In [ ]:

Projet DBScan & K-Means

ESP-SID

Mohamed Habib _ 22106

Objectif du Projet

Le projet vise à optimiser les points stratégiques des chauffeurs Uber dans la ville de New York en utilisant des techniques de clustering de machine learning. Les principales questions auxquelles nous cherchons à répondre sont les suivantes :

1. **Détermination du Nombre Optimal de Clusters :** Identifier le nombre optimal de clusters en utilisant la méthode du coude pour regrouper les emplacements de ramassage d'Uber.
2. **Visualisation sur une Carte :** Représenter graphiquement les clusters sur une carte interactive en utilisant la bibliothèque Bokeh, facilitant ainsi la compréhension spatiale des points stratégiques.
3. **Segmentation Temporelle des Clusters :** Séparer les clusters en fonction des périodes de la journée pour comprendre les variations de la demande ou des schémas de déplacement des utilisateurs au fil du temps.
4. **Comparaison entre KMeans et DBSCAN :** Examiner les différences entre les méthodes de clustering KMeans et DBSCAN pour évaluer leur efficacité dans la définition des clusters.

Les données Uber du mois de septembre 2014 sont utilisées, comprenant des observations sur la date/heure, la latitude, la longitude et le code de la société de base pour chaque ramassage Uber. L'objectif ultime est d'optimiser la répartition des chauffeurs Uber dans la ville, améliorant ainsi leur efficacité et la satisfaction des utilisateurs.

Chargement du dataset

In [3]:
```
import pandas as pd

url = "https://raw.githubusercontent.com/fivethirtyeight/uber-tlc-foil-response/master/uber-trip-data/uber-raw-data-apr14.csv"
data = pd.read_csv(url)
print(data.head())
```

```
          Date/Time      Lat      Lon    Base
0  4/1/2014 0:11:00  40.7690 -73.9549  B02512
1  4/1/2014 0:17:00  40.7267 -74.0345  B02512
2  4/1/2014 0:21:00  40.7316 -73.9873  B02512
3  4/1/2014 0:28:00  40.7588 -73.9776  B02512
4  4/1/2014 0:33:00  40.7594 -73.9722  B02512
```

1 .1 _ Exploration des Données

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 564516 entries, 0 to 564515
Data columns (total 6 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   Date/Time  564516 non-null  datetime64[ns]
 1   Lat        564516 non-null  float64
 2   Lon        564516 non-null  float64
 3   Base       564516 non-null  object
 4   Date       564516 non-null  object
 5   Time       564516 non-null  object
dtypes: datetime64[ns](1), float64(2), object(3)
memory usage: 25.8+ MB
```

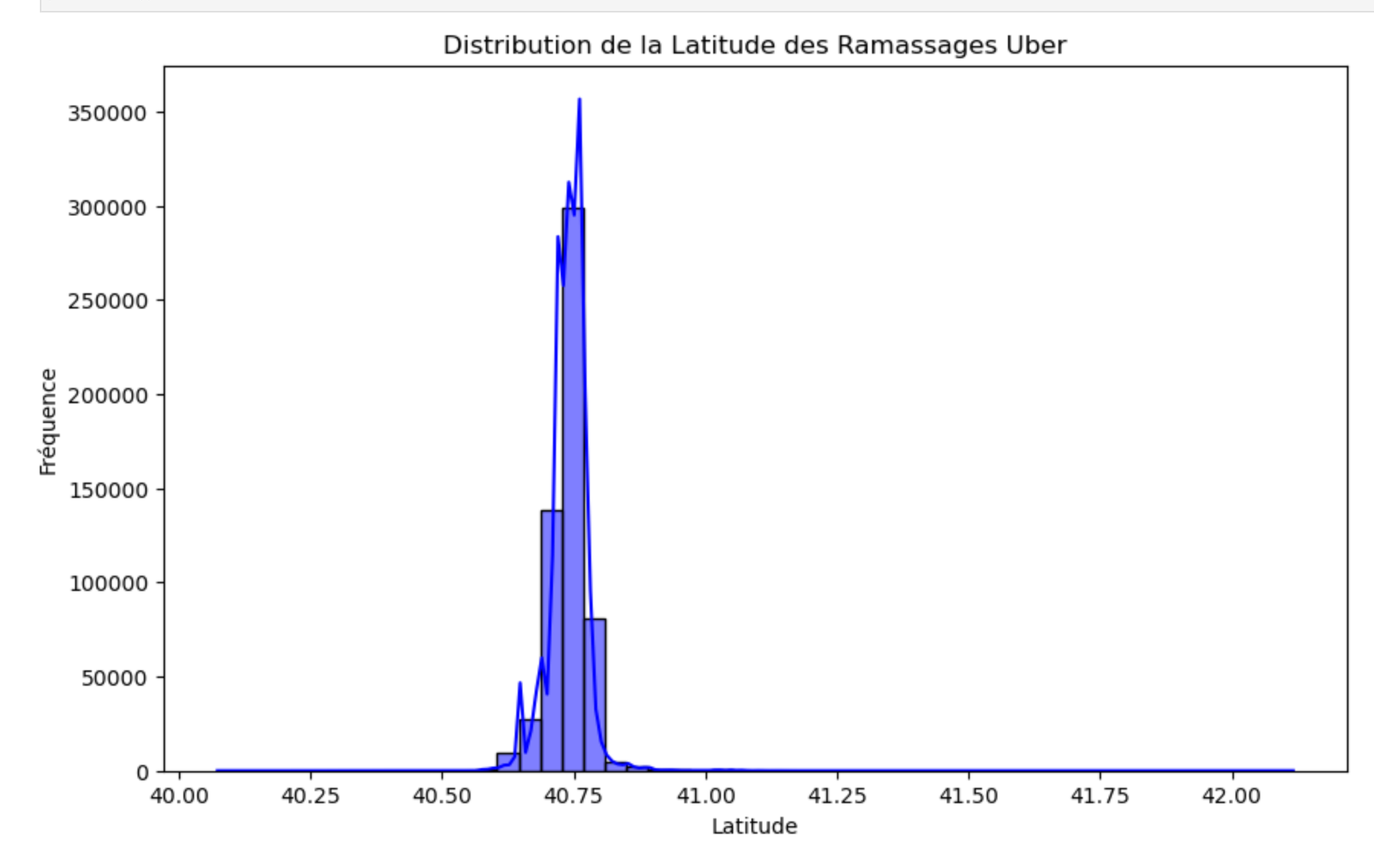
In [4]:
```
data.describe()
```

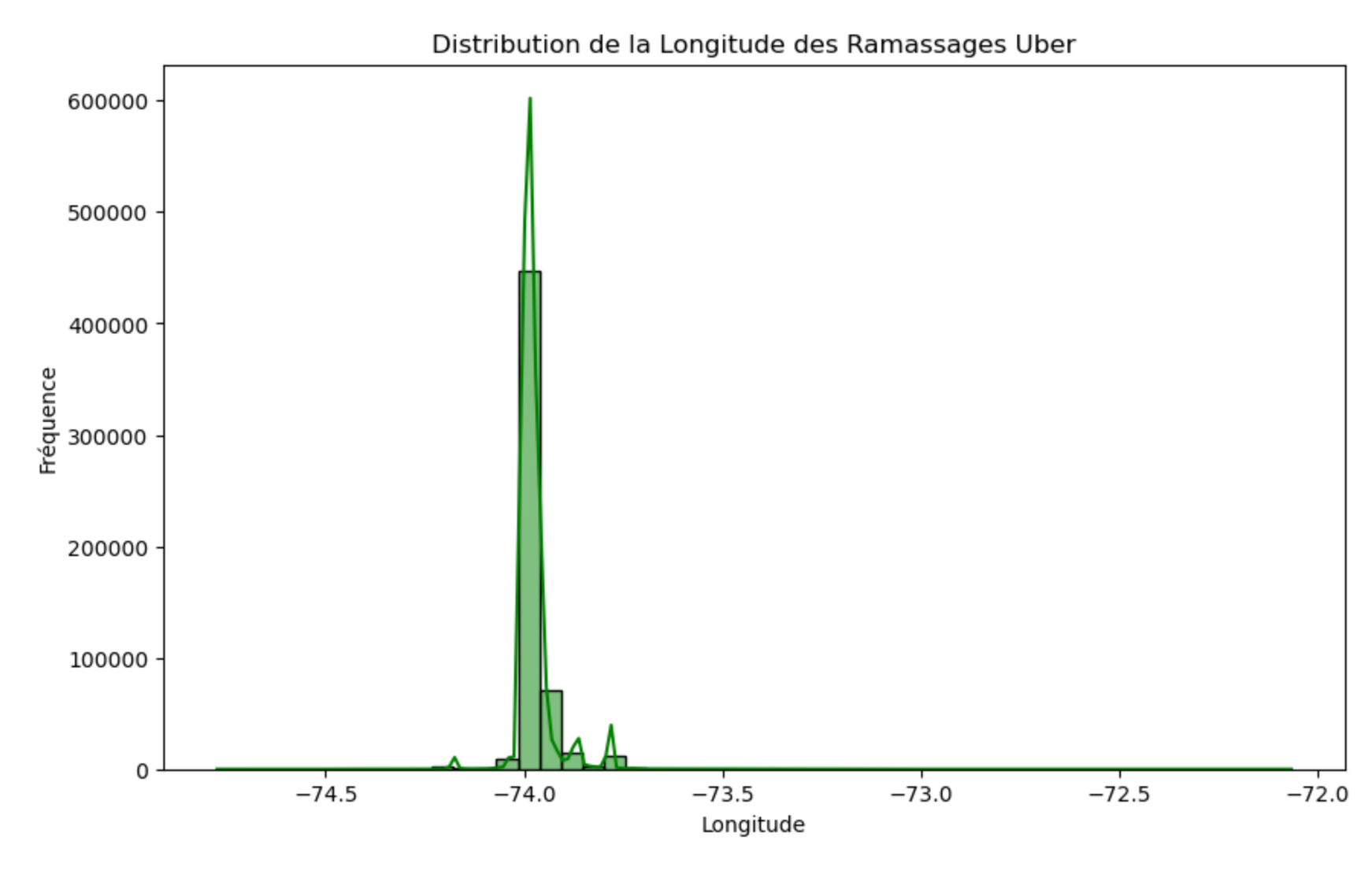
Out[4]:

	Date/Time	Lat	Lon
count	564516	564516.000000	564516.000000
mean	2014-04-16 17:46:01.296261120	40.740005	-73.976817
min	2014-04-01 00:00:00	40.072900	-74.773300
25%	2014-04-08 16:27:00	40.722500	-73.997700
50%	2014-04-16 19:19:00	40.742500	-73.984800
75%	2014-04-24 22:14:00	40.760700	-73.970000
max	2014-04-30 23:59:00	42.116600	-72.066600
std	NaN	0.036083	0.050426

1.2_Visualisation des Distributions

In [5]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.histplot(data['Lat'], bins=50, kde=True, color='blue')
plt.title('Distribution de la Latitude des Ramassages Uber')
plt.xlabel('Latitude')
plt.ylabel('Fréquence')
plt.show()

plt.figure(figsize=(10, 6))
sns.histplot(data['Lon'], bins=50, kde=True, color='green')
plt.title('Distribution de la Longitude des Ramassages Uber')
plt.xlabel('Longitude')
plt.ylabel('Fréquence')
plt.show()
```





Interprétation des Distributions de Latitude et Longitude des Ramassages Uber

Les histogrammes ci-dessus offrent un aperçu des distributions de la latitude et de la longitude des points de ramassage Uber à New York.

Distribution de la Latitude des Ramassages Uber

La première visualisation révèle que la densité des ramassages est plus élevée autour de valeurs de latitude spécifiques. La courbe KDE indique une concentration significative des ramassages autour de la latitude 40.7, suggérant des zones où l'activité d'Uber est plus fréquente. La dispersion des points est relativement plus faible, indiquant des zones spécifiques de forte demande.

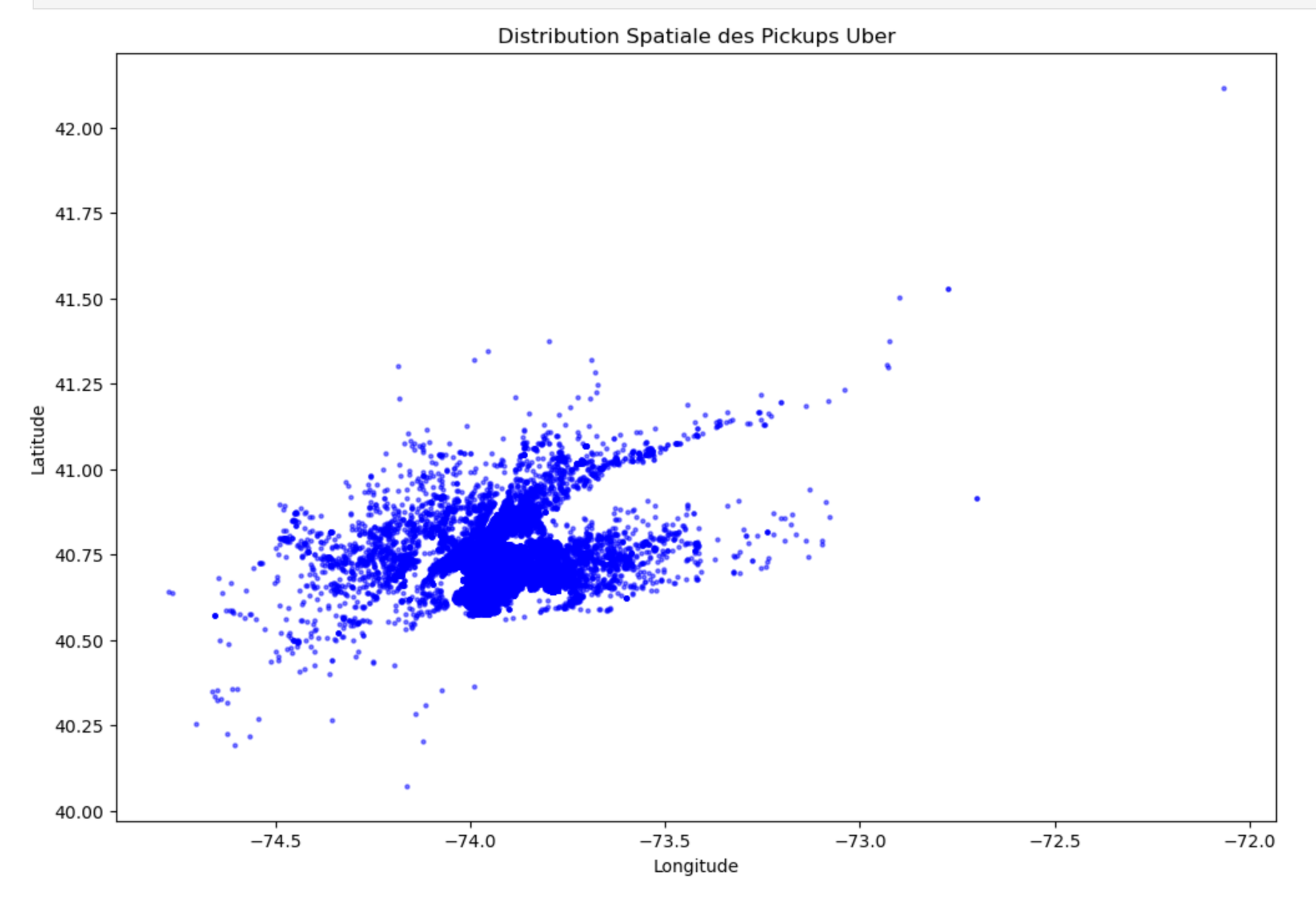
Distribution de la Longitude des Ramassages Uber

Le deuxième graphique montre que la densité des ramassages varie également autour de valeurs de longitude spécifiques. La courbe KDE suggère des zones de concentration plus étalées en longitude par rapport à la latitude. Cela peut indiquer une plus grande diversité dans les zones couvertes par les ramassages, avec des clusters de ramassages dans différentes parties de la ville.

Visualisation de la Distribution Spatiale des Pickups Uber

Le code ci-dessous utilise un scatter plot pour représenter la répartition des pickups Uber en fonction de la latitude et de la longitude dans votre ensemble de données

In [6]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 8))
plt.scatter(data['Lon'], data['Lat'], s=5, alpha=0.5, color='blue')
plt.title('Distribution Spatiale des Pickups Uber')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```



Interprétation de la Distribution Spatiale des Pickups Uber

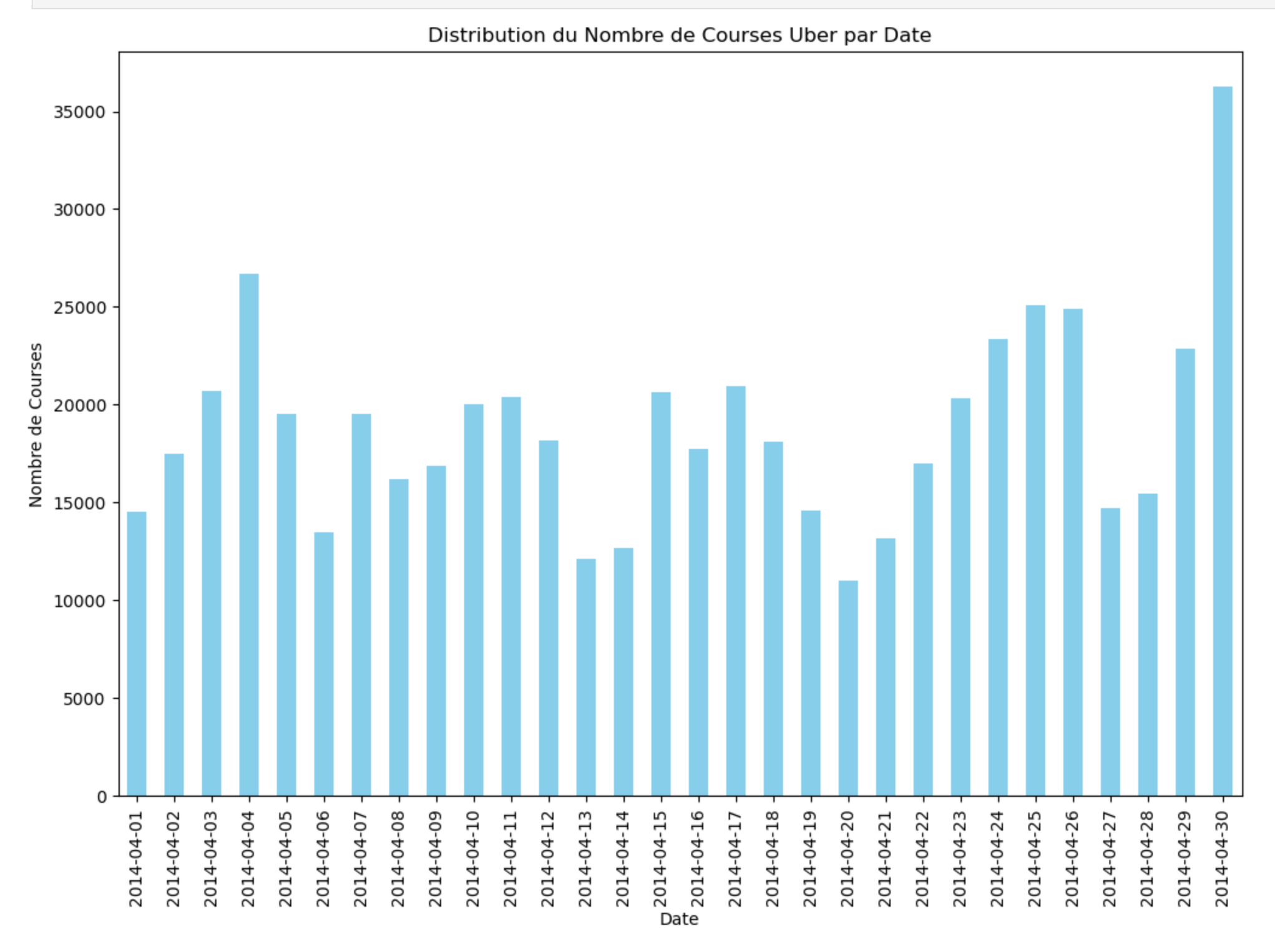
La visualisation du scatter plot suggère plusieurs observations concernant la distribution spatiale des pickups Uber :

- Les clusters de points denses indiquent des zones de forte concentration de pickups Uber. Ces zones pourraient correspondre à des quartiers populaires, des centres-villes ou des lieux de forte demande.
- Les zones où les points sont plus dispersés suggèrent une répartition moins dense de pickups. Ces régions pourraient correspondre à des zones résidentielles moins fréquentées ou des zones avec une demande moins constante.

Distribution du Nombre de Courses Uber par Date

Le code ci-dessous génère un histogramme illustrant la distribution du nombre de courses Uber pour chaque date dans le jeu de données.

In [7]:
```
rides_per_date = data.groupby('Date').size()
plt.figure(figsize=(12, 8))
rides_per_date.plot(kind='bar', color='skyblue')
plt.title('Distribution du Nombre de Courses Uber par Date')
plt.xlabel('Date')
plt.ylabel('Nombre de Courses')
plt.show()
```



Interprétation de la Distribution du Nombre de Courses Uber par Date

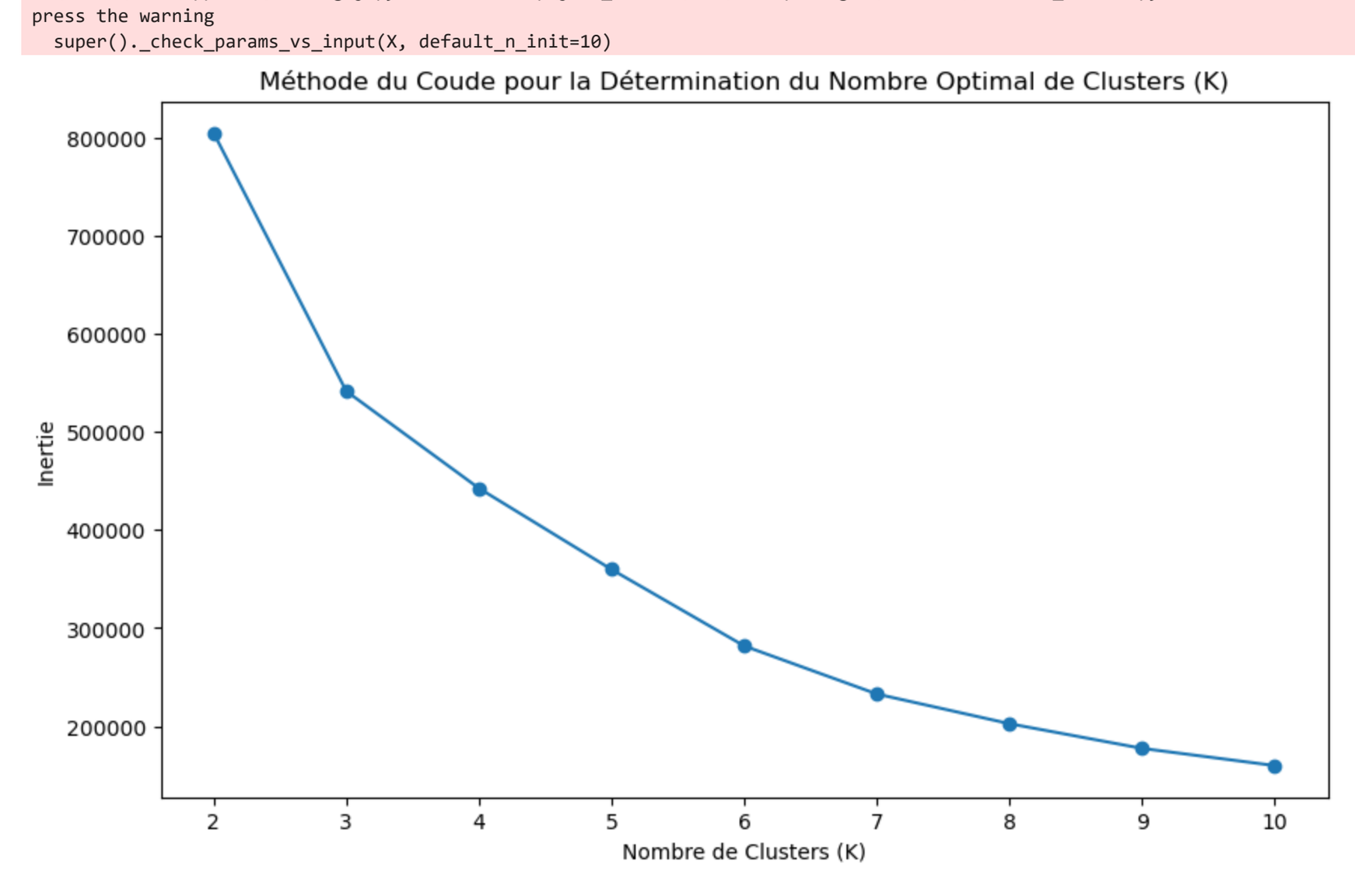
L'histogramme "Distribution du Nombre de Courses Uber par Date" offre un aperçu des tendances de la demande au cours du mois de septembre 2014. Voici quelques observations :

- **Pics de Demande Élevée :** Les pics dans l'histogramme correspondent à des journées où la demande de courses Uber était exceptionnellement élevée. Ces jours pourraient être associés à des événements spéciaux, des conditions météorologiques particulières ou d'autres facteurs qui ont stimulé la demande.
- **Périodes de Demande Plus Faible :** Les jours où les barres de l'histogramme sont plus basses suggèrent des périodes de demande plus faible. Cela peut être lié à des jours de la semaine où la demande est généralement moins élevée ou à des jours sans événements particuliers.

2_Détermination du Nombre Optimal de Clusters

In [8]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

data_for_clustering = data[['Lat', 'Lon']]

scaler = StandardScaler()
scaled_data = scaler.fit_transform(data_for_clustering)
inertia_values = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_data)
    inertia_values.append(kmeans.inertia_)

plt.figure(figsize=(10, 6))
plt.plot(range(2, 11), inertia_values, marker='o')
plt.title("Méthode du Coude pour la Détermination du Nombre Optimal de Clusters (K)")
plt.xlabel("Nombre de Clusters (K)")
plt.ylabel("Inertie")
plt.show()
```

```
C:\Users\Dell\AppData\Roaming\jupyterlab-desktop\jlab_server\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to sup
press the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Dell\AppData\Roaming\jupyterlab-desktop\jlab_server\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to sup
press the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Dell\AppData\Roaming\jupyterlab-desktop\jlab_server\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to sup
press the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Dell\AppData\Roaming\jupyterlab-desktop\jlab_server\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to sup
press the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Dell\AppData\Roaming\jupyterlab-desktop\jlab_server\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to sup
press the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Dell\AppData\Roaming\jupyterlab-desktop\jlab_server\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to sup
press the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Dell\AppData\Roaming\jupyterlab-desktop\jlab_server\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to sup
press the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Dell\AppData\Roaming\jupyterlab-desktop\jlab_server\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to sup
press the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Dell\AppData\Roaming\jupyterlab-desktop\jlab_server\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to sup
press the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



Interprétation de la Méthode du Coude pour le Clustering avec KMeans

Le graphique du coude ci-dessus a été généré en utilisant la méthode du coude pour déterminer le nombre optimal de clusters (K) dans l'algorithme KMeans. Voici quelques observations :

- **Pente de l'Inertie :** L'inertie, représentée sur l'axe des ordonnées, mesure la somme des carrés des distances des points de données à leur centre de cluster. Une pente plus raide de la courbe d'inertie indique une meilleure séparation des clusters.
- **Nombre Optimal de Clusters (K) :** La méthode du coude vise à identifier le point où l'ajout de clusters supplémentaires n'apporte pas une réduction significative de l'inertie. Dans le graphique, le "coude" est le point où l'inertie cesse de diminuer rapidement. Dans ce cas egal a 3, le nombre optimal de clusters (K) semble être là où le coude est situé.

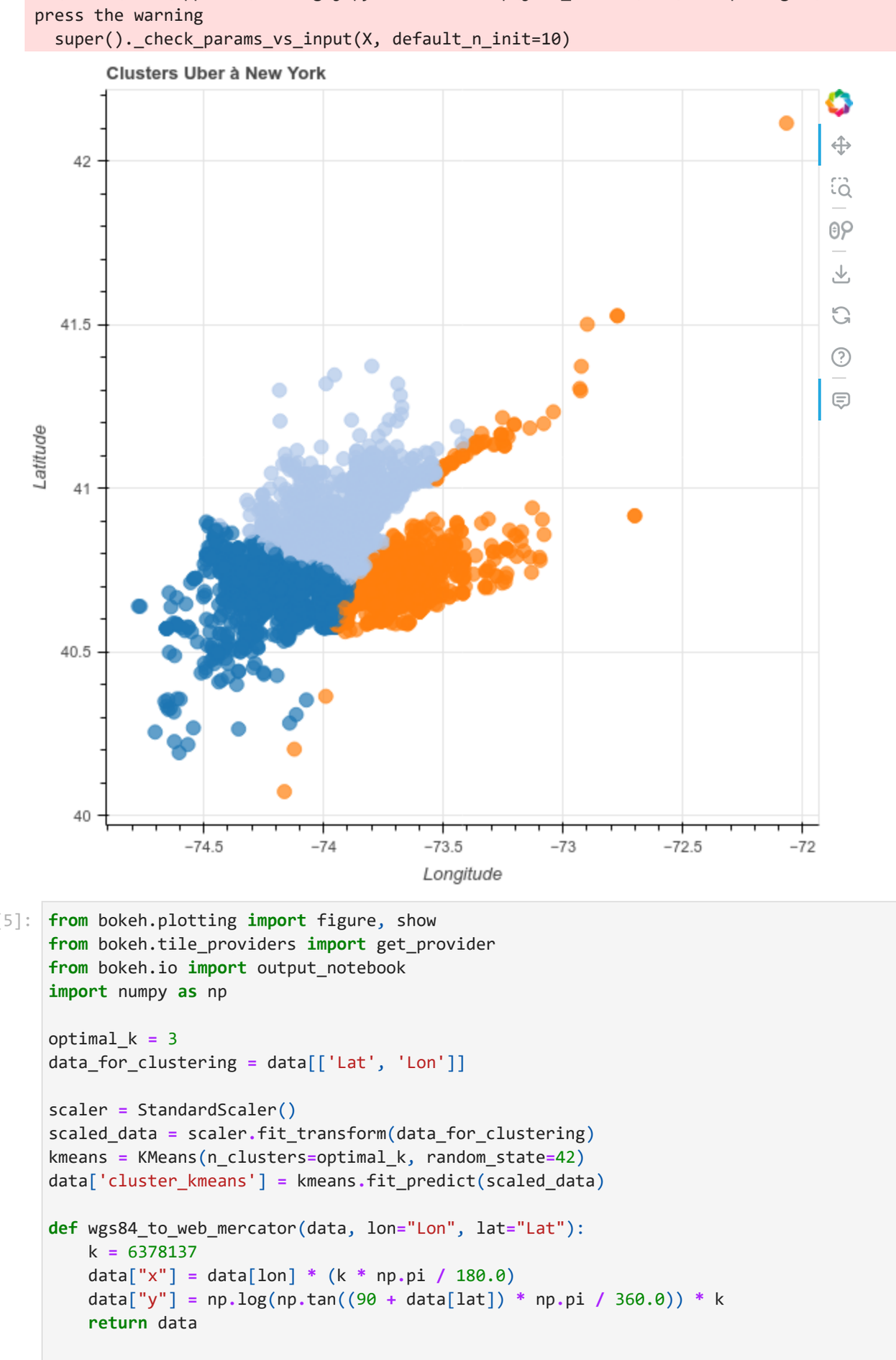
3_ Visualisation des Clusters Uber à New York avec Bokeh

Le code ci-dessous utilise la bibliothèque Bokeh pour créer une visualisation interactive des clusters Uber à New York.

In [4]:
```
from bokeh.io import output_notebook, show
from bokeh.plotting import figure
from bokeh.models import HoverTool, ColumnDataSource
from bokeh.palettes import Category20
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

output_notebook()

data_for_clustering = data[['Lat', 'Lon']]

scaler = StandardScaler()
scaled_data = scaler.fit_transform(data_for_clustering)

optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
data['Cluster_KMeans'] = kmeans.fit_predict(scaled_data)
colors = Category20[optimal_k]
data['Color'] = data['Cluster_KMeans'].map(lambda cluster: colors[cluster])

source = ColumnDataSource(data)

p = figure(title="Clusters Uber à New York", x_axis_label='Longitude', y_axis_label='Latitude')

p.scatter(x='Lon', y='Lat', size=10, color='Color', alpha=0.7, source=source)

hover = HoverTool()
hover.tooltips = [("Cluster", "@Cluster_KMeans"), ("Latitude", "@Lat"), ("Longitude", "@Lon")]
p.add_tools(hover)

show(p)
```

Loading BokehJS ...

```
C:\Users\Dell\AppData\Roaming\jupyterlab-desktop\jlab_server\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to sup
press the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

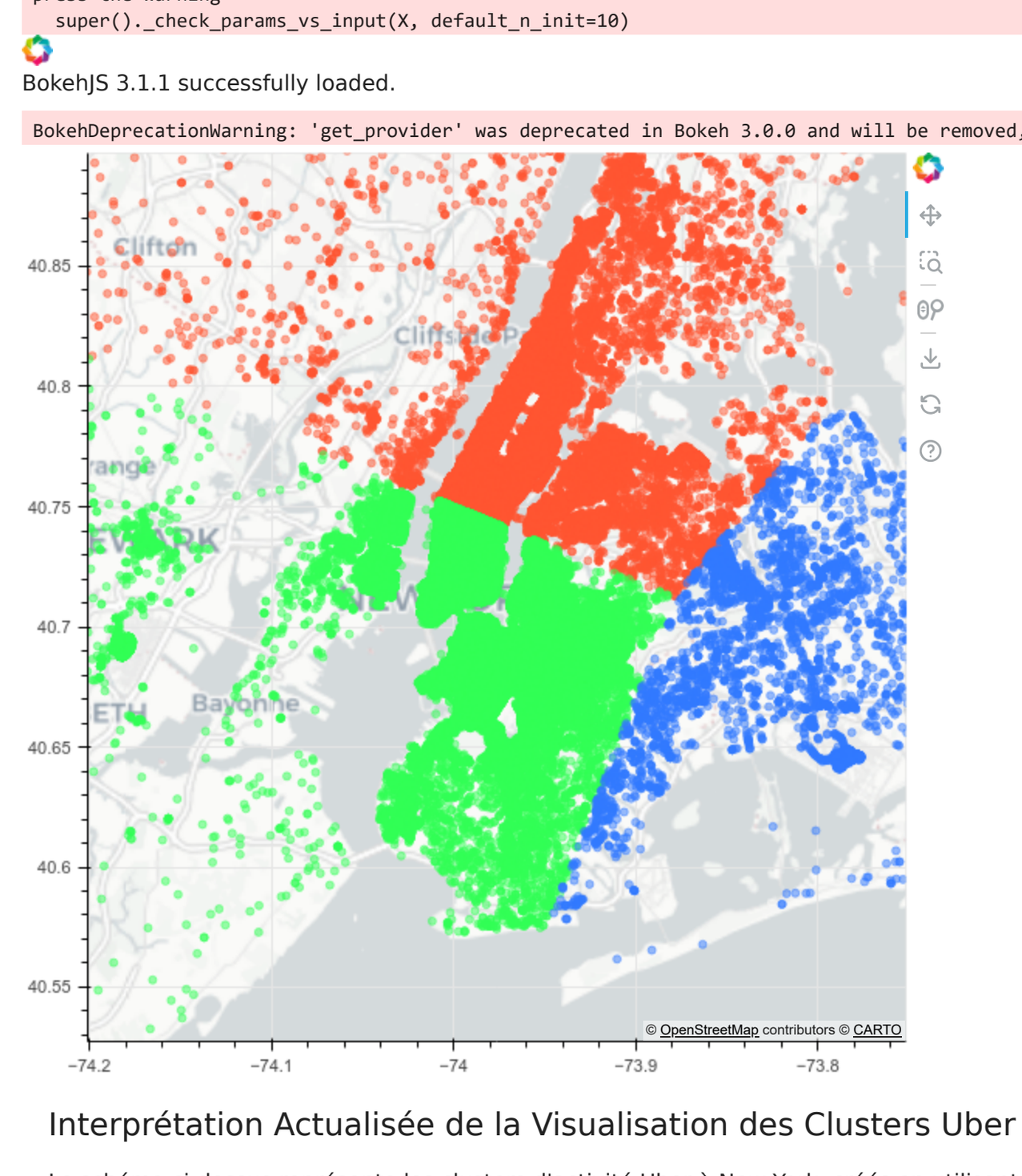


In [5]:
```
from bokeh.plotting import figure, show
from bokeh.tile_providers import get_provider
from bokeh.io import output_notebook
import numpy as np

optimal_k = 3
data_for_clustering = data[['Lat', 'Lon']]

scaler = StandardScaler()
scaled_data = scaler.fit_transform(data_for_clustering)
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
data['cluster_kmeans'] = kmeans.fit_predict(scaled_data)

def wgs84_to_web_mercator(data, lon="Lon", lat="Lat"):
    k = 6378137
    data["x"] = data[lon] * (k * np.pi / 180.0)
    data["y"] = np.log(np.tan((90 + data[lat]) * np.pi / 360.0)) * k
    return data

data = wgs84_to_web_mercator(data)

output_notebook()
tile_provider = get_provider('CARTODBPOSITRON')
p = figure(x_range=(-8260000, -8210000), y_range=(4950000, 4990000), x_axis_type="mercator", y_axis_type="mercator")
p.add_tile(tile_provider)

colors = ['#FF5733', '#33FF57', '#337AFF']  # Couleurs pour chaque cluster
for cluster_id, color in zip(data['cluster_kmeans'].unique(), colors):
    cluster_data = data[data['cluster_kmeans'] == cluster_id]
    p.circle(x='x', y='y', size=5, color=color, alpha=0.5, source=ColumnDataSource(cluster_data))

show(p)
```

```
BokehDeprecationWarning: 'tile_providers module' was deprecated in Bokeh 3.0.0 and will be removed, use 'add_tile directly' instead.
C:\Users\Dell\AppData\Roaming\jupyterlab-desktop\jlab_server\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to sup
press the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

BokehJS 3.1.1 successfully loaded.

```
BokehDeprecationWarning: 'get_provider' was deprecated in Bokeh 3.0.0 and will be removed, use 'add_tile directly' instead.
```



Interprétation Actualisée de la Visualisation des Clusters Uber à New York

Le schéma ci-dessus représente les clusters d'activité Uber à New York, créés en utilisant l'algorithme KMeans avec un nombre optimal de clusters égal à 7. Voici une mise à jour de l'interprétation :

- **Couleurs de Cluster :** Chaque cluster est identifié par une couleur différente, facilitant la distinction visuelle entre les zones de concentration d'activité.
- **Répartition Géographique :** Les cercles sur la carte représentent les points de pickup Uber, situés en fonction de leur longitude et latitude. La répartition des couleurs indique les zones géographiques où les clusters sont localisés.
- **Concentration de l'Activité :** Les zones où les points sont plus denses suggèrent une concentration plus élevée d'activité Uber. Ces zones peuvent correspondre à des quartiers urbains denses ou à des zones de forte demande.

Appliquer DBSCAN

In [ ]:
```
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=eps_value, min_samples=min_samples_value)
sampled_data['cluster_dbscan'] = dbscan.fit_predict(scaled_data)

# Créez une figure Bokeh
p_dbscan = figure(x_range=(-8260000, -8210000), y_range=(4950000, 4990000), x_axis_type="mercator", y_axis_type="mercator")
p_dbscan.add_tile(tile_provider)

# Affichez les clusters sur la carte
colors_dbscan = ['#FF5733', '#33FF57', '#337AFF']
for cluster_id, color in zip(sampled_data['cluster_dbscan'].unique(), colors_dbscan):
    cluster_data_dbscan = sampled_data[sampled_data['cluster_dbscan'] == cluster_id]
    p_dbscan.circle(x='x', y='y', size=5, color=color, alpha=0.5, source=ColumnDataSource(cluster_data_dbscan))

# Affichez la carte
show(p_dbscan)
```

In [ ]: