```
!pip install pyspark

     Collecting pyspark
       Downloading pyspark-3.2.1.tar.gz (281.4 MB)
          |████████████████████████████████| 281.4 MB 33 kB/s
     Collecting py4j==0.10.9.3
       Downloading py4j-0.10.9.3-py2.py3-none-any.whl (198 kB)
          |████████████████████████████████| 198 kB 50.7 MB/s
     Building wheels for collected packages: pyspark
       Building wheel for pyspark (setup.py) ... done
       Created wheel for pyspark: filename=pyspark-3.2.1-py2.py3-none-any.whl size=281853642 sha256=9439b3d7c46701a485e2a84363ba21d7
       Stored in directory: /root/.cache/pip/wheels/9f/f5/07/7cd8017084dce4e93e84e92efd1e1d5334db05f2e83bcef74f
     Successfully built pyspark
     Installing collected packages: py4j, pyspark
     Successfully installed py4j-0.10.9.3 pyspark-3.2.1
```

```
import csv
import json

import matplotlib.path as mplPath
from pyspark.sql import SparkSession
from pyspark.sql.functions import udf, col, mean
from pyspark.sql.types import StringType, StructType, FloatType, IntegerType
from datetime import datetime


spark·=·SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

dfArrondissement·=·spark.read.option("header",·True)·\
··.option("delimiter",·';')·\
··.csv("/content/sample_data/arrondissements.csv")·\
··#·.schema(schema)


dfArrondissement.printSchema()
dfArrondissement.show()
```

```
print(dfArrondissement.head())
```

```
root
 |-- Identifiant séquentiel de l’arrondissement: string (nullable = true)
 |-- Numéro d’arrondissement: string (nullable = true)
 |-- Numéro d’arrondissement INSEE: string (nullable = true)
 |-- Nom de l’arrondissement: string (nullable = true)
 |-- Nom officiel de l’arrondissement: string (nullable = true)
 |-- N_SQ_CO: string (nullable = true)
 |-- Surface: string (nullable = true)
 |-- Périmètre: string (nullable = true)
 |-- Geometry X Y: string (nullable = true)
 |-- Geometry: string (nullable = true)

+------------------------------------------+-----------------------+-----------------------------+-----------------------+-----
|Identifiant séquentiel de l’arrondissement|Numéro d’arrondissement|Numéro d’arrondissement INSEE|Nom de l’arrondissement|Nom c
+------------------------------------------+-----------------------+-----------------------------+-----------------------+-----
|                                 750000006|                      6|                        75106|              6ème Ardt|
|                                 750000009|                      9|                        75109|              9ème Ardt|
|                                 750000020|                     20|                        75120|             20ème Ardt|
|                                 750000018|                     18|                        75118|             18ème Ardt|
|                                 750000002|                      2|                        75102|              2ème Ardt|
|                                 750000010|                     10|                        75110|             10ème Ardt|
|                                 750000001|                      1|                        75101|               1er Ardt|
|                                 750000007|                      7|                        75107|              7ème Ardt|
|                                 750000008|                      8|                        75108|              8ème Ardt|
|                                 750000003|                      3|                        75103|              3ème Ardt|
|                                 750000011|                     11|                        75111|             11ème Ardt|
|                                 750000014|                     14|                        75114|             14ème Ardt|
|                                 750000012|                     12|                        75112|             12ème Ardt|
|                                 750000004|                      4|                        75104|              4ème Ardt|
|                                 750000016|                     16|                        75116|             16ème Ardt|
|                                 750000019|                     19|                        75119|             19ème Ardt|
|                                 750000013|                     13|                        75113|             13ème Ardt|
|                                 750000017|                     17|                        75117|             17ème Ardt|
|                                 750000015|                     15|                        75115|             15ème Ardt|
|                                 750000005|                      5|                        75105|              5ème Ardt|
+------------------------------------------+-----------------------+-----------------------------+-----------------------+-----

Row(Identifiant séquentiel de l’arrondissement='750000006', Numéro d’arrondissement='6', Numéro d’arrondissement INSEE='75106',
def my_function(geo_point_2d_array_column):
  point = (float(geo_point_2d_array_column.split(",")[0]), float(geo_point_2d_array_column.split(",")[1]))
  with open('/content/sample_data/arrondissements.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=';')
    for i, row in enumerate(csv_reader):
      if i > 0:
        j = str(row[9])
        t = json.loads(j)
        c = t["coordinates"][0]
        poly = [[i[1], i[0]] for i in c]
        poly_path = mplPath.Path(poly)
        if poly_path.contains_point(point) == True:
          return int(row[1])
  return 0


def my_function2(kColumn) :
  if kColumn is None:
    return "Inconnu"
  if (kColumn >= 0 and kColumn < 15):
    return "Fluide"
  if (kColumn >= 15 and kColumn < 30):
    return "Pré-saturé"
  if (kColumn >= 30 and kColumn < 50):
    return "Pré-saturé"
  else:
    return "Bloqué"


udf_arrondissement·=·udf(lambda·x:·my_function(x),·IntegerType())
udf_etat_trafic·=·udf(lambda·x:·my_function2(x),·StringType())

schema·=·StructType()·\
··.add("iu_ac",·StringType(),·True)·\
··.add("libelle",·StringType(),·True)·\
··.add("t_1h",·StringType(),·True)·\
··.add("q",·StringType(),·True)·\
··.add("q",·StringType(),·True)·\
··.add("k",·StringType(),·True)·\
··.add("etat_trafic",·StringType(),·True)·\
··.add("iu_nd_amont",·StringType(),·True)·\
··.add("libelle_nd_amont",·StringType(),·True)·\
··.add("iu_nd_aval",·StringType(),·True)·\
··.add("libelle_nd_aval",·StringType(),·True)·\
··.add("etat_barre",·StringType(),·True)·\
··.add("date_debut",·StringType(),·True)·\
··.add("date_fin",·StringType(),·True)·\
··.add("geo_point_2d",·StringType(),·True)·\
··.add("geo_shape",·StringType(),·True)




df·=·spark.read.options(header='True',·delimiter=';')·\
··.schema(schema=schema)·\
··.csv("/content/sample_data/data-2021-01-02.csv")

df.printSchema()
df.show()
df.withColumn("arrondissement",udf_arrondissement(col("geo_point_2d")))·\
··.groupBy("arrondissement",·"t_1h")·\
··.agg(mean(col("k")).alias("K"))·\
··.withColumn("etat_trafic",·udf_etat_trafic(col("K")))·\
··.orderBy(col("arrondissement").asc(),·col("t_1h").asc())·\
··.write.option("header",True)·\
··.option("delimiter",";")·\
··.csv("/content/sample_data/sortie3")·
··
··
print(df.head())
df.printSchema()

print("FINISH")
```

```
 |-- k: string (nullable = true)
 |-- etat_trafic: string (nullable = true)
 |-- iu_nd_amont: string (nullable = true)
 |-- libelle_nd_amont: string (nullable = true)
 |-- iu_nd_aval: string (nullable = true)
 |-- libelle_nd_aval: string (nullable = true)
 |-- etat_barre: string (nullable = true)
 |-- date_debut: string (nullable = true)
 |-- date_fin: string (nullable = true)
 |-- geo_point_2d: string (nullable = true)
 |-- geo_shape: string (nullable = true)

+-----+----------------+--------------------+----+----+-----------+-----------+--------------------+----------+------------
|iu_ac|         libelle|                t_1h|   q|   k|etat_trafic|iu_nd_amont|    libelle_nd_amont|iu_nd_aval|     libelle
+-----+----------------+--------------------+----+----+-----------+-----------+--------------------+----------+------------
| 4654|    Pte_St_Cloud|2021-01-02T15:00:...|null|null|    Inconnu|       2477|Pte_St_Cloud-Mura...|      2722|Pte_St_Cloud
| 4654|    Pte_St_Cloud|2021-01-02T01:00:...|null|null|    Inconnu|       2477|Pte_St_Cloud-Mura...|      2722|Pte_St_Cloud
| 4654|    Pte_St_Cloud|2021-01-02T10:00:...|null|null|    Inconnu|       2477|Pte_St_Cloud-Mura...|      2722|Pte_St_Cloud
| 4654|    Pte_St_Cloud|2021-01-02T08:00:...|null|null|    Inconnu|       2477|Pte_St_Cloud-Mura...|      2722|Pte_St_Cloud
| 4654|    Pte_St_Cloud|2021-01-02T04:00:...|null|null|    Inconnu|       2477|Pte_St_Cloud-Mura...|      2722|Pte_St_Cloud
| 6375|       AI_Lafont|2021-01-02T16:00:...|null|28.3| Pré-saturé|       3304|         AI_Lafont-1|      2723|   AI_George
| 6375|       AI_Lafont|2021-01-02T22:00:...|null|2.45|     Fluide|       3304|         AI_Lafont-1|      2723|   AI_George
| 6375|       AI_Lafont|2021-01-02T11:00:...|null| 8.9|     Fluide|       3304|         AI_Lafont-1|      2723|   AI_George
| 6375|       AI_Lafont|2021-01-02T17:00:...|null|13.1|     Fluide|       3304|         AI_Lafont-1|      2723|   AI_George
| 6375|       AI_Lafont|2021-01-02T21:00:...|null|6.95|     Fluide|       3304|         AI_Lafont-1|      2723|   AI_George
| 6375|       AI_Lafont|2021-01-02T09:00:...|null| 4.4|     Fluide|       3304|         AI_Lafont-1|      2723|   AI_George
| 6375|       AI_Lafont|2021-01-02T08:00:...|null|2.95|     Fluide|       3304|         AI_Lafont-1|      2723|   AI_George
| 6375|       AI_Lafont|2021-01-02T04:00:...|null| 0.6|     Fluide|       3304|         AI_Lafont-1|      2723|   AI_George
| 6375|       AI_Lafont|2021-01-02T10:00:...|null|7.35|     Fluide|       3304|         AI_Lafont-1|      2723|   AI_George
| 6375|       AI_Lafont|2021-01-02T06:00:...|null| 1.6|     Fluide|       3304|         AI_Lafont-1|      2723|   AI_George
| 5449|PI_Georges_Lafont|2021-01-02T19:00:...|null|null|    Inconnu|       2724|         SI_St_Cloud|      2723|   AI_George
| 5449|PI_Georges_Lafont|2021-01-02T18:00:...|null|null|    Inconnu|       2724|         SI_St_Cloud|      2723|   AI_George
| 5449|PI_Georges_Lafont|2021-01-02T15:00:...|null|null|    Inconnu|       2724|         SI_St_Cloud|      2723|   AI_George
| 5449|PI_Georges_Lafont|2021-01-02T13:00:...|null|null|    Inconnu|       2724|         SI_St_Cloud|      2723|   AI_George
| 5449|PI_Georges_Lafont|2021-01-02T20:00:...|null|null|    Inconnu|       2724|         SI_St_Cloud|      2723|   AI_George
+-----+----------------+--------------------+----+----+-----------+-----------+--------------------+----------+------------
only showing top 20 rows

Row(iu_ac='4654', libelle='Pte_St_Cloud', t_1h='2021-01-02T15:00:00+00:00', q=None, k=None, etat_trafic='Inconnu', iu_nd_amo
root
 |-- iu_ac: string (nullable = true)
 |-- libelle: string (nullable = true)
 |-- t_1h: string (nullable = true)
 |-- q: string (nullable = true)
 |-- K: string (nullable = true)
 |-- etat_trafic: string (nullable = true)
 |-- iu_nd_amont: string (nullable = true)
 |-- libelle_nd_amont: string (nullable = true)
 |-- iu_nd_aval: string (nullable = true)
 |-- libelle_nd_aval: string (nullable = true)
 |-- etat_barre: string (nullable = true)
 |-- date_debut: string (nullable = true)
 |-- date_fin: string (nullable = true)
 |-- geo_point_2d: string (nullable = true)
 |-- geo_shape: string (nullable = true)

FINISH
```

```
def my_function3(t1hColumn) :
    print(t1hColumn)
    datetimeAux = t1hColumn
    datetime_obj = datetime.strptime(datetimeAux, "%Y-%m-%dT%H:%M:%S+00:00").strftime("%H:%M:%S")
    return str(datetime_obj)


udf_heure = udf(lambda x: my_function3(x), StringType())


schema2 = StructType() \
      .add("arrondissement", IntegerType(), True) \
      .add("t_1h", StringType(), True) \
      .add("K", StringType(), True) \
      .add("etat_trafic", StringType(), True) \

dfFinal = spark.read.options(header='True', delimiter=';') \
      .schema(schema=schema2) \
      .csv("/content/sample_data/sortie3/part-00000-99ff3e83-febd-4418-9698-bffe471bf6e0-c000.csv")



dfFinal.withColumn("time", udf_heure(col("t_1h"))) \
      .groupBy("arrondissement", "time") \
      .agg(mean(col("K")).alias("K")) \
      .withColumn("etat_trafic", udf_etat_trafic(col("K"))) \
      .orderBy(col("arrondissement").asc(), col("time").asc()) \
      .show()
```

```
+--------------+--------+-----------------+-----------+
|arrondissement|    time|                K|etat_trafic|
+--------------+--------+-----------------+-----------+
|            16|00:00:00|             null|    Inconnu|
|            16|01:00:00|              0.9|     Fluide|
|            16|02:00:00|            0.425|     Fluide|
|            16|03:00:00|             null|    Inconnu|
|            16|04:00:00|0.774999999999999|     Fluide|
|            16|05:00:00|             null|    Inconnu|
|            16|06:00:00|            1.925|     Fluide|
|            16|07:00:00|             3.05|     Fluide|
|            16|08:00:00|             2.95|     Fluide|
|            16|09:00:00|              5.1|     Fluide|
|            16|10:00:00| 8.149999999999999|     Fluide|
|            16|11:00:00|            9.575|     Fluide|
|            16|12:00:00|            11.85|     Fluide|
|            16|13:00:00|             null|    Inconnu|
|            16|14:00:00|             12.2|     Fluide|
|            16|15:00:00|             14.6|     Fluide|
|            16|16:00:00|             28.3| Pré-saturé|
|            16|17:00:00|             12.1|     Fluide|
|            16|18:00:00|             null|    Inconnu|
|            16|19:00:00|             11.1|     Fluide|
+--------------+--------+-----------------+-----------+
only showing top 20 rows
```

```
print(datetime.strptime("2021-12-14T10:00:00+00:00", "%Y-%m-%dT%H:%M:%S+00:00").strftime("%H:%M:%S"))
```

```
10:00:00
```

✓ 0 s terminée à 20:40 ● ✕