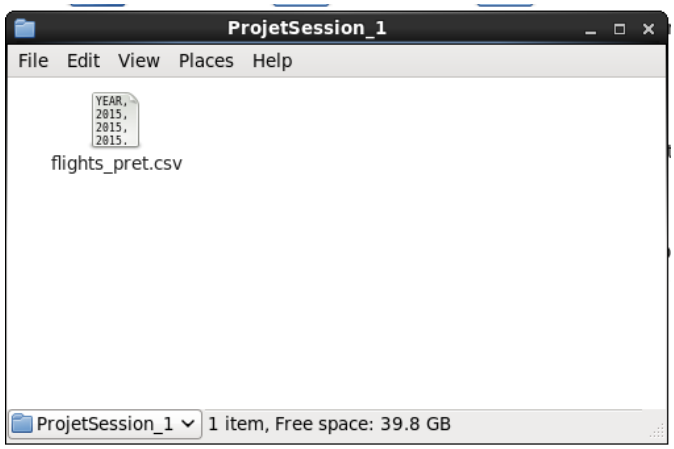
Le dataset **result** contient **5726566 lignes** et **10 colonnes**. Nous allons maintenant le sauvegarder dans le fichier .csv nommé **"flights_pret.csv".**

```
result.to_csv("flights_pret.csv", index=False)
```

II. Ajout de la source de données dans HDFS

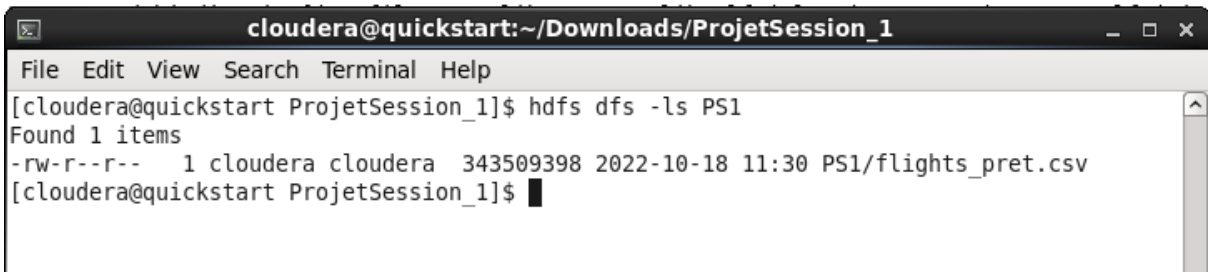
En ce qui concerne l'ajout du fichier **"flights_pret.csv"** dans HDFS, nous avons procédé comme suit :

- Connexion à la machine cloudera, puis téléchargement du fichier depuis le OneDrive de l'UQAM dans un répertoire local **« /home/cloudera/Downloads/ProjetSession_1 »** crée à cet effet (voir figure)



- Importation du fichier dans le dossier **PS1** de HDFS à partir d'un terminal ouvert dans le dossier **ProjetSession_1** avec la commande :
  [cloudera@quickstart ProjetSession_1]$ hdfs dfs -put flights_pret.csv PS1

- Vérifions que le fichier a effectivement été copié avec la commande :
  [cloudera@quickstart ProjetSession_1]$ hdfs dfs -ls PS1



III. Premier traitement et code MapReduce

QUESTION : Quel est le pourcentage de retard de chaque compagnie aérienne durant l'année ?

Mapper

```
#Declaration de l'environnement et choix de l'encodage
#! usr/bin/env python
#-*-coding:utf-8-*-

import sys

for line in sys.stdin:
 #Supprimer les espaces en debut et fin de la ligne courrante
 line = line.strip()
 #Chaque ligne est reparti en un tableau d'element avec le separateur ','
 token = line.split(',')
 #Gestion des erreurs lors de la conversion de type
 try:
  #Conversion en float la duree prevue et la duree effectuee respectivement aux positions 6 et 7 dans la liste
"token"
  duree_prevue = float(token[6])
  duree_effectuee = float(token[7])
  #Recuperation de la compagnie aerienne
  compagnie = token[9].replace(" ","_")
  #Afficher dans le terminal la compagnie et la difference entre la duree prevue et celle éffectuée
  #La difference est negative s'il y'a eu un retard et positive ou nulle dans le cas contraire
  print("%s\t%s" % (compagnie,(duree_prevue - duree_effectuee)))
 except ValueError:
  #En cas d'erreur, ignorer la ligne
  continue
```

Reducer

```
#Declaration de l'environnement et choix de l'encodage
#! usr/lib/env python
#-*-coding:utf-8-*-

import sys

#Dictionnaire qui aura pour cle la compagnie et valeur une liste
compagnie2retard = {}
for line in sys.stdin:
 #Suppression des espaces en fin et debut de ligne
 line = line.strip()
 #La ligne est repartie en fonction du separateur '\t'
 compagnie,diff_temps = line.split('\t')
 try:
  #Conversion en float de la valeur
  diff_temps = float(diff_temps)
 except ValueError:
  continue

 try:
  #Si la valeur de diff_temps est negative, i.e un retard, on ajoute alors pour la compagnie un nouveau retard
representee ici par 1.
  if diff_temps < 0.0:
   compagnie2retard[compagnie].append(1)
  else:
   compagnie2retard[compagnie].append(0)
 except:
  #Si la cle de la compagnie n'existe pas le dictionnaire, on cree la cle
  compagnie2retard[compagnie] = []
  #on ajoute dans la liste la valeur 1 s'il s'agit d'un retard et 0 dans le cas contraire
  if diff_temps < 0.0:
   compagnie2retard[compagnie].append(1)
  else:
   compagnie2retard[compagnie].append(0)

#On recupere la cle (k) et la valeur(v: une liste)
for k, v in compagnie2retard.items():
 #On calcule le pourcentage de retard de la compagnie et on arrondi a 2 chiffres apres la virgule
 som = sum(v)
 #Cast de la taille de la liste en float pour division floattante
 taille = float(len(v))
 moy = round((som/taille)*100,2)
 print("%s\t%s" % (k, moy))
```

Exécution et Résultat en sortie :

```
[cloudera@quickstart PS1]$ cat flights_pret.csv | python mapper.py | python reducer.py | sort -n -k 2,2
Delta_Air_Lines_Inc.    28.25
Alaska_Airlines_Inc.    32.33
American_Airlines_Inc.  34.46
United_Air_Lines_Inc.   35.57
American_Eagle_Airlines_Inc.    36.8
Southwest_Airlines_Co.  36.85
JetBlue_Airways 37.17
Atlantic_Southeast_Airlines     37.97
Skywest_Airlines_Inc.   38.15
Virgin_America  38.29
US_Airways_Inc. 38.37
Hawaiian_Airlines_Inc.  39.42
Frontier_Airlines_Inc.  43.39
Spirit_Air_Lines        46.58
```

IV. Deuxième traitement et code Spark SQL

QUESTION : Quelles sont les statistiques sur les trois compagnies ayant le moins de retard en termes de nombre de vols et de destinations durant l'année 2015 ?

- Sauvegarde de l'exécution de la première question pour utilisation avec pySpark **[cloudera@quickstart PS1]$ cat flights_pret.csv | python mapper.py | python reducer.py | sort -n -k 2,2 > compagnieClassement.txt**
- Importation du fichier « compagnieClassement.txt » dans HDFS **[cloudera@quickstart PS1]$ hdfs dfs -put compagnieClassement.txt PS1**

Code pyspark

```
#! /urs/env/lib python
#-*-coding:utf-8-*-

from pyspark.sql import SQLContext
from pyspark import SparkContext
from pyspark.sql.types import *
from pyspark.sql.functions import *

sc = SparkContext("local","codeSparksql")
sqlContext = SQLContext(sc)

#Creation du schema du futur dataframe
flight_schema = StructType([StructField("YEAR",IntegerType(),True),
StructField("MONTH",IntegerType(),True),
StructField("DAY",IntegerType(),True),
StructField("FLIGHT_NUMBER",IntegerType(),True),
StructField("ORIGIN_AIRPORT",StringType(),True),
StructField("DESTINATION_AIRPORT",StringType(),True),
StructField("SCHEDULED_ARRIVAL",FloatType(),True),
StructField("ARRIVAL_TIME",FloatType(),True),
StructField("IATA_CODE",StringType(),True),
StructField("AIRLINE",StringType(),True)])

#Creation du dataframe flight
flight = sc.textFile("/user/cloudera/PS1/flights_pret.csv")
flight_rdd = flight.map(lambda x: x.split(",")).map(lambda x: (int(x[0]), int(x[1]), int(x[2]), int(x[3]), x[4],
x[5], float(x[6]), float(x[7]), x[8], x[9]))
flight_df = sqlContext.createDataFrame(flight_rdd, flight_schema)

#Afficher les 10 premieres lignes du dataframe
print("\nLes dix premières lignes du dataframe Flights\n")
flight_df.show(10)

#Afficher le schema du dataframe pour voir les colonnes et leur type de donnee
print("\nSchéma du dataframe Flights\n")
flight_df.printSchema()

#Recuperation des trois compagnies qui ont moins de retard
compagnie = sc.textFile("/user/cloudera/PS1/compagnieClassement.txt")
compagnie_rdd = compagnie.map(lambda x: x.split("\t")).map(lambda x: ( x[0], float(x[1])))

#Recuperation des 3 compagnies avec le plus faible taux de retard
compagnie_header = ["compagnie","Taux de retard"]
compagnie_df = sqlContext.createDataFrame(compagnie_rdd,compagnie_header)
```

```
top_3_df = compagnie_df.select(regexp_replace("compagnie","_"," ").alias("compagnie"),"Taux de retard").orderBy
(asc("Taux de retard")).limit(3)
print("\nLes trois compagnies ayant le moins de retard\n")
top_3_df.show()
top_3_lst = top_3_df.select("compagnie").rdd.flatMap(lambda x: x).collect()
#print(top_3_lst)

#Calcul du nombre de vols par compagnie par mois et par destination des trois compagnies avec le plus faible
taux #de retard
cpd = flight_df.select("*").where((col("AIRLINE") == top_3_lst[0]) | (col("AIRLINE") == top_3_lst[1]) | (col
("AIRLINE") == top_3_lst[2]))

print("\nNombre de vols par compagnie par mois et par destination des trois compagnies avec le plus faible taux
de retard\n")
destination_vol = cpd.select("YEAR","MONTH","AIRLINE","DESTINATION_AIRPORT").groupBy
("MONTH","AIRLINE","DESTINATION_AIRPORT").count().orderBy("MONTH","DESTINATION_AIRPORT")
destination_vol.show(10)

print("\nLe nombre de vols enregistres par compagnie durant l'année, ainsi que le nombre de destination
différente désservie\n")

cpd.registerTempTable ("Table_Airline")
sqlContext.sql("select AIRLINE, count(AIRLINE) AS Nbre_Vol, count(DISTINCT(DESTINATION_AIRPORT)) AS
Nbre_Destinations from Table_Airline Group By AIRLINE").show()
```

NOS AFFICHAGES :

- **Vérification du chargement effectif du dataframe flights_pret**

```
Les dix premières lignes du dataframe Flights

22/10/19 10:56:26 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.
+----+-----+---+-------------+-------------+-------------------+----------------+------------+---------+--------------------+
|YEAR|MONTH|DAY|FLIGHT_NUMBER|ORIGIN_AIRPORT|DESTINATION_AIRPORT|SCHEDULED_ARRIVAL|ARRIVAL_TIME|IATA_CODE|             AIRLINE|
+----+-----+---+-------------+-------------+-------------------+----------------+------------+---------+--------------------+
|2015|    1|  1|           98|          ANC|                SEA|           430.0|       408.0|       AS|Alaska Airlines Inc.|
|2015|    1|  1|          135|          SEA|                ANC|           320.0|       259.0|       AS|Alaska Airlines Inc.|
|2015|    1|  1|          108|          ANC|                SEA|           509.0|       455.0|       AS|Alaska Airlines Inc.|
|2015|    1|  1|          122|          ANC|                PDX|           525.0|       507.0|       AS|Alaska Airlines Inc.|
|2015|    1|  1|          130|          FAI|                SEA|           548.0|       545.0|       AS|Alaska Airlines Inc.|
|2015|    1|  1|          134|          ANC|                SEA|           633.0|       558.0|       AS|Alaska Airlines Inc.|
|2015|    1|  1|          144|          ANC|                PDX|           630.0|       619.0|       AS|Alaska Airlines Inc.|
|2015|    1|  1|          114|          ANC|                SEA|           640.0|       628.0|       AS|Alaska Airlines Inc.|
|2015|    1|  1|          695|          GEG|                SEA|           610.0|       610.0|       AS|Alaska Airlines Inc.|
|2015|    1|  1|          730|          ANC|                SEA|           930.0|       916.0|       AS|Alaska Airlines Inc.|
+----+-----+---+-------------+-------------+-------------------+----------------+------------+---------+--------------------+
only showing top 10 rows
```

- **Vérification du schéma du dataframe flights_pret**

```
Schéma du dataframe Flights

root
 |-- YEAR: integer (nullable = true)
 |-- MONTH: integer (nullable = true)
 |-- DAY: integer (nullable = true)
 |-- FLIGHT_NUMBER: integer (nullable = true)
 |-- ORIGIN_AIRPORT: string (nullable = true)
 |-- DESTINATION_AIRPORT: string (nullable = true)
 |-- SCHEDULED_ARRIVAL: float (nullable = true)
 |-- ARRIVAL_TIME: float (nullable = true)
 |-- IATA_CODE: string (nullable = true)
 |-- AIRLINE: string (nullable = true)
```

- **Affichage des trois compagnies ayant le plus faible taux de retard**

```
Les trois compagnies ayant le moins de retard

+--------------------+--------------+
|           compagnie|Taux de retard|
+--------------------+--------------+
|Delta Air Lines Inc.|         28.25|
|Alaska Airlines Inc.|         32.33|
|American Airlines...|         34.46|
+--------------------+--------------+
```

- **Calcul du nombre de vols effectués par ces compagnies dans chacune des destinations et affichage (des dix premiers et dix derniers étant donné le nombre de lignes)**

```
Nombre de vols par compagnie par mois et par destination des trois compagnies avec le plus faible taux de retard

+-----+--------------------+-------------------+-----+
|MONTH|             AIRLINE|DESTINATION_AIRPORT|count|
+-----+--------------------+-------------------+-----+
|    1|Delta Air Lines Inc.|                ABE|   23|
|    1|Delta Air Lines Inc.|                ABQ|   66|
|    1|American Airlines...|                ABQ|  123|
|    1|Alaska Airlines Inc.|                ABQ|   31|
|    1|Alaska Airlines Inc.|                ADK|    9|
|    1|Alaska Airlines Inc.|                ADQ|   29|
|    1|Delta Air Lines Inc.|                AGS|   39|
|    1|Delta Air Lines Inc.|                ALB|   59|
|    1|Alaska Airlines Inc.|                ANC|  987|
|    1|Delta Air Lines Inc.|                ANC|  114|
+-----+--------------------+-------------------+-----+
only showing top 10 rows
```

```
Nombre de vols par compagnie par mois et par destination des trois compagnies avec le plus faible taux de retard

+-----+--------------------+-------------------+-----+
|MONTH|             AIRLINE|DESTINATION_AIRPORT|count|
+-----+--------------------+-------------------+-----+
|   12|Delta Air Lines Inc.|                ABE|   18|
|   12|Alaska Airlines Inc.|                ABQ|   31|
|   12|Delta Air Lines Inc.|                ABQ|  127|
|   12|American Airlines...|                ABQ|  134|
|   12|Alaska Airlines Inc.|                ADK|    6|
|   12|Alaska Airlines Inc.|                ADQ|   26|
|   12|Delta Air Lines Inc.|                AGS|   35|
|   12|American Airlines...|                ALB|   82|
|   12|Delta Air Lines Inc.|                ALB|  129|
|   12|Delta Air Lines Inc.|                ANC|  117|
+-----+--------------------+-------------------+-----+
```

- **Calcul du nombre de vols effectués par ces compagnies durant l'année, ainsi que le nombre de destinations distinctes desservies**

```
Le nombre de vols enregistres par compagnie durant l'année, ainsi que le nombre de destination différente désservie

+--------------------+--------+----------------+
|             AIRLINE|Nbre_Vol|Nbre_Destinations|
+--------------------+--------+----------------+
|Delta Air Lines Inc.|  871946|             299|
|Alaska Airlines Inc.|  171692|             130|
|American Airlines...|  714855|             187|
+--------------------+--------+----------------+
```

V. Conclusion

Rendus au terme de ce projet de session 1, nous avons pu mettre en pratique les techniques de traitement parallèle vu en cours. En effet, en passant par les étapes de prétraitement d'un jeu de données sélectionné par nos soins, nous avons effectué d'une part un traitement en parallèle dans HDFS avec le paradigme MapReduce, et d'autre part, un autre traitement avec l'utilisation de la librairie pyspark qui vient étendre les fonctionnalités de spark.