```
# 此内容为代码格式
```

révision: 26/09/2024

Auteur :

runlin Zhou 28717281

zhile Zhang 21201131

TP 2024 Fenêtres OVER,

Sujet seul sans solutions

- **Données**: Foursquare
- **Moteur SQL**: DuckDB

Préparation

+ 代码   + 文本

Lire la documentation DuckDB SQL :

- Functions
- Window functions

Affichage convivial des résultats d'une requête

```
pip install -q itables
```

```
                    1.4/1.4 MB 13.1 MB/s eta 0:00:0
```

```
# pour améliorer l'affichage des tableaux contenant le resultat d'une requete
from itables import init_notebook_mode
init_notebook_mode(all_interactive=True)
```

```
import pandas as pd

def display(query, n=30):
    return query.limit(n).df()

# seulement dans colab
# from google.colab import data_table
# def display(query, n=100):
#     return data_table.DataTable(query.limit(n).df(), include_index=False, num
```

```
!pip install duckdb
```

```
Requirement already satisfied: duckdb in /usr/local/lib/python3.10/dist-pac
```

Définir le tag **%%sql** pour pouvoir écrire plus simplement des requêtes en SQL dans une cellule

```
from IPython.core.magic import (register_line_magic, register_cell_magic, regis

def removeComments(query):
  result = ""
  for line in query.split('\n'):
    if not(line.strip().startswith("--")):
      result += line + "\n"
  return result

@register_line_cell_magic
def sql(line, cell=None):
    "To run a sql query. Use:  %%sql"
    val = cell if cell is not None else line
    tabRequetes = removeComments(val).split(";")
    derniere = None
    est_requete = False
    for r in tabRequetes:
        r = r.strip()
        if len(r) > 2:
          derniere = db.sql(r)
          est_requete = ( r.lower().startswith('select')or r.lower().startswith
    if(est_requete):
      return display(derniere)
    else:
      return print('ok')

print("fonction définie")
```

```
fonction définie
```

Démarrer le service DuckDB

```
import duckdb

# db = duckdb.connect('foursquare')
db = duckdb.connect(':memory:')

# vérifier que le service fonctrionne
r = db.sql("SELECT 'hello' as col")
display(r)
```

col
hello

```
def display_schema(table):
  return display(db.sql(f"""
    SELECT column_name, data_type
    FROM duckdb_columns
    WHERE table_name = '{table}'
  """))

def display_all_views():
  return display(db.sql("select view_name, sql, temporary from duckdb_views"))

print("fonctions définies")
```

```
fonctions définies
```

Premier exemple

Un premier exemple pour comprendre les fonctions sur des fenêtres

Les données

Les visites

```
visite_attrs = ["photoId",  "personId", "date", "lon",  "lat", "note"]
visite_tuples = [("p1", "Bob",  "03/09/2020",   41.5,   12.8, 3),
                 ("p2", "Alice", "01/09/2020",  41.6,   12.8, 5),
                 ("p3", "Bob",  "04/09/2020",   41.6,   13.2, 1),
                 ("p4", "Bob",  "04/09/2020",   40.1,   12.0, 2),
                 ("p5", "Alice", "04/09/2020",  40.1,   12.0, 2),
                 ("p6", "Alice", "05/09/2020",  41.6,   12.7, 4),
                 ("p7", "Alice", "05/10/2020",  41.8, 12.8, 4),
                 ("p8", "Carole", "25/12/2019", 30.1, 10.1, 2),
                 ("p9", "David", "25/12/2019",  30.1, 10.1, 1),
                 ("p10","Eva", "25/12/2019",    30.1, 10.1, 5),
                 ("p11","Eva", "26/12/2019",    31.1, 10.1, 3),
                 ("p12","Alice", "01/02/2020",  32.1, 12.1, 3),
                 ("p13","Bob", "01/02/2020",    32.1, 12.1, 5),
                 ("p14","Carole", "01/02/2020", 32.1, 12.1, 2),
                 ("p15",    "Carole", "11/11/2019", 49.1, 10.1, 1),
                 ("p16",    "Alice", "25/12/2019",  30.1, 10.1, 4),
                 ("p17",    "Alice", "02/02/2020",  49.1, 10.1, 5),
                ]

visites_df = pd.DataFrame(visite_tuples, columns=visite_attrs)
```

```
db.sql("""
DROP TABLE IF EXISTS Visites CASCADE;

CREATE TABLE Visites AS
SELECT photoId,
       personId,
       strptime(date, '%d/%m/%Y') as date_num,
       extract(year from date_num) as année,
       extract(month from date_num) as mois,
       lon,
      lat,
      note
FROM visites_df""")

Visites = db.table("Visites")

display(Visites)
```

10 entries per page   Search:

photoId	personId	date_num	année	mois	lon	lat	note
p1	Bob	2020-09-03	2020	9	41.5	12.8	3
p2	Alice	2020-09-01	2020	9	41.6	12.8	5
p3	Bob	2020-09-04	2020	9	41.6	13.2	1
p4	Bob	2020-09-04	2020	9	40.1	12	2
p5	Alice	2020-09-04	2020	9	40.1	12	2
p6	Alice	2020-09-05	2020	9	41.6	12.7	4
p7	Alice	2020-10-05	2020	10	41.8	12.8	4
p8	Carole	2019-12-25	2019	12	30.1	10.1	2
p9	David	2019-12-25	2019	12	30.1	10.1	1
p10	Eva	2019-12-25	2019	12	30.1	10.1	5

Showing 1 to 10 of 17 entries   « ‹ 1 2 › »

```
display_schema("Visites")
```

column_name	data_type
photoId	VARCHAR
personId	VARCHAR
date_num	TIMESTAMP
année	BIGINT
mois	BIGINT
lon	DOUBLE
lat	DOUBLE
note	BIGINT

```
# # exemple de cellule avec requete SQL

# %%sql
# SELECT *
# FROM Visites
```

Les lieux

```
lieux_attributs = ["lon","lat", "ville", "pays"]

lieux_tuples = [(41.5, 12.8, "Aix", "France"),
                (41.6, 12.8, "Aix", "France"),
                (41.6, 12.7, "Nice", "France"),
                (41.8, 12.8, "Marseille", "France"),
                (41.6, 13.2, "Rome", "Italie"),
                (40.1, 12.0, "Oslo", "Norvege"),
                (30.1, 10.1, "Paris", "France"),
                (31.1, 10.1, "StDenis", "France"),
                (32.1, 12.1, "Lille", "France"),
                (49.1, 10.1, "Pau", "France") ]


lieux_df = pd.DataFrame(lieux_tuples, columns=lieux_attributs)
db.sql("""
drop table if exists Lieux;
create table Lieux as
select ville, pays, lon, lat from lieux_df""")

Lieux = db.table("Lieux")

display(Lieux)
```

ville	pays	lon	lat
Aix	France	41.5	12.8
Aix	France	41.6	12.8
Nice	France	41.6	12.7
Marseille	France	41.8	12.8
Rome	Italie	41.6	13.2
Oslo	Norvege	40.1	12
Paris	France	30.1	10.1
StDenis	France	31.1	10.1
Lille	France	32.1	12.1
Pau	France	49.1	10.1

Les professions

```
profession_attributs = ["personID","profession"]

profession_tuples = [("Alice", "Architecte"),
                     ("Bob", "Data science"),
                     ("Carole", "Data science"),
                     ("David", "Architecte"),
                     ("Eva", "Vendeuse"),
                     ("Franck", "Vendeur"),
                     ("Greta", "Economiste")
                    ]

profession_df = pd.DataFrame(profession_tuples, columns=profession_attributs)
db.sql("""
drop table if exists Professions;
create table Professions as
select * from profession_df""")

Professions = db.table("Professions")

display(Professions)
```

personID	profession
Alice	Architecte
Bob	Data science
Carole	Data science
David	Architecte
Eva	Vendeuse
Franck	Vendeur
Greta	Economiste

Le détail des visites

Définir une vue contenant le détail des visites

```
db.sql("""
CREATE OR REPLACE VIEW VisitesDetail AS

SELECT v.photoID, p.personID, p.profession, v.date_num, v.mois, v.année, v.lon,
FROM Visites v
JOIN Lieux l ON (v.lon = l.lon AND v.lat = l.lat )
JOIN Professions p ON (v.personId = p.personId)
ORDER BY pays, ville
""")

# Même requête mais écrite avec un NATURAL JOIN
# visitesDetail = db.sql("""
# CREATE OR REPLACE VIEW VisitesDetail AS
# SELECT v.photoID, p.personID, p.profession, v.date_num, v.mois, v.année, v.lc
# FROM Visites v NATURAL JOIN Lieux l NATURAL JOIN Professions p
# ORDER BY pays, ville
# """)

VisitesDetail=db.view('VisitesDetail')

display(VisitesDetail)
```

10 entries per page   Search:

photoId	personID	profession	date_num	mois	année	lon
p1	Bob	Data science	2020-09-03	9	2020	41.5
p2	Alice	Architecte	2020-09-01	9	2020	41.6
p12	Alice	Architecte	2020-02-01	2	2020	32.1
p13	Bob	Data science	2020-02-01	2	2020	32.1
p14	Carole	Data science	2020-02-01	2	2020	32.1
p7	Alice	Architecte	2020-10-05	10	2020	41.8
p6	Alice	Architecte	2020-09-05	9	2020	41.6
p8	Carole	Data science	2019-12-25	12	2019	30.1
p9	David	Architecte	2019-12-25	12	2019	30.1
p10	Eva	Vendeuse	2019-12-25	12	2019	30.1

Showing 1 to 10 of 17 entries   « ‹ 1 2 › »

Requete avec fenêtre de taille croissante

Numéroter les tuples

```
r = db.sql("""
SELECT v.personId, v.photoId, v.note,
       row_number() OVER (ORDER BY personId, photoId) as numeroPhoto
FROM Visites v
""")

display(r)
```

10 entries per page   Search:

personId	photoId	note	numeroPhoto
Alice	p12	3	1
Alice	p16	4	2
Alice	p17	5	3
Alice	p2	5	4
Alice	p5	2	5
Alice	p6	4	6
Alice	p7	4	7
Bob	p1	3	8
Bob	p13	5	9
Bob	p3	1	10

Showing 1 to 10 of 17 entries   « ‹ 1 2 › »

Classement des visites par note décroissantes

```
r = db.sql("""
SELECT v.personId, v.photoId, v.note, rank() over (order by note desc) as rang
FROM Visites v
""")
display(r)
```

10 entries per page Search:

personId	photoId	note	rang
Alice	p2	5	1
Eva	p10	5	1
Bob	p13	5	1
Alice	p17	5	1
Alice	p6	4	5
Alice	p7	4	5
Alice	p16	4	5
Bob	p1	3	8
Eva	p11	3	8
Alice	p12	3	8

Showing 1 to 10 of 17 entries « ‹ 1 2 › »

Requête Top K

Le top 5 des Visites avec les meilleures notes. Il peut y avoir plus de k visites dans le résultat à cause des **ex aequos**.

```
r=db.sql("""
WITH Classement as (
  SELECT v.personId,
         v.photoId, v.note,
         rank() over (order by note desc) as rang
  FROM Visites v
)

SELECT *
FROM Classement
WHERE rang <=5
ORDER BY rang
""")

display(r)
```

personId	photoId	note	rang
Alice	p2	5	1
Eva	p10	5	1
Bob	p13	5	1
Alice	p17	5	1
Alice	p6	4	5
Alice	p7	4	5
Alice	p16	4	5

Classement dense

```
r=db.sql("""
SELECT  v.personId,
        v.photoId,
        v.note,
        dense_rank() over (order by note desc) as rang_dense
FROM Visites v
""")

display(r)
```

10 entries per page   Search:

personId	photoId	note	rang_dense
Alice	p2	5	1
Eva	p10	5	1
Bob	p13	5	1
Alice	p17	5	1
Alice	p6	4	2
Alice	p7	4	2
Alice	p16	4	2
Bob	p1	3	3
Eva	p11	3	3
Alice	p12	3	3

Showing 1 to 10 of 17 entries « ‹ 1 2 › »

Fenêtrage et partitionnement : Partition by

Une partition est définie par un sous-ensemble des attributs projetés dans le select.

Une fenêtre par partition.

Le classement **par ville** des visites les mieux notées

```
%%sql

SELECT v.photoId, v.personID, v.ville,
       rank() OVER( PARTITION BY ville ORDER BY note) as rang

FROM VisitesDetail v

ORDER BY ville, rang
```

10 entries per page   Search:

photoId	personID	ville	rang
p1	Bob	Aix	1
p2	Alice	Aix	2
p14	Carole	Lille	1
p12	Alice	Lille	2
p13	Bob	Lille	3
p7	Alice	Marseille	1
p6	Alice	Nice	1
p4	Bob	Oslo	1
p5	Alice	Oslo	1
p9	David	Paris	1

Showing 1 to 10 of 17 entries « ‹ 1 2 › »

Le classement **par pays** des villes ayant de plus de visites

```
%%sql
SELECT pays, ville, count(*) as nbVisite,
       rank() over (partition by pays order by count(*) desc) as rang
From VisitesDetail v
Group by pays, ville
Order by pays, rang
```

pays	ville	nbVisite	rang
France	Paris	4	1
France	Lille	3	2
France	Pau	2	3
France	Aix	2	3
France	Nice	1	5
France	Marseille	1	5
France	StDenis	1	5
Italie	Rome	1	1
Norvege	Oslo	2	1

Questions

Question 1 : temps écoulé

Pour chaque visite de chaque personne, le temps écoulé depuis la **première** visite de cette personne. Exprimer le temps écoulé par un intervalle de temps résultant d'une différence entre deux dates. L'intervalle indique le nombre de jours.

Le schema attendu est (photoID , personID, date, datePremiereVisiste, jours_ecoules)

Astuce: commencer par compléter chaque visite avec la date de la première visite de la personne.

```
%%sql
-- solution dans TP2_solutions
with T as (
  select  photoID, personID, date_num, min(date_num) over (partition by personI
  from Visites
)

SELECT photoID, personID, date_num, minD, date_num-minD as jours_ecoules
FROM T
ORDER BY personID, date_num

-- AUTRE solution avec datediff
--SELECT photoID, personID, date_num, minD, datediff('day', minD, date_num) as
--FROM T
--ORDER BY personID, date_num
```

10 entries per page   Search:

photoId	personId	date_num	minD	jours_ecoules
p16	Alice	2019-12-25	2019-12-25	0 days
p12	Alice	2020-02-01	2019-12-25	38 days
p17	Alice	2020-02-02	2019-12-25	39 days
p2	Alice	2020-09-01	2019-12-25	251 days
p5	Alice	2020-09-04	2019-12-25	254 days
p6	Alice	2020-09-05	2019-12-25	255 days
p7	Alice	2020-10-05	2019-12-25	285 days
p13	Bob	2020-02-01	2020-02-01	0 days
p1	Bob	2020-09-03	2020-02-01	215 days
p3	Bob	2020-09-04	2020-02-01	216 days

Showing 1 to 10 of 17 entries   « ‹ 1 2 › »

Question 2: Requête avec fenêtre glissante de taille fixe

Pour chaque visite de chaque personne, le nombre de jours depuis la visite **précédente** de cette personne.

Le schéma attendu est (photoID, personID, date, precedenteDate, nbJours)

Astuce: penser à utiliser la fonction datediff pour comparer deux dates et obtenir un nombre de jours

```
%%sql
WITH
V2 as (
select  photoID , personID, date_num, LAG(date_num, 1, date_num) OVER (PARTITIC
from Visites
)

select photoID, personID, date_num, precedenteDate, datediff('day', precedenteD
from V2
order by personID, date_num
```

10 entries per page   Search:

photoId	personId	date_num	precedenteDate	nbJours
p16	Alice	2019-12-25	2019-12-25	0
p12	Alice	2020-02-01	2019-12-25	38
p17	Alice	2020-02-02	2020-02-01	1
p2	Alice	2020-09-01	2020-02-02	212
p5	Alice	2020-09-04	2020-09-01	3
p6	Alice	2020-09-05	2020-09-04	1
p7	Alice	2020-10-05	2020-09-05	30
p13	Bob	2020-02-01	2020-02-01	0
p1	Bob	2020-09-03	2020-02-01	215
p3	Bob	2020-09-04	2020-09-03	1

Showing 1 to 10 of 17 entries   « ‹ 1 2 › »

Question 3 : Semaine glissante

Pour chaque jour de visite de chaque personne, le nombre de visite sur 7 jours glissants.

Schéma: (personId, date_num, nbVisites_depuis_7jours)

```
%%sql
WITH T as (
  SELECT personId, date_num, count(*) as nb_visites
  FROM Visites
  GROUP BY personId, date_num
  ORDER BY nb_visites desc
)

SELECT  personID, date_num, sum(nb_visites) over (partition by personID order b
FROM T
ORDER BY personID, date_num
```

10 entries per page   Search:

personId	date_num	nbVisites_depuis_7jours
Alice	2019-12-25	1
Alice	2020-02-01	1
Alice	2020-02-02	2
Alice	2020-09-01	1
Alice	2020-09-04	2
Alice	2020-09-05	3
Alice	2020-10-05	1
Bob	2020-02-01	1
Bob	2020-09-03	1
Bob	2020-09-04	3

Showing 1 to 10 of 16 entries   « ‹ 1 2 › »

Question 4 : Trajets

On veut enrichir les visites avec une information de trajet. Un trajet représente toutes les visites d'une personne telles que le nombre de jours entre deux visites consécutives soit inférieur à une semaine.

Schema: (photoID, personID, date_num, nb_jours, trajet)

Astuces : utiliser la réponse à la question 2, la fonction if dans la clause select peut indiquer si la visite est le début du trajet suivant ou non.

```
%%sql
with T1 as (
select  photoID , personID, date_num, LAG(date_num, 1, date_num) OVER (PARTITIC
from Visites
order by personID, date_num),

T2 as (
select photoID, personID, date_num, datediff('day', precedenteDate, date_num) a
from T1
order by personID, date_num),

T3 as (
select photoID, personID, date_num, nb_jours, if(nb_jours > 7, 1, 0) as plus1
from T2
)

select photoID, personID, date_num, nb_jours, sum(plus1) over (partition by per
from T3
order by personID, date_num
```

10 entries per page Search:

photoId	personId	date_num	nb_jours	numTrajet
p16	Alice	2019-12-25	0	1
p12	Alice	2020-02-01	38	2
p17	Alice	2020-02-02	1	2
p2	Alice	2020-09-01	212	3
p5	Alice	2020-09-04	3	3
p6	Alice	2020-09-05	1	3
p7	Alice	2020-10-05	30	4
p13	Bob	2020-02-01	0	1
p1	Bob	2020-09-03	215	2
p3	Bob	2020-09-04	1	2

Showing 1 to 10 of 17 entries « ‹ 1 2 › »

4 b)

Ajouter la condition: un trajet ne peut pas durer plus de 3 jours au total

```
%%sql
with T1 as (
select  photoID , personID, date_num, LAG(date_num, 1, date_num) OVER (PARTITIC
from Visites
order by personID, date_num),

T2 as (
select photoID, personID, date_num, datediff('day', precedenteDate, date_num) a
from T1
order by personID, date_num),

T3 as (
select photoID, personID, date_num, nb_jours, if(nb_jours >= 3, 1, 0) as plus1
from T2
)

select photoID, personID, date_num, nb_jours, sum(plus1) over (partition by per
from T3
order by personID, date_num
```

10 entries per page   Search:

photoId	personId	date_num	nb_jours	numTrajet
p16	Alice	2019-12-25	0	1
p12	Alice	2020-02-01	38	2
p17	Alice	2020-02-02	1	2
p2	Alice	2020-09-01	212	3
p5	Alice	2020-09-04	3	4
p6	Alice	2020-09-05	1	4
p7	Alice	2020-10-05	30	5
p13	Bob	2020-02-01	0	1
p1	Bob	2020-09-03	215	2
p3	Bob	2020-09-04	1	2

Showing 1 to 10 of 17 entries   « ‹ 1 2 › »

Données Foursquare

Données issues du réseau social Foursquare

```
import os
# local_dir = "/local/data"
local_dir = os.environ["HOME"] + "/data"
print("local_dir is", local_dir)
os.makedirs(local_dir, exist_ok=True)
os.listdir(local_dir)
```

```
local_dir is /root/data
[]
```

URL pour l'accès aux datasets

```
# ---------------------------------------------------------------------------
# en cas de problème avec le téléchargement des datasets, aller directement sur
PUBLIC_DATASET_URL = "https://nuage.lip6.fr/s/LqD9N23kxrfHopr"
PUBLIC_DATASET=PUBLIC_DATASET_URL + "/download?path="

print("URL pour les datasets ", PUBLIC_DATASET_URL)
```

```
URL pour les datasets  https://nuage.lip6.fr/s/LqD9N23kxrfHopr
```

```
import os
from urllib import request
import zipfile

# download dataset if not already donwloaded
def download_file(web_dir, local_dir, file):
  local_file = local_dir + "/" + file
  web_file = web_dir + "/" + file
  if(os.path.isfile(local_file)):
    print(file, "is already stored")
  else:
    print("downloading from URL: ", web_file , "save in : " + local_file)
    request.urlretrieve(web_file , local_file)

def unzip_file(local_dir, file):
  with zipfile.ZipFile(local_dir + "/" + file, 'r') as zip_ref:
    zip_ref.extractall(local_dir)
  # os.remove(local_dir + "/" + file)



web_dir = PUBLIC_DATASET + "/foursquare"

# ce fichier vient de kaggle : https://www.kaggle.com/datasets/chetanism/foursc
zip_filename = "dataset_TSMC2014_NYC.zip"



download_file(web_dir, local_dir, zip_filename)
unzip_file(local_dir, zip_filename)

# Liste des fichiers
os.listdir(local_dir)
```

```
downloading from URL:  https://nuage.lip6.fr/s/LqD9N23kxrfHopr/download?pat
['dataset_TSMC2014_NYC.zip', 'dataset_TSMC2014_NYC.csv']
```

```
# !head -4 /local/data/dataset_TSMC2014_NYC.csv
with open(local_dir + "/dataset_TSMC2014_NYC.csv", 'r') as data_file:
    for i in range(4):
        print(data_file.readline().strip())
```

```
userId,venueId,venueCategoryId,venueCategory,latitude,longitude,timezoneOff
470,49bbd6c0f964a520f4531fe3,4bf58dd8d48988d127951735,Arts & Crafts Store,4
979,4a43c0aef964a520c6a61fe3,4bf58dd8d48988d1df941735,Bridge,40.60679958,-7
69,4c5cc7b485a1e21e00d35711,4bf58dd8d48988d103941735,Home (private),40.7161
```

Table Visit

```
# duckdb.read_csv(f"{local_dir}/notes1M.json")

stmt = f"""
DROP TABLE if exists Visit;
CREATE TABLE Visit as
  SELECT *
  FROM '{local_dir}/dataset_TSMC2014_NYC.csv';
"""
db.sql(stmt)
print("done")
```

```
done
```

Afficher le contenu d'une table

```
visit = db.table('Visit')
visit
```

```
┌────────┬──────────────────────────┬──────────────────────────┬
│ userId │         venueId          │     venueCategoryId      │
venueCategory      │  latitude    │  longitude    │ timezoneOffset │
utcTimestamp            │
│ int64  │         varchar          │         varchar          │
varchar           │  double      │    double     │     int64      │
varchar              │
├────────┼──────────────────────────┼──────────────────────────┼
│    470 │ 49bbd6c0f964a520f4531fe3 │ 4bf58dd8d48988d127951735 │ Arts &
Crafts Store  │ 40.71981038 │ -74.00258103 │           -240 │ Tue Apr 03
18:00:09 +0000 2012 │
│    979 │ 4a43c0aef964a520c6a61fe3 │ 4bf58dd8d48988d1df941735 │ Bridge
│ 40.60679958 │ -74.04416981 │           -240 │ Tue Apr 03 18:00:25 +0000
2012 │
│     69 │ 4c5cc7b485a1e21e00d35711 │ 4bf58dd8d48988d103941735 │ Home
(private)       │ 40.71616168 │ -73.88307006 │           -240 │ Tue Apr
03 18:02:24 +0000 2012 │
│    395 │ 4bc7086715a7ef3bef9878da │ 4bf58dd8d48988d104941735 │ Medical
Center       │  40.7451638 │ -73.98251878 │           -240 │ Tue Apr 03
18:02:41 +0000 2012 │
│     87 │ 4cf2c5321d18a143951b5cec │ 4bf58dd8d48988d1cb941735 │ Food
Truck           │ 40.74010383 │ -73.98965836 │           -240 │ Tue Apr
03 18:03:00 +0000 2012 │
│    484 │ 4b5b981bf964a520900929e3 │ 4bf58dd8d48988d118951735 │ Food &
Drink Shop    │ 40.69042712 │ -73.95468678 │           -240 │ Tue Apr 03
18:04:00 +0000 2012 │
│    642 │ 4ab966c3f964a5203c7f20e3 │ 4bf58dd8d48988d1e0931735 │ Coffee
Shop          │ 40.75159143 │  -73.9741214 │           -240 │ Tue Apr 03
18:04:38 +0000 2012 │
│    292 │ 4d0cc47f903d37041864bf55 │ 4bf58dd8d48988d12b951735 │ Bus
Station          │ 40.77942173 │ -73.95534113 │           -240 │ Tue Apr
03 18:04:42 +0000 2012 │
│    428 │ 4ce1863bc4f6a35d8bd2db6c │ 4bf58dd8d48988d103941735 │ Home
(private)       │ 40.61915107 │  -74.0358876 │           -240 │ Tue Apr
```

```
03 18:06:18 +0000 2012 |
|    877 | 4be319b321d5a59352311811 | 4bf58dd8d48988d10a951735 | Bank
| 40.61900594 | -73.99037473 |           -240 | Tue Apr 03 18:06:19 +0000
2012 |
|      .  |            .            |            .             |  .
|       .      |      .      |            .  |             .
|
|      .  |            .            |            .             |  .
|       .      |      .      |            .  |             .
|
|      .  |            .            |            .             |  .
|       .      |      .      |            .  |             .
|
|    847 | 49fe2f8af964a5207b6f1fe3 | 4bf58dd8d48988d176941735 | Gym /
Fitness Center | 40.72923881 |  -74.0052724 |           -240 | Wed Apr 11
17:04:02 +0000 2012 |
|     45 | 4b735fe3f964a5205bab2de3 | 4d4ae6fc7a7b7dea34424761 | Fried
Chicken Joint  | 40.73245806 |  -73.9851737 |           -240 | Wed Apr 11
17:04:03 +0000 2012 |
|    621 | 4f85b99de4b0d19a2f21e5a1 | 4bf58dd8d48988d177941735 | Medical
Center       | 40.89911233 | -73.97102783 |           -240 | Wed Apr 11
17:04:36 +0000 2012 |
|    537 | 4f6491dce4b05c1d59696fc2 | 4bf58dd8d48988d174941735 | Office
|   40.761782 |   -73.958256 |           -240 | Wed Apr 11 17:05:05 +0000
```

```
display(visit,5)
```

userId	utcTimestamp	venueId	venueCategoryId
975	Fri Apr 13 21:35:37 +0000 2012	4ba61935f964a520673339e3	4bf58dd8d48988d
976	Fri Apr 06 16:33:37 +0000 2012	4f7ecbf9e4b068381402db75	4bf58dd8d48988d
994	Fri Apr 06 13:03:29 +0000 2012	4ab3a2f4f964a520e56d20e3	4bf58dd8d48988d
1011	Fri Apr 13 10:41:56 +0000 2012	4d99e712b188721e88944837	4bf58dd8d48988d
1058	Fri Apr 13 12:24:17 +0000 2012	42829c80f964a5206a221fe3	4bf58dd8d48988d

单元格类型不受支持。双击即可检查/修改内容。

```
%%sql

select userId, utcTimestamp, count(*) as nb_visites, count(distinct venueId) nb
from visit
group by userId, utcTimestamp
having nb_visites > 1
order by nb_visites desc
limit 10
```

userId	utcTimestamp	nb_visites	nb_distinct_poi
739	Thu Apr 19 12:00:12 +0000 2012	4	4
739	Sat Apr 21 12:00:10 +0000 2012	4	4
739	Thu May 03 12:00:10 +0000 2012	4	4
739	Fri May 04 12:00:10 +0000 2012	4	4
739	Mon May 14 12:00:11 +0000 2012	4	4
344	Mon Nov 19 16:38:14 +0000 2012	4	1
739	Sat Apr 28 12:00:10 +0000 2012	4	4
739	Tue May 15 12:00:07 +0000 2012	4	4
315	Tue Jun 26 23:42:59 +0000 2012	4	1
315	Fri Oct 26 05:09:45 +0000 2012	3	1

```
visit_344_exemple = db.sql("""
select *
from visit
where userId = 344
and utcTimestamp = 'Mon Nov 19 16:38:14 +0000 2012'
""")
display(visit_344_exemple)
```

userId	venueId	venueCategoryId	venueCategory
344	4d51d1dadcce224bb4f9e51b	4bf58dd8d48988d19b941735	College Academic Bu
344	4d51d1dadcce224bb4f9e51b	4bf58dd8d48988d19b941735	College Academic Bu
344	4d51d1dadcce224bb4f9e51b	4bf58dd8d48988d19b941735	College Academic Bu
344	4d51d1dadcce224bb4f9e51b	4bf58dd8d48988d19b941735	College Academic Bu

```
visit_739_exemple = db.sql("""
select *
from visit
where userId = 739
and utcTimestamp = 'Thu Apr 19 12:00:12 +0000 2012'
""")
display(visit_739_exemple)
```

userId	venueId	venueCategoryId	venueCategory
739	4eab92207beb32cb143f9d8d	4bf58dd8d48988d103941735	Home (private)
739	4ba8c654f964a520dbed39e3	4bf58dd8d48988d1fd931735	Subway
739	4f40351fa17cf53abe49a0f4	4bf58dd8d48988d103941735	Home (private)
739	4e8aaccd4fc68f2d71de9384	4bf58dd8d48988d1c7941735	Snack Place

```
visit_no_duplicate = db.sql("""
WITH T1 AS (
  SELECT *, ROW_NUMBER() OVER (PARTITION BY userId ORDER BY utcTimestamp ASC) A
  FROM visit
)
SELECT userId, utcTimestamp, venueId, venueCategoryId, venueCategory, latitude,
FROM T1
WHERE event_nb = 1
""")

display(visit_no_duplicate, 2)
```

userId	utcTimestamp	venueId	venueCategoryId
694	Fri Apr 13 10:21:00 +0000 2012	4c138eb6f1e0b7136cab34bc	4bf58dd8d48988d1
736	Fri Dec 07 19:53:17 +0000 2012	4f22ca77e4b0ed3396a83a05	4bf58dd8d48988d1

On vérifie qu'il n'y a plus de doublons

```
db.sql("""
select userId, utcTimestamp, count(*) as nb
from visit_no_duplicate
group by userId, utcTimestamp
having nb > 1
""")
```

```
┌────────┬──────────────┬───────┐
│ userId │ utcTimestamp │  nb   │
│ int64  │   varchar    │ int64 │
├────────┴──────────────┴───────┤
│            0 rows             │
└───────────────────────────────┘
```

Remplacer la table Visit par celle sans doublons

```
nb_visites_avant = db.sql("select count(*) from  visit").fetchone()[0]
print("nb de visites avant d'enlever les doublons:", nb_visites_avant)
```

```
nb de visites avant d'enlever les doublons: 227428
```

```
db.sql("""
create or replace table Visit as
select * from  visit_no_duplicate
""")
```

```
nb_visites_apres = db.sql("select count(*) from  visit").fetchone()[0]
print("nb de visites après avoir enlevé les doublons:", nb_visites_apres)
```

```
nb de visites après avoir enlevé les doublons: 1083
```

```
db.sql("""
select userId, utcTimestamp, count(*) as nb
from visit
group by userId, utcTimestamp
having nb > 1
""")
```

```
┌────────┬──────────────┬───────┐
│ userId │ utcTimestamp │  nb   │
│ int64  │   varchar    │ int64 │
├────────┴──────────────┴───────┤
│            0 rows             │
└───────────────────────────────┘
```

```
display_schema('Visit')
```

column_name	data_type
userId	BIGINT
utcTimestamp	VARCHAR
venueId	VARCHAR
venueCategoryId	VARCHAR
venueCategory	VARCHAR
latitude	DOUBLE
longitude	DOUBLE
timezoneOffset	BIGINT

Requêtes

convertir le timestamp

```
visit2 = db.sql("""
  SELECT *, strptime(utcTimestamp, '%a %b %d %X %z %Y') as datetime
  FROM visit
""")
display(visit2)
```

10 entries per page   Search:

userId	venueId	venueCategoryId	venueCategory
470	49bbd6c0f964a520f4531fe3	4bf58dd8d48988d127951735	Arts & Crafts Store
979	4a43c0aef964a520c6a61fe3	4bf58dd8d48988d1df941735	Bridge
69	4c5cc7b485a1e21e00d35711	4bf58dd8d48988d103941735	Home (private)
395	4bc7086715a7ef3bef9878da	4bf58dd8d48988d104941735	Medical Center
87	4cf2c5321d18a143951b5cec	4bf58dd8d48988d1cb941735	Food Truck
484	4b5b981bf964a520900929e3	4bf58dd8d48988d118951735	Food & Drink Shop
642	4ab966c3f964a5203c7f20e3	4bf58dd8d48988d1e0931735	Coffee Shop
292	4d0cc47f903d37041864bf55	4bf58dd8d48988d12b951735	Bus Station
428	4ce1863bc4f6a35d8bd2db6c	4bf58dd8d48988d103941735	Home (private)
877	4be319b321d5a59352311811	4bf58dd8d48988d10a951735	Bank

Showing 1 to 10 of 30 entries   « ‹ 1 2 3 › »

```
visit3 = db.sql("""
  SELECT userId, venueId, datetime
  FROM visit2
""")

display(visit3)
```

10 entries per page   Search:

userId	venueId	datetime
470	49bbd6c0f964a520f4531fe3	2012-04-03 18:00:09+00:00
979	4a43c0aef964a520c6a61fe3	2012-04-03 18:00:25+00:00
69	4c5cc7b485a1e21e00d35711	2012-04-03 18:02:24+00:00
395	4bc7086715a7ef3bef9878da	2012-04-03 18:02:41+00:00
87	4cf2c5321d18a143951b5cec	2012-04-03 18:03:00+00:00
484	4b5b981bf964a520900929e3	2012-04-03 18:04:00+00:00
642	4ab966c3f964a5203c7f20e3	2012-04-03 18:04:38+00:00
292	4d0cc47f903d37041864bf55	2012-04-03 18:04:42+00:00
428	4ce1863bc4f6a35d8bd2db6c	2012-04-03 18:06:18+00:00
877	4be319b321d5a59352311811	2012-04-03 18:06:19+00:00

Showing 1 to 10 of 30 entries   « ‹ 1 2 3 › »

Ex1 Sequences de visites

Numéro de POI

Définir les vues

- venueId_poi(venueId, poi) qui associe le numéro original de venueId avec poi.
    - Indication, utiliser la fonction row_number() over(...)
- visit_poi(userId, poi, datetime) avec des numéros de poi allant de 1 à n

```
venueIds = db.sql("""

  SELECT distinct venueId
  FROM visit2

""")

venueId_poi = db.sql("""

  SELECT venueId, row_number() over (order by venueId) as poi
  FROM venueIds

""")

display(db.sql("select * from venueId_poi order by poi"))
```

10 entries per page Search:

venueId	poi
3fd66200f964a52000e71ee3	1
3fd66200f964a52000e81ee3	2
3fd66200f964a52000f11ee3	3
3fd66200f964a52001e51ee3	4
3fd66200f964a52001e81ee3	5
3fd66200f964a52002eb1ee3	6
3fd66200f964a52003e51ee3	7
3fd66200f964a52003e71ee3	8
3fd66200f964a52003e81ee3	9
3fd66200f964a52004e41ee3	10

Showing 1 to 10 of 30 entries « ‹ 1 2 3 › »

```
db.sql("""
  select count(*)
  from venueId_poi
""")
```

```
┌──────────────┐
│ count_star() │
│    int64     │
├──────────────┤
│        38333 │
└──────────────┘
```

```
visit_poi = db.sql("""
  select v.userId, p.poi, v.datetime
  from visit2 v JOIN venueId_poi p ON
  v.venueId = p.venueId;
""")

db.sql("select * from visit_poi order by datetime")
```

```
┌────────┬───────┬──────────────────────────┐
│ userId │  poi  │         datetime         │
│ int64  │ int64 │ timestamp with time zone │
├────────┼───────┼──────────────────────────┤
│    470 │  2389 │ 2012-04-03 18:00:09+00   │
│    979 │  3922 │ 2012-04-03 18:00:25+00   │
│     69 │ 20329 │ 2012-04-03 18:02:24+00   │
│    395 │ 15115 │ 2012-04-03 18:02:41+00   │
│     87 │ 23551 │ 2012-04-03 18:03:00+00   │
│    484 │ 10590 │ 2012-04-03 18:04:00+00   │
│    642 │  6366 │ 2012-04-03 18:04:38+00   │
│    292 │ 23900 │ 2012-04-03 18:04:42+00   │
│    428 │ 23304 │ 2012-04-03 18:06:18+00   │
│    877 │ 16382 │ 2012-04-03 18:06:19+00   │
│     .  │     . │            .             │
│     .  │     . │            .             │
│     .  │     . │            .             │
│    847 │  3181 │ 2012-04-11 17:04:02+00   │
│     45 │ 11548 │ 2012-04-11 17:04:03+00   │
│    621 │ 34672 │ 2012-04-11 17:04:36+00   │
│    537 │ 34074 │ 2012-04-11 17:05:05+00   │
│    976 │ 12622 │ 2012-04-11 17:05:37+00   │
│    389 │  9326 │ 2012-04-11 17:06:11+00   │
│    663 │ 13766 │ 2012-04-11 17:08:18+00   │
│    217 │ 25584 │ 2012-04-11 17:08:38+00   │
│   1066 │  7273 │ 2012-04-11 17:09:12+00   │
│   1079 │  7862 │ 2012-04-11 17:09:13+00   │
├────────┴───────┴──────────────────────────┤
│ ? rows (>9999 rows, 20 shown)   3 columns │
└───────────────────────────────────────────┘
```

1.1) Rang

Pour chaque utilisateur, ordonner les visites par date et leur attribuer un **rang** allant de 1 (plus ancienne) à n (la plus récente)

```
visit_rank = db.sql("""
SELECT *, row_number() over (partition by userId order by datetime) as rank
FROM visit_poi
ORDER BY userId, datetime
""")

display(visit_rank)
```

10 entries per page   Search:

userId	poi	datetime	rank
1	6428	2012-04-04 23:31:31+00:00	1
1	24554	2012-04-07 17:42:24+00:00	2
1	26196	2012-04-08 18:20:29+00:00	3
1	4131	2012-04-08 20:02:10+00:00	4
1	963	2012-04-09 16:20:52+00:00	5
1	525	2012-04-10 00:24:31+00:00	6
1	1241	2012-04-10 03:36:56+00:00	7
1	33237	2012-04-10 16:21:48+00:00	8
1	2643	2012-04-12 17:19:21+00:00	9
1	33237	2012-04-13 15:41:41+00:00	10

Showing 1 to 10 of 30 entries « ‹ 1 2 3 › »

1.2) Date précédente et déplacement

On veut connaitre la durée de déplacement séparant deux visites consécutives d'un meme utilisateur. Pour chaque visite d'un utilisateur, ajouter l'attribut **prev_date** contenant la date de la visite qu'il a effectuée précédemment.

Puis ajouter l'attribut **interval_duration** calculé par différence entre la date courante et la date précédente. Indication, voir la fonction *date_diff('sec', a, b)*

```
visit_prev_date = db.sql("""
select *, COALESCE(LAG(datetime) OVER (PARTITION BY userId ORDER BY rank), date
from visit_rank
order by userId, rank

""")
display(visit_prev_date)
```

10 entries per page   Search:

userId	poi	datetime	rank	prev_date
1	6428	2012-04-04 23:31:31+00:00	1	2012-04-04 23:31:31+00:00
1	24554	2012-04-07 17:42:24+00:00	2	2012-04-04 23:31:31+00:00
1	26196	2012-04-08 18:20:29+00:00	3	2012-04-07 17:42:24+00:00
1	4131	2012-04-08 20:02:10+00:00	4	2012-04-08 18:20:29+00:00
1	963	2012-04-09 16:20:52+00:00	5	2012-04-08 20:02:10+00:00
1	525	2012-04-10 00:24:31+00:00	6	2012-04-09 16:20:52+00:00
1	1241	2012-04-10 03:36:56+00:00	7	2012-04-10 00:24:31+00:00
1	33237	2012-04-10 16:21:48+00:00	8	2012-04-10 03:36:56+00:00
1	2643	2012-04-12 17:19:21+00:00	9	2012-04-10 16:21:48+00:00
1	33237	2012-04-13 15:41:41+00:00	10	2012-04-12 17:19:21+00:00

Showing 1 to 10 of 30 entries   « ‹ 1 2 3 › »

```
visit_duration = db.sql("""
  select *, COALESCE(date_diff('second', LAG(datetime) OVER (PARTITION BY userI
  from visit_prev_date
  order by userId, rank
""")

display(visit_duration)
```

10 entries per page Search:

userId	poi	datetime	rank	prev_date	inte
1	6428	2012-04-04 23:31:31+00:00	1	2012-04-04 23:31:31+00:00	
1	24554	2012-04-07 17:42:24+00:00	2	2012-04-04 23:31:31+00:00	
1	26196	2012-04-08 18:20:29+00:00	3	2012-04-07 17:42:24+00:00	
1	4131	2012-04-08 20:02:10+00:00	4	2012-04-08 18:20:29+00:00	
1	963	2012-04-09 16:20:52+00:00	5	2012-04-08 20:02:10+00:00	
1	525	2012-04-10 00:24:31+00:00	6	2012-04-09 16:20:52+00:00	
1	1241	2012-04-10 03:36:56+00:00	7	2012-04-10 00:24:31+00:00	
1	33237	2012-04-10 16:21:48+00:00	8	2012-04-10 03:36:56+00:00	
1	2643	2012-04-12 17:19:21+00:00	9	2012-04-10 16:21:48+00:00	
1	33237	2012-04-13 15:41:41+00:00	10	2012-04-12 17:19:21+00:00	

Showing 1 to 10 of 30 entries « ‹ 1 2 3 › »

1.3) Numéro de trajectoire

Séparer les visites de chaque utilisateur en trajectoires.

- Dans une trajectoire, la durée entre deux visites consécutives ne peut pas dépasser 20 heures (soit 20 * 3600 secondes).

Ajouter l'attribut **traj** indiquant le numéro de la trajectoire pour un utilisateur donné. Indication: on peut utiliser une expression *case when then else end*

```
visit_traj1 = db.sql("""
select *, CASE
    WHEN interval_duration > 72000 OR interval_duration = 0 THEN 1
    ELSE 0
  END AS debut
from visit_duration
""")


display(visit_traj1)
```

10 entries per page   Search:

userId	poi	datetime	rank	prev_date	inte
1	6428	2012-04-04 23:31:31+00:00	1	2012-04-04 23:31:31+00:00	
1	24554	2012-04-07 17:42:24+00:00	2	2012-04-04 23:31:31+00:00	
1	26196	2012-04-08 18:20:29+00:00	3	2012-04-07 17:42:24+00:00	
1	4131	2012-04-08 20:02:10+00:00	4	2012-04-08 18:20:29+00:00	
1	963	2012-04-09 16:20:52+00:00	5	2012-04-08 20:02:10+00:00	
1	525	2012-04-10 00:24:31+00:00	6	2012-04-09 16:20:52+00:00	
1	1241	2012-04-10 03:36:56+00:00	7	2012-04-10 00:24:31+00:00	
1	33237	2012-04-10 16:21:48+00:00	8	2012-04-10 03:36:56+00:00	
1	2643	2012-04-12 17:19:21+00:00	9	2012-04-10 16:21:48+00:00	
1	33237	2012-04-13 15:41:41+00:00	10	2012-04-12 17:19:21+00:00	

Showing 1 to 10 of 30 entries   « ‹ 1 2 3 › »

```
visit_traj2 = db.sql("""
select *, SUM(debut) OVER (PARTITION BY userId ORDER BY datetime) AS traj
from visit_traj1
order by userId, rank
""")

display(visit_traj2)
```

10 entries per page   Search:

userId	poi	datetime	rank	prev_date	inte
1	6428	2012-04-04 23:31:31+00:00	1	2012-04-04 23:31:31+00:00	
1	24554	2012-04-07 17:42:24+00:00	2	2012-04-04 23:31:31+00:00	
1	26196	2012-04-08 18:20:29+00:00	3	2012-04-07 17:42:24+00:00	
1	4131	2012-04-08 20:02:10+00:00	4	2012-04-08 18:20:29+00:00	
1	963	2012-04-09 16:20:52+00:00	5	2012-04-08 20:02:10+00:00	
1	525	2012-04-10 00:24:31+00:00	6	2012-04-09 16:20:52+00:00	
1	1241	2012-04-10 03:36:56+00:00	7	2012-04-10 00:24:31+00:00	
1	33237	2012-04-10 16:21:48+00:00	8	2012-04-10 03:36:56+00:00	
1	2643	2012-04-12 17:19:21+00:00	9	2012-04-10 16:21:48+00:00	
1	33237	2012-04-13 15:41:41+00:00	10	2012-04-12 17:19:21+00:00	

Showing 1 to 10 of 30 entries   « ‹ 1 2 3 › »

Ex2: Fenètres glissantes

2.1) Nombre de POI visités en une semaine

Ajouter l'attribut **nb_POI_7jours** donnant le nombre de POI visités dans les 7 jours qui précèdent le jour de la visite courante. Ne pas

```
# pour ne pas inclure les visites du jour courant, la borne supérieure de la fe
visit_7j = db.sql("""
select * , ...... as nb_POI_7jours
from visit_poi
order by userid, datetime
""")
display(visit_7j)
```

2.2) Nombre cumulé de visites par utilisateur

Ajouter un attribut **cumul_visites** indiquant le nombre de POI qu'un utilisateur a déjà visités auparavant.

Indications: Le cumul n'est pas agrégé par utilisateur. Pour chaque visite, déterminer combien de POI ont déja été visités par cet utilisateur avant la visite courante.

```
# # Tenir compte dans le cumul des POI visités plusieurs fois : les compter qu'
```

Exercice 3

3.1) Visites dans une trajectoire

On constate qu'une trajectoire peut contenir plusieurs visites consécutives d'un même POI et on veut "fusionner" ces visites. Pour chaque trajectoire, agréger les visites consécutives d'un meme POI. Définir les attributs date_début, date_fin pour la visite d'un POI.

Rmq1: s'il n'y a pas plusieurs visites consécutives pour un POI alors ses dates de début et de fin sont identiques.

Rmq2: une trajectoire peut contenir plusieurs visites non consécutives d'un même POI.

Autre solution avec first() au lieu de lag()

```
trajectoire1 = db.sql("""
select userId, poi, datetime, rank, interval_duration, traj,
     COALESCE(LAG(poi) OVER (PARTITION BY userId, traj ORDER BY rank), poi) AS
from visit_traj2
order by userId, traj, rank

""")
display(trajectoire1)
```

10 entries per page   Search:

userId	poi	datetime	rank	interval_duration	traj
1	6428	2012-04-04 23:31:31+00:00	1	0	1
1	24554	2012-04-07 17:42:24+00:00	2	238253	2
1	26196	2012-04-08 18:20:29+00:00	3	88685	3
1	4131	2012-04-08 20:02:10+00:00	4	6101	3
1	963	2012-04-09 16:20:52+00:00	5	73122	4
1	525	2012-04-10 00:24:31+00:00	6	29019	4
1	1241	2012-04-10 03:36:56+00:00	7	11545	4
1	33237	2012-04-10 16:21:48+00:00	8	45892	4
1	2643	2012-04-12 17:19:21+00:00	9	176253	5
1	33237	2012-04-13 15:41:41+00:00	10	80540	6

Showing 1 to 10 of 30 entries   « ‹ 1 2 3 › »

```
trajectoire2 = db.sql("""
select *, ROW_NUMBER() OVER (PARTITION BY userId, traj ORDER BY rank) AS rank_i
from trajectoire1
order by userId, traj, rank_in_traj
""")

display(trajectoire2)
```

10 entries per page

Search:

userId	poi	datetime	rank	interval_duration	traj
1	6428	2012-04-04 23:31:31+00:00	1	0	1
1	24554	2012-04-07 17:42:24+00:00	2	238253	2
1	26196	2012-04-08 18:20:29+00:00	3	88685	3
1	4131	2012-04-08 20:02:10+00:00	4	6101	3
1	963	2012-04-09 16:20:52+00:00	5	73122	4
1	525	2012-04-10 00:24:31+00:00	6	29019	4
1	1241	2012-04-10 03:36:56+00:00	7	11545	4
1	33237	2012-04-10 16:21:48+00:00	8	45892	4
1	2643	2012-04-12 17:19:21+00:00	9	176253	5
1	33237	2012-04-13 15:41:41+00:00	10	80540	6

Showing 1 to 10 of 30 entries

« ‹ 1 2 3 › »

afficher les visites de l'utilisateur 984

```
trajectoire3 = db.sql("""
select *, CASE
    -- If it's the first row in the trajectory, prev_poi is NULL, in this case
    WHEN rank_in_traj = 1 THEN 1
    -- If the previous POI is different from the current POI, this is a new POI
    WHEN prev_poi IS NULL OR prev_poi != poi THEN 1
    -- Otherwise, it indicates a continuous visit to the same POI, debut_poi is
    ELSE 0
  END AS debut_poi
from trajectoire2
order by userId, traj, rank

""")
```

```
trajectoire3
```

```
┌────────┬───────┬──────────────────────────┬───────┬──────────────────┬─
│ userId │  poi  │         datetime         │ rank  │ interval_duration │
traj   │ prev_poi  │ rank_in_traj │ debut_poi
│ int64  │ int64 │ timestamp with time zone │ int64 │       int64       │
int128 │  int64   │    int64     │   int32   │
├────────┼───────┼──────────────────────────┼───────┼──────────────────┼─
│      1 │  6428 │ 2012-04-04 23:31:31+00   │     1 │                 0 │
1 │     6428 │            1 │           1 │
│      1 │ 24554 │ 2012-04-07 17:42:24+00   │     2 │            238253 │
2 │    24554 │            1 │           1 │
│      1 │ 26196 │ 2012-04-08 18:20:29+00   │     3 │             88685 │
3 │    26196 │            1 │           1 │
│      1 │  4131 │ 2012-04-08 20:02:10+00   │     4 │              6101 │
3 │    26196 │            2 │           1 │
│      1 │   963 │ 2012-04-09 16:20:52+00   │     5 │             73122 │
4 │      963 │            1 │           1 │
│      1 │   525 │ 2012-04-10 00:24:31+00   │     6 │             29019 │
4 │      963 │            2 │           1 │
│      1 │  1241 │ 2012-04-10 03:36:56+00   │     7 │             11545 │
4 │      525 │            3 │           1 │
│      1 │ 33237 │ 2012-04-10 16:21:48+00   │     8 │             45892 │
4 │     1241 │            4 │           1 │
│      1 │  2643 │ 2012-04-12 17:19:21+00   │     9 │            176253 │
5 │     2643 │            1 │           1 │
│      1 │ 33237 │ 2012-04-13 15:41:41+00   │    10 │             80540 │
6 │    33237 │            1 │           1 │
│      · │   ·   │            ·             │     · │                 ·  │
· │        ·  │            · │           · │
│      · │   ·   │            ·             │     · │                 ·  │
· │        ·  │            · │           · │
│      · │   ·   │            ·             │     · │                 ·  │
· │        ·  │            · │           · │
│     56 │  5511 │ 2012-12-22 01:22:30+00   │   308 │              5186 │
73 │     3709 │            3 │           1 │
│     56 │  3681 │ 2012-12-22 19:30:05+00   │   309 │             65255 │
73 │     5511 │            4 │           1 │
│     56 │ 31764 │ 2012-12-28 13:01:17+00   │   310 │            495072 │
74 │    31764 │            1 │           1 │
│     57 │ 15816 │ 2012-04-03 22:26:51+00   │     1 │                 0 │
1 │    15816 │            1 │           1 │
│     57 │  5651 │ 2012-04-04 13:07:31+00   │     2 │             52840 │
1 │    15816 │            2 │           1 │
│     57 │ 15816 │ 2012-04-07 13:54:16+00   │     3 │            262005 │
2 │    15816 │            1 │           1 │
│     57 │ 21139 │ 2012-04-07 16:22:45+00   │     4 │              8909 │
2 │    15816 │            2 │           1 │
│     57 │ 25176 │ 2012-04-07 20:29:23+00   │     5 │             14798 │
2 │    21139 │            3 │           1 │
│     57 │  5651 │ 2012-04-09 12:58:54+00   │     6 │            145771 │
3 │     5651 │            1 │           1 │
│     57 │ 15816 │ 2012-04-09 22:18:52+00   │     7 │             33598 │
3 │     5651 │            2 │           1 │
├────────┴───────┴──────────────────────────┴───────┴──────────────────┴─
│ ? rows (>9999 rows, 20 shown)
```

```
9 columns │
└───────────────────────────────────────────────────────────────────────
```

```
db.sql("""
select *
from trajectoire3
where userId=984
order by traj
""")
```

```
┌────────┬───────┬──────────────────────────┬───────┬───────────────────┬─
│ userId │  poi  │         datetime         │ rank  │ interval_duration │
traj   │ prev_poi  rank_in_traj │ debut_poi
│ int64  │ int64 │ timestamp with time zone │ int64 │       int64       │
int128 │  int64   │     int64     │    int32    │
├────────┼───────┼──────────────────────────┼───────┼───────────────────┼─
│    984 │  2532 │ 2012-04-03 21:31:50+00   │     1 │                 0 │
1 │    2532 │            1 │          1 │
│    984 │  2532 │ 2012-04-04 09:30:00+00   │     2 │             43090 │
1 │    2532 │            2 │          0 │
│    984 │  2532 │ 2012-04-05 10:06:32+00   │     3 │             88592 │
2 │    2532 │            1 │          1 │
│    984 │  2532 │ 2012-04-10 22:40:41+00   │     4 │            477249 │
3 │    2532 │            1 │          1 │
│    984 │  2532 │ 2012-04-11 09:30:15+00   │     5 │             38974 │
3 │    2532 │            2 │          0 │
│    984 │ 20194 │ 2012-04-11 16:03:05+00   │     6 │             23570 │
3 │    2532 │            3 │          1 │
│    984 │  2532 │ 2012-04-12 13:54:55+00   │     7 │             78710 │
4 │    2532 │            1 │          1 │
│    984 │  2532 │ 2012-04-13 10:33:28+00   │     8 │             74313 │
5 │    2532 │            1 │          1 │
│    984 │  2532 │ 2012-04-16 10:36:29+00   │     9 │            259381 │
6 │    2532 │            1 │          1 │
│    984 │ 20194 │ 2012-04-16 15:02:14+00   │    10 │             15945 │
6 │    2532 │            2 │          1 │
│     .  │    .  │            .             │     . │                 . │
. │       .  │            . │          . │
│     .  │    .  │            .             │     . │                 . │
. │       .  │            . │          . │
│     .  │    .  │            .             │     . │                 . │
. │       .  │            . │          . │
│    984 │  2532 │ 2013-01-25 11:24:30+00   │    96 │             89587 │
82 │    2532 │            1 │          1 │
│    984 │  2532 │ 2013-01-28 10:26:33+00   │    97 │            255723 │
83 │    2532 │            1 │          1 │
│    984 │  2532 │ 2013-01-29 10:35:14+00   │    98 │             86921 │
84 │    2532 │            1 │          1 │
│    984 │  2532 │ 2013-01-30 10:38:46+00   │    99 │             86612 │
85 │    2532 │            1 │          1 │
│    984 │  2532 │ 2013-02-01 10:34:44+00   │   100 │            172558 │
86 │    2532 │            1 │          1 │
│    984 │  2532 │ 2013-02-06 20:55:05+00   │   101 │            469221 │
87 │    2532 │            1 │          1 │
│    984 │  2532 │ 2013-02-07 11:00:42+00   │   102 │             50737 │
```

```
87 │      2532 │             2 │          0 │
│    984 │  2532 │ 2013-02-08 10:34:26+00   │   103 │              84824 │
88 │      2532 │             1 │          1 │
│    984 │  2532 │ 2013-02-11 10:35:41+00   │   104 │             259275 │
89 │      2532 │             1 │          1 │
│    984 │  2532 │ 2013-02-13 11:29:11+00   │   105 │             176010 │
90 │      2532 │             1 │          1 │
├────────┴───────┴──────────────────────────┴───────┴────────────────────┴──
│ 105 rows (20 shown)
9 columns │
└─────────────────────────────────────────────────────────────────────────────
```

```
trajectoire4 = db.sql("""
select *,
    sum(debut_poi) over (partition by userid, traj order by rank rows between u
from trajectoire3
order by userId, traj, rank
""")

trajectoire4
```

[redacted]

```
┌────────┬───────┬──────────────────────────┬───────┬────────────────────┬──
│ userId │  poi  │         datetime         │ rank  │ interval_duration │
traj    │ prev_poi  │ rank_in_traj │ debut_poi │ visitid │
│ int64  │ int64 │ timestamp with time zone │ int64 │       int64       │
int128 │  int64     │    int64     │   int32    │ int128  │
├────────┼───────┼──────────────────────────┼───────┼────────────────────┼──
│      1 │  6428 │ 2012-04-04 23:31:31+00   │     1 │                  0 │
1 │     6428 │             1 │          1 │     1 │
│      1 │ 24554 │ 2012-04-07 17:42:24+00   │     2 │             238253 │
2 │    24554 │             1 │          1 │     1 │
│      1 │ 26196 │ 2012-04-08 18:20:29+00   │     3 │              88685 │
3 │    26196 │             1 │          1 │     1 │
│      1 │  4131 │ 2012-04-08 20:02:10+00   │     4 │               6101 │
3 │    26196 │             2 │          1 │     2 │
│      1 │   963 │ 2012-04-09 16:20:52+00   │     5 │              73122 │
4 │      963 │             1 │          1 │     1 │
│      1 │   525 │ 2012-04-10 00:24:31+00   │     6 │              29019 │
4 │      963 │             2 │          1 │     2 │
│      1 │  1241 │ 2012-04-10 03:36:56+00   │     7 │              11545 │
4 │      525 │             3 │          1 │     3 │
│      1 │ 33237 │ 2012-04-10 16:21:48+00   │     8 │              45892 │
4 │     1241 │             4 │          1 │     4 │
│      1 │  2643 │ 2012-04-12 17:19:21+00   │     9 │             176253 │
5 │     2643 │             1 │          1 │     1 │
│      1 │ 33237 │ 2012-04-13 15:41:41+00   │    10 │              80540 │
6 │    33237 │             1 │          1 │     1 │
│      · │     · │            ·             │     · │                  · │
· │       ·  │             · │          · │     · │
│      · │     · │            ·             │     · │                  · │
· │       ·  │             · │          · │     · │
```

```
trajectoire5 = db.sql("""
select userid, traj, visitid, poi,
    min(datetime) as date_in,
    max(datetime) as date_out,
    count(*) as nb_checkin
from trajectoire4
group by userid, traj, visitid, poi
order by userid, traj, visitid
""")
trajectoire5
```

```
┌────────┬────────┬─────────┬───────┬──────────────────────────┬──────────
│ userId │  traj  │ visitid │  poi  │         date_in          │
date_out         │ nb_checkin │
│ int64  │ int128 │ int128  │ int64 │ timestamp with time zone │ timestamp
with time zone │   int64
├────────┼────────┼─────────┼───────┼──────────────────────────┼──────────
│      1 │      1 │       1 │  6428 │ 2012-04-04 23:31:31+00   │ 2012-04-
04 23:31:31+00   │          1 │
│      1 │      2 │       1 │ 24554 │ 2012-04-07 17:42:24+00   │ 2012-04-
07 17:42:24+00   │          1 │
│      1 │      3 │       1 │ 26196 │ 2012-04-08 18:20:29+00   │ 2012-04-
08 18:20:29+00   │          1 │
│      1 │      3 │       2 │  4131 │ 2012-04-08 20:02:10+00   │ 2012-04-
08 20:02:10+00   │          1 │
│      1 │      4 │       1 │   963 │ 2012-04-09 16:20:52+00   │ 2012-04-
09 16:20:52+00   │          1 │
│      1 │      4 │       2 │   525 │ 2012-04-10 00:24:31+00   │ 2012-04-
10 00:24:31+00   │          1 │
│      1 │      4 │       3 │  1241 │ 2012-04-10 03:36:56+00   │ 2012-04-
10 03:36:56+00   │          1 │
│      1 │      4 │       4 │ 33237 │ 2012-04-10 16:21:48+00   │ 2012-04-
10 16:21:48+00   │          1 │
│      1 │      5 │       1 │  2643 │ 2012-04-12 17:19:21+00   │ 2012-04-
12 17:19:21+00   │          1 │
│      1 │      6 │       1 │ 33237 │ 2012-04-13 15:41:41+00   │ 2012-04-
13 15:41:41+00   │          1 │
│      · │      · │       · │   ·   │            ·             │
·               │          ·
│      · │      · │       · │   ·   │            ·             │
·               │          ·
│      · │      · │       · │   ·   │            ·             │
·               │          ·
│     59 │      2 │       2 │ 16256 │ 2012-04-08 21:49:13+00   │ 2012-04-
08 21:49:13+00   │          1 │
│     59 │      2 │       3 │ 21679 │ 2012-04-09 13:21:21+00   │ 2012-04-
09 13:21:21+00   │          1 │
│     59 │      2 │       4 │ 28871 │ 2012-04-09 22:08:36+00   │ 2012-04-
09 22:08:36+00   │          1 │
│     59 │      3 │       1 │ 26959 │ 2012-04-10 22:01:49+00   │ 2012-04-
10 22:01:49+00   │          1 │
│     59 │      3 │       2 │  7780 │ 2012-04-10 22:02:08+00   │ 2012-04-
10 22:02:08+00   │          1 │
│     59 │      3 │       3 │ 30998 │ 2012-04-10 22:02:27+00   │ 2012-04-
10 22:02:27+00   │          1 │
│     59 │      4 │       1 │ 21027 │ 2012-04-13 19:13:00+00   │ 2012-04-
```

```
13 19:13:00+00   │          1 │
│     59 │      4 │       2 │ 28599 │ 2012-04-14 01:22:46+00   │ 2012-04-
14 01:22:46+00   │          1 │
│     59 │      5 │       1 │ 29513 │ 2012-04-15 19:06:38+00   │ 2012-04-
15 19:06:38+00   │          1 │
│     59 │      5 │       2 │ 14923 │ 2012-04-16 14:06:45+00   │ 2012-04-
16 14:06:45+00   │          1 │
├────────┴────────┴─────────┴───────┴──────────────────────────┴─────────
│ ? rows (>9999 rows, 20 shown)
7 columns │
└────────────────────────────────────────────────────────────────────────
```

```
trajectoire6 = db.sql("""
SELECT T1.userId, T1.traj, T1.visitId, T2.poi, T1.nb_checkin, T1.date_in, T1.da
FROM trajectoire5 T1 JOIN trajectoire4 T2
ON T1.userId = T2.userId AND T1.traj = T2.traj AND T1.visitId = T2.visitId
GROUP BY T1.userId, T1.traj, T1.visitId, T2.poi, T1.nb_checkin, T1.date_in, T1.
ORDER BY T1.userId, T1.traj, T1.visitId
""")
trajectoire6
```

[redacted]

```
┌────────┬────────┬─────────┬───────┬────────────┬──────────────────────────
│ userId │  traj  │ visitid │  poi  │ nb_checkin │          date_in
      date_out          │ duration │
│ int64  │ int128 │ int128  │ int64 │   int64    │ timestamp with time zone
  timestamp with time zone │  int32   │
├────────┼────────┼─────────┼───────┼────────────┼──────────────────────────
│      1 │      1 │       1 │  6428 │          1 │ 2012-04-04 23:31:31+00
  2012-04-04 23:31:31+00    │        0 │
│      1 │      2 │       1 │ 24554 │          1 │ 2012-04-07 17:42:24+00
  2012-04-07 17:42:24+00    │        0 │
│      1 │      3 │       1 │ 26196 │          1 │ 2012-04-08 18:20:29+00
  2012-04-08 18:20:29+00    │        0 │
│      1 │      3 │       2 │  4131 │          1 │ 2012-04-08 20:02:10+00
  2012-04-08 20:02:10+00    │        0 │
│      1 │      4 │       1 │   963 │          1 │ 2012-04-09 16:20:52+00
  2012-04-09 16:20:52+00    │        0 │
│      1 │      4 │       2 │   525 │          1 │ 2012-04-10 00:24:31+00
  2012-04-10 00:24:31+00    │        0 │
│      1 │      4 │       3 │  1241 │          1 │ 2012-04-10 03:36:56+00
  2012-04-10 03:36:56+00    │        0 │
│      1 │      4 │       4 │ 33237 │          1 │ 2012-04-10 16:21:48+00
  2012-04-10 16:21:48+00    │        0 │
│      1 │      5 │       1 │  2643 │          1 │ 2012-04-12 17:19:21+00
  2012-04-12 17:19:21+00    │        0 │
│      1 │      6 │       1 │ 33237 │          1 │ 2012-04-13 15:41:41+00
  2012-04-13 15:41:41+00    │        0 │
│      · │      · │       · │   ·   │          · │            ·
            ·               │        · │
│      · │      · │       · │   ·   │          · │            ·
            ·               │        · │
```

```
db.sql("""
select *
from trajectoire6
where userId=984
order by traj, visitId

""")
```

```
┌────────┬────────┬─────────┬───────┬────────────┬──────────────────────────
│ userId │  traj  │ visitid │  poi  │ nb_checkin │          date_in
│          date_out         │ duration │
│ int64  │ int128 │ int128  │ int64 │    int64     │ timestamp with time
zone │ timestamp with time zone │  int32   │
├────────┼────────┼─────────┼───────┼─────────────┼──────────────────────
│    984 │      1 │       1 │  2532 │           2 │ 2012-04-03 21:31:50+00
│ 2012-04-04 09:30:00+00    │    43090 │
│    984 │      2 │       1 │  2532 │           1 │ 2012-04-05 10:06:32+00
│ 2012-04-05 10:06:32+00    │        0 │
│    984 │      3 │       1 │  2532 │           2 │ 2012-04-10 22:40:41+00
│ 2012-04-11 09:30:15+00    │    38974 │
│    984 │      3 │       2 │ 20194 │           1 │ 2012-04-11 16:03:05+00
│ 2012-04-11 16:03:05+00    │        0 │
│    984 │      4 │       1 │  2532 │           1 │ 2012-04-12 13:54:55+00
│ 2012-04-12 13:54:55+00    │        0 │
│    984 │      5 │       1 │  2532 │           1 │ 2012-04-13 10:33:28+00
│ 2012-04-13 10:33:28+00    │        0 │
│    984 │      6 │       1 │  2532 │           1 │ 2012-04-16 10:36:29+00
│ 2012-04-16 10:36:29+00    │        0 │
│    984 │      6 │       2 │ 20194 │           1 │ 2012-04-16 15:02:14+00
│ 2012-04-16 15:02:14+00    │        0 │
│    984 │      7 │       1 │  2532 │           1 │ 2012-04-17 12:59:26+00
│ 2012-04-17 12:59:26+00    │        0 │
│    984 │      8 │       1 │ 13334 │           1 │ 2012-04-18 14:16:55+00
│ 2012-04-18 14:16:55+00    │        0 │
│     ·  │      · │       · │    ·  │           · │              ·
│           ·               │        · │
│     ·  │      · │       · │    ·  │           · │              ·
│           ·               │        · │
│     ·  │      · │       · │    ·  │           · │              ·
│           ·               │        · │
│    984 │     81 │       1 │  2532 │           1 │ 2013-01-24 10:31:23+00
│ 2013-01-24 10:31:23+00    │        0 │
│    984 │     82 │       1 │  2532 │           1 │ 2013-01-25 11:24:30+00
│ 2013-01-25 11:24:30+00    │        0 │
│    984 │     83 │       1 │  2532 │           1 │ 2013-01-28 10:26:33+00
│ 2013-01-28 10:26:33+00    │        0 │
│    984 │     84 │       1 │  2532 │           1 │ 2013-01-29 10:35:14+00
│ 2013-01-29 10:35:14+00    │        0 │
│    984 │     85 │       1 │  2532 │           1 │ 2013-01-30 10:38:46+00
│ 2013-01-30 10:38:46+00    │        0 │
│    984 │     86 │       1 │  2532 │           1 │ 2013-02-01 10:34:44+00
│ 2013-02-01 10:34:44+00    │        0 │
│    984 │     87 │       1 │  2532 │           2 │ 2013-02-06 20:55:05+00
│ 2013-02-07 11:00:42+00    │    50737 │
│    984 │     88 │       1 │  2532 │           1 │ 2013-02-08 10:34:26+00
│ 2013-02-08 10:34:26+00    │        0 │
│    984 │     89 │       1 │  2532 │           1 │ 2013-02-11 10:35:41+00
│ 2013-02-11 10:35:41+00    │        0 │
│    984 │     90 │       1 │  2532 │           1 │ 2013-02-13 11:29:11+00
│ 2013-02-13 11:29:11+00    │        0 │
├────────┴────────┴─────────┴───────┴─────────────┴──────────────────────
│ 99 rows (20 shown)
8 columns │
└────────────────────────────────────────────────────────────────────────
```