ДОМАШНЕЕ ЗАДАНИЕ 2. Анализ поездок посредством Spark DataFrame API

Марчук Иван ИУ6-31М

========== **Задача 1** ==========

1. определите для каждой станции количество начала поездок и количество завершения поездок
2. сопоставьте станции с кварталами города (zones) и определите суммы количества начала и завершения для каждого квартала
3. выведите по убыванию количества поездок и
4. отобразите в виде картограмм (Choropleth).

In [16]:
```
# зависимости из дз1
import os

# пути к Java и Spark
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/home/ubuntu/_practice/spark-3.5.4-bin-hadoop3"
os.environ["PATH"] += os.pathsep + os.path.join(os.environ["SPARK_HOME"], "bin")

import findspark
findspark.init()

import pyspark
print(pyspark.__version__)

from pyspark import SparkContext, SparkConf
```

```
3.5.4
```

In [17]:
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, count
import geopandas as gpd
import json
from shapely.geometry import Point, shape

# Инициализация Spark-сессии
spark = SparkSession.builder.appName("Bikes Marchuk").getOrCreate()

# Путь к данным
bikeshare_path = "hdfs:///dataset/201902-citibike-tripdata.csv"
zones_path = "hdfs:///dataset/NYC-Taxi-Zones.geojson"

# ==== Загрузка данных ====

# Загрузка поездок
bikeshare_df = spark.read.csv(bikeshare_path, header=True, inferSchema=True)

# Загрузка GeoJSON из HDFS
zones_rdd = spark.sparkContext.textFile(zones_path)
zones_json_str = zones_rdd.collect()
zones_json = json.loads("".join(zones_json_str))
zones_gdf = gpd.GeoDataFrame.from_features(zones_json["features"])

# Установка CRS для zones_gdf
zones_gdf.set_crs("EPSG:4326", inplace=True)
```

Out[17]:

	geometry	shape_area	objectid	shape_leng	location_id	zone	borough
0	MULTIPOLYGON (((-74.18445 40.69500, -74.18449 ...	0.0007823067885	1	0.116357453189	1	Newark Airport	EWR
1	MULTIPOLYGON (((-73.82338 40.63899, -73.82277 ...	0.00486634037837	2	0.43346966679	2	Jamaica Bay	Queens
2	MULTIPOLYGON (((-73.84793 40.87134, -73.84725 ...	0.000314414156821	3	0.0843411059012	3	Allerton/Pelham Gardens	Bronx
3	MULTIPOLYGON (((-73.97177 40.72582, -73.97179 ...	0.000111871946192	4	0.0435665270921	4	Alphabet City	Manhattan
4	MULTIPOLYGON (((-74.17422 40.56257, -74.17349 ...	0.000497957489363	5	0.0921464898574	5	Arden Heights	Staten Island
...	...	...	...	...	...	...	...
258	MULTIPOLYGON (((-73.95834 40.71331, -73.95681 ...	0.000168611097013	256	0.0679149669603	256	Williamsburg (South Side)	Brooklyn
259	MULTIPOLYGON (((-73.85107 40.91037, -73.85207 ...	0.000394552487366	259	0.126750305191	259	Woodlawn/Wakefield	Bronx
260	MULTIPOLYGON (((-73.90175 40.76078, -73.90147 ...	0.000422345326907	260	0.133514154636	260	Woodside	Queens
261	MULTIPOLYGON (((-74.01333 40.70503, -74.01327 ...	0.0000343423231652	261	0.0271204563616	261	World Trade Center	Manhattan
262	MULTIPOLYGON (((-73.94383 40.78286, -73.94376 ...	0.000122330270966	262	0.0490636231541	262	Yorkville East	Manhattan

263 rows × 7 columns

In [18]:
```
print("1. Количество начала и завершений поездок для каждой станции:")

# Подсчет начатых поездок
start_trip_counts = (
    bikeshare_df.groupBy("start station id", "start station name")
    .agg(count("*").alias("start_trip_count"))
)
print("  Количество начатых поездок:")
start_trip_counts.show(truncate=False)

# Подсчет завершенных поездок
end_trip_counts = (
    bikeshare_df.groupBy("end station id", "end station name")
    .agg(count("*").alias("end_trip_count"))
)
```

```
print("  Количество завершенных поездок:")
end_trip_counts.show(truncate=False)
```

```
1. Количество начала и завершений поездок для каждой станции:
  Количество начатых поездок:
```

```
+----------------+----------------------------+----------------+
|start station id|start station name          |start_trip_count|
+----------------+----------------------------+----------------+
|432             |E 7 St & Avenue A           |3672            |
|458             |11 Ave & W 27 St            |2632            |
|372             |Franklin Ave & Myrtle Ave   |240             |
|473             |Rivington St & Chrystie St  |1422            |
|3602            |31 Ave & 34 St              |347             |
|3092            |Berry St & N 8 St           |1482            |
|377             |6 Ave & Canal St            |2099            |
|3078            |Broadway & Roebling St      |1291            |
|3070            |McKibbin St & Manhattan Ave |203             |
|439             |E 4 St & 2 Ave              |2885            |
|426             |West St & Chambers St       |3256            |
|164             |E 47 St & 2 Ave             |1862            |
|363             |West Thames St              |1631            |
|344             |Monroe St & Bedford Ave     |761             |
|274             |Lafayette Ave & Fort Greene Pl|862           |
|3395            |Henry St & W 9 St           |107             |
|3565            |36 Ave & 10 St              |138             |
|3075            |Division Ave & Marcy Ave    |241             |
|408             |Market St & Cherry St       |946             |
|3453            |Devoe St & Lorimer St       |720             |
+----------------+----------------------------+----------------+
only showing top 20 rows
```

```
  Количество завершенных поездок:
+--------------+----------------------------+--------------+
|end station id|end station name            |end_trip_count|
+--------------+----------------------------+--------------+
|3602          |31 Ave & 34 St              |341           |
|473           |Rivington St & Chrystie St  |1448          |
|458           |11 Ave & W 27 St            |2677          |
|432           |E 7 St & Avenue A           |3310          |
|372           |Franklin Ave & Myrtle Ave   |221           |
|3092          |Berry St & N 8 St           |1497          |
|377           |6 Ave & Canal St            |2899          |
|3078          |Broadway & Roebling St      |1214          |
|3070          |McKibbin St & Manhattan Ave |202           |
|363           |West Thames St              |1640          |
|439           |E 4 St & 2 Ave              |2917          |
|164           |E 47 St & 2 Ave             |1860          |
|426           |West St & Chambers St       |3377          |
|344           |Monroe St & Bedford Ave     |686           |
|274           |Lafayette Ave & Fort Greene Pl|1050        |
|3395          |Henry St & W 9 St           |107           |
|3565          |36 Ave & 10 St              |138           |
|3075          |Division Ave & Marcy Ave    |226           |
|3726          |Center Blvd & 51 Ave        |262           |
|3453          |Devoe St & Lorimer St       |700           |
+--------------+----------------------------+--------------+
only showing top 20 rows
```

In [19]:

```
# 1.2 Сопоставление станций с кварталами
print("1.2 Сопоставление станций с кварталами и подсчет суммарных поездок:")

# Преобразование станций в GeoDataFrame
stations = bikeshare_df.select(
    "start station id", "start station name", "start station latitude", "start station longitude"
).distinct()

stations_gdf = gpd.GeoDataFrame(
    stations.toPandas(),
    geometry=gpd.points_from_xy(
        stations.toPandas()["start station longitude"],
        stations.toPandas()["start station latitude"],
    ),
    crs="EPSG:4326",
)

# Пространственное объединение
stations_with_zones = gpd.sjoin(stations_gdf, zones_gdf, how="left", op="intersects")

# Преобразование геометрии в строку WKT для совместимости с Spark
stations_with_zones["geometry"] = stations_with_zones["geometry"].apply(lambda geom: geom.wkt if geom else None)

# Конвертация результата обратно в Spark DataFrame
stations_with_zones_spark = spark.createDataFrame(stations_with_zones.drop(columns=["geometry"]))

# Объединение с подсчетом начальных и завершенных поездок
stations_with_counts = (
    stations_with_zones_spark
    .join(start_trip_counts, stations_with_zones_spark["start station id"] == start_trip_counts["start station id"], "left")
    .join(end_trip_counts, stations_with_zones_spark["start station id"] == end_trip_counts["end station id"], "left")
)

# Подсчет начальных и завершенных поездок для каждого квартала
zone_trip_counts = (
    stations_with_counts.groupBy("zone")
    .agg(
        count("start_trip_count").alias("total_start_trips"),
        count("end_trip_count").alias("total_end_trips")
```

```
    )
)

zone_trip_counts.show(truncate=False)
```

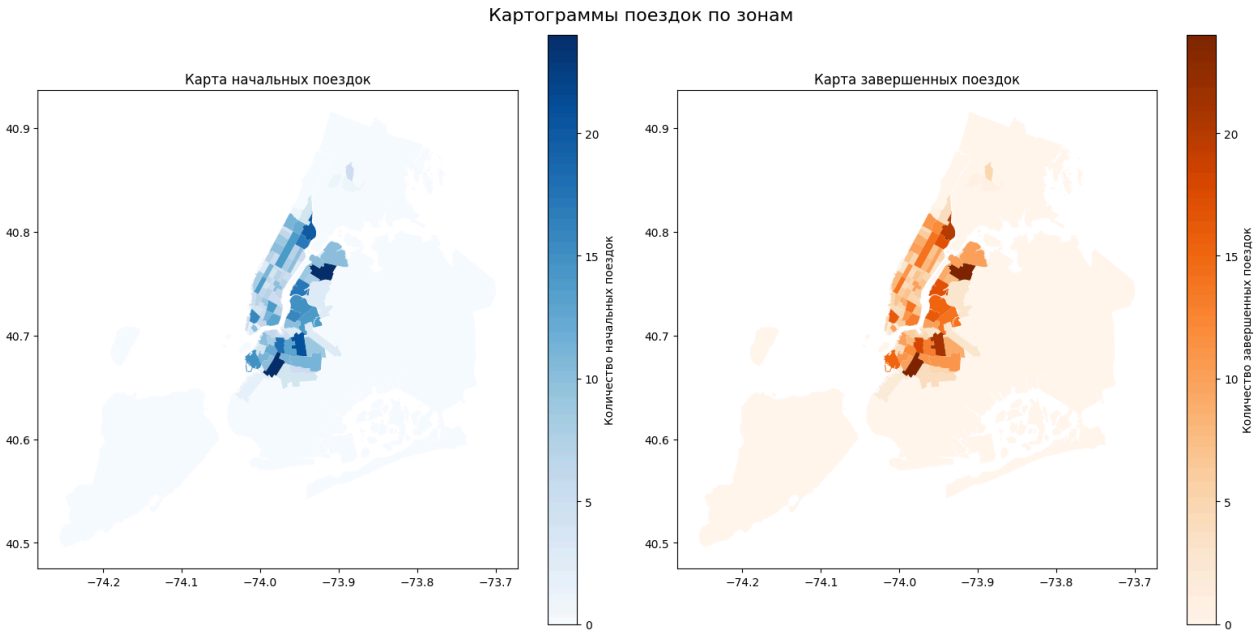
```
1.2 Сопоставление станций с кварталами и подсчет суммарных поездок:
+------------------------------+-----------------+---------------+
|zone                          |total_start_trips|total_end_trips|
+------------------------------+-----------------+---------------+
|Mount Hope                    |1                |1              |
|Upper East Side South         |8                |8              |
|TriBeCa/Civic Center          |16               |16             |
|Yorkville West                |10               |10             |
|Stuy Town/Peter Cooper Village|5                |5              |
|Bushwick South                |3                |3              |
|Old Astoria                   |10               |10             |
|Clinton Hill                  |13               |13             |
|Stuyvesant Heights            |9                |9              |
|Upper West Side North         |9                |9              |
|East Harlem South             |17               |17             |
|Crown Heights North           |11               |11             |
|Hudson Sq                     |4                |4              |
|Lenox Hill East               |3                |3              |
|Upper East Side North         |7                |7              |
|Columbia Street               |4                |4              |
|UN/Turtle Bay South           |7                |7              |
|Prospect Park                 |4                |4              |
|Long Island City/Hunters Point|17               |17             |
|Union Sq                      |7                |7              |
+------------------------------+-----------------+---------------+
only showing top 20 rows
```

In [20]:

```
import matplotlib.pyplot as plt

# 1.3 Вывод данных в порядке убывания и создание картограммы

# Сортировка по количеству поездок
zone_trip_counts_sorted = (
    zone_trip_counts.orderBy(col("total_start_trips").desc())
)

print("1.3 Количество поездок по кварталам в порядке убывания:")
zone_trip_counts_sorted.show(truncate=False)

# Подготовка данных
# Преобразуем Spark DataFrame обратно в Pandas DataFrame для интеграции с GeoPandas
zone_trip_counts_pandas = zone_trip_counts_sorted.toPandas()

# Объединение данных с GeoDataFrame
zones_gdf["total_start_trips"] = zones_gdf["zone"].map(
    zone_trip_counts_pandas.set_index("zone")["total_start_trips"].to_dict()
)

zones_gdf["total_end_trips"] = zones_gdf["zone"].map(
    zone_trip_counts_pandas.set_index("zone")["total_end_trips"].to_dict()
)

# Заменяем NaN на 0 для правильного отображения
zones_gdf["total_start_trips"] = zones_gdf["total_start_trips"].fillna(0)
zones_gdf["total_end_trips"] = zones_gdf["total_end_trips"].fillna(0)

# Создание картограммы
fig, axes = plt.subplots(1, 2, figsize=(16, 8))

# Карта начальных поездок
zones_gdf.plot(
    column="total_start_trips",
    cmap="Blues",
    legend=True,
    legend_kwds={"label": "Количество начальных поездок"},
    ax=axes[0]
)
axes[0].set_title("Карта начальных поездок")

# Карта завершенных поездок
zones_gdf.plot(
    column="total_end_trips",
    cmap="Oranges",
    legend=True,
    legend_kwds={"label": "Количество завершенных поездок"},
    ax=axes[1]
)
axes[1].set_title("Карта завершенных поездок")

# Общий заголовок
plt.suptitle("Картограммы поездок по зонам", fontsize=16)

# Сохранение или отображение карты
plt.tight_layout()
plt.show()
```

```
1.3 Количество поездок по кварталам в порядке убывания:
```

```
+------------------------------+-----------------+---------------+
|zone                          |total_start_trips|total_end_trips|
+------------------------------+-----------------+---------------+
|Astoria                       |24               |24             |
|Park Slope                    |24               |24             |
|Bedford                       |21               |21             |
|East Harlem North             |20               |20             |
|Fort Greene                   |18               |18             |
|Long Island City/Hunters Point|17               |17             |
|East Harlem South             |17               |17             |
|TriBeCa/Civic Center          |16               |16             |
|Williamsburg (North Side)     |16               |16             |
|Red Hook                      |15               |15             |
|Greenpoint                    |15               |15             |
|East Village                  |14               |14             |
|Central Park                  |14               |14             |
|East Williamsburg             |14               |14             |
|East Chelsea                  |14               |14             |
|Clinton Hill                  |13               |13             |
|Two Bridges/Seward Park       |13               |13             |
|Central Harlem                |12               |12             |
|Boerum Hill                   |12               |12             |
|Clinton East                  |11               |11             |
+------------------------------+-----------------+---------------+
only showing top 20 rows
```



.

.

.

.

========== Задача 2 ==========

1. оцените дистанцию поездок (в метрах) на основе координат начальной и конечной станций (без учета поездок, завершившихся там же где и начались)
2. выведите максимальное, среднее значение, стандартное отклонение и медиан

In [21]:
```
from pyspark.sql.functions import col, when, lit, sqrt, avg, stddev, expr
from geopy.distance import geodesic
from pyspark.sql.functions import udf
from pyspark.sql.types import DoubleType


# 2.1 Оценка дистанции поездок
print("2.1 Оценка дистанции поездок:")

# Функция для расчета расстояния
def calculate_distance(lat1, lon1, lat2, lon2):
    if lat1 is None or lon1 is None or lat2 is None or lon2 is None:
        return None
    return geodesic((lat1, lon1), (lat2, lon2)).meters

# Регистрация пользовательской функции (UDF)
distance_udf = udf(calculate_distance, DoubleType())

# Добавление колонки с дистанцией
bikeshare_df = bikeshare_df.withColumn(
    "distance_meters",
    when(
        (col("start station latitude") != col("end station latitude")) |
        (col("start station longitude") != col("end station longitude")),
        distance_udf(
            col("start station latitude"),
            col("start station longitude"),
```

```
            col("end station latitude"),
            col("end station longitude")
        )
    ).otherwise(lit(None))
)

# Фильтрация поездок, завершившихся там же, где начались
bikeshare_df_filtered = bikeshare_df.filter(col("distance_meters").isNotNull())
bikeshare_df_filtered.show(5, truncate=False)
```

2.1 Оценка дистанции поездок:

```
[Stage 59:>                                                 (0 + 1) / 1]
+-----------+----------------------+----------------------+---------------+------------------------+---------------------+------------------
----+-------------+--------------------+-------------------+-------------------+------+----------+----------+------+-----------------+
|tripduration|starttime             |stoptime              |start station id|start station name      |start station latitude|start station longi
tude|end station id|end station name      |end station latitude|end station longitude|bikeid|usertype  |birth year|gender|distance_meters   |
+-----------+----------------------+----------------------+---------------+------------------------+---------------------+------------------
----+-------------+--------------------+-------------------+-------------------+------+----------+----------+------+-----------------+
|219         |2019-02-01 00:00:06.257|2019-02-01 00:03:46.109|3494           |E 115 St & Lexington Ave|40.797911            |-73.9423
|3501          |E 118 St & Madison Ave|40.8014866          |-73.9442507        |33450 |Subscriber|1989      |1     |429.8420597898003 |
|143         |2019-02-01 00:00:28.032|2019-02-01 00:02:51.746|438            |St Marks Pl & 1 Ave     |40.72779126          |-73.98564945
|236           |St Marks Pl & 2 Ave   |40.7284186          |-73.98713956       |25626 |Subscriber|1990      |1     |143.87545219406928|
|296         |2019-02-01 00:01:13.987|2019-02-01 00:06:10.734|3571           |Bedford Ave & Bergen St |40.676368            |-73.952918
|3549          |Grand Ave & Bergen St |40.678045           |-73.962408         |35568 |Subscriber|1987      |1     |823.6544611342246 |
|478         |2019-02-01 00:01:14.152|2019-02-01 00:09:12.787|167            |E 39 St & 3 Ave         |40.7489006           |-73.97604882
|477           |W 41 St & 8 Ave       |40.75640548         |-73.9900262        |25045 |Subscriber|1964      |2     |1444.9419911022665|
|225         |2019-02-01 00:01:49.341|2019-02-01 00:05:34.498|3458           |W 55 St & 6 Ave         |40.76309387270797    |-73.9783501625061
|3443          |W 52 St & 6 Ave       |40.76132983124814   |-73.97982001304626 |34006 |Subscriber|1979      |1     |231.90139521523992|
+-----------+----------------------+----------------------+---------------+------------------------+---------------------+------------------
----+-------------+--------------------+-------------------+-------------------+------+----------+----------+------+-----------------+
only showing top 5 rows
```

In [22]:
```
# Более красивый вывод
# Преобразование DataFrame -> PySpark в Pandas
filtered_pandas_df = bikeshare_df_filtered.toPandas()
# Отображение таблицы
from IPython.display import display
display(filtered_pandas_df.head(10)) # первые 10
```

	tripduration	starttime	stoptime	start station id	start station name	start station latitude	start station longitude	end station id	end station name	end station latitude	end station longitude	bikeid	usertype	birth year	gender
0	219	2019-02-01 00:00:06.257	2019-02-01 00:03:46.109	3494	E 115 St & Lexington Ave	40.797911	-73.942300	3501	E 118 St & Madison Ave	40.801487	-73.944251	33450	Subscriber	1989	1
1	143	2019-02-01 00:00:28.032	2019-02-01 00:02:51.746	438	St Marks Pl & 1 Ave	40.727791	-73.985649	236	St Marks Pl & 2 Ave	40.728419	-73.987140	25626	Subscriber	1990	1
2	296	2019-02-01 00:01:13.987	2019-02-01 00:06:10.734	3571	Bedford Ave & Bergen St	40.676368	-73.952918	3549	Grand Ave & Bergen St	40.678045	-73.962408	35568	Subscriber	1987	1
3	478	2019-02-01 00:01:14.152	2019-02-01 00:09:12.787	167	E 39 St & 3 Ave	40.748901	-73.976049	477	W 41 St & 8 Ave	40.756405	-73.990026	25045	Subscriber	1964	2
4	225	2019-02-01 00:01:49.341	2019-02-01 00:05:34.498	3458	W 55 St & 6 Ave	40.763094	-73.978350	3443	W 52 St & 6 Ave	40.761330	-73.979820	34006	Subscriber	1979	1
5	457	2019-02-01 00:02:04.001	2019-02-01 00:09:41.564	3078	Broadway & Roebling St	40.709248	-73.960631	3016	Kent Ave & N 7 St	40.720368	-73.961651	33858	Subscriber	1981	2
6	175	2019-02-01 00:02:12.296	2019-02-01 00:05:08.093	411	E 6 St & Avenue D	40.722281	-73.976687	317	E 6 St & Avenue B	40.724537	-73.981854	24839	Subscriber	1995	1
7	248	2019-02-01 00:02:13.045	2019-02-01 00:06:21.396	3628	Lenox Ave & W 117 St	40.802557	-73.949078	3506	Lexington Ave & E 120 St	40.801307	-73.939817	31798	Subscriber	1991	1
8	541	2019-02-01 00:02:28.075	2019-02-01 00:11:29.549	293	Lafayette St & E 8 St	40.730207	-73.991026	383	Greenwich Ave & Charles St	40.735238	-74.000271	32175	Subscriber	1981	0
9	406	2019-02-01 00:03:06.542	2019-02-01 00:09:53.431	380	W 4 St & 7 Ave S	40.734011	-74.002939	3263	Cooper Square & Astor Pl	40.729515	-73.990753	27688	Subscriber	1973	2

In [23]:
```
# 2.2 Вычисление статистики
distance_stats = bikeshare_df_filtered.select(
    expr("percentile_approx(distance_meters, 0.5)").alias("median_distance"),
    avg("distance_meters").alias("mean_distance"),
    stddev("distance_meters").alias("stddev_distance"),
    expr("max(distance_meters)").alias("max_distance")
)

print("2.2 Статистика дистанции поездок:")
distance_stats.show(truncate=False)
```

2.2 Статистика дистанции поездок:

```
[Stage 61:=============================================>             (3 + 1) / 4]
```

```
+------------------+-----------------+------------------+------------------+
|median_distance   |mean_distance    |stddev_distance   |max_distance      |
+------------------+-----------------+------------------+------------------+
|1263.7084978119299|1640.599536747154|1288.0338886204604|15306.495948871732|
+------------------+-----------------+------------------+------------------+
```

.

.

.

.

========== **Задача 3** ==========

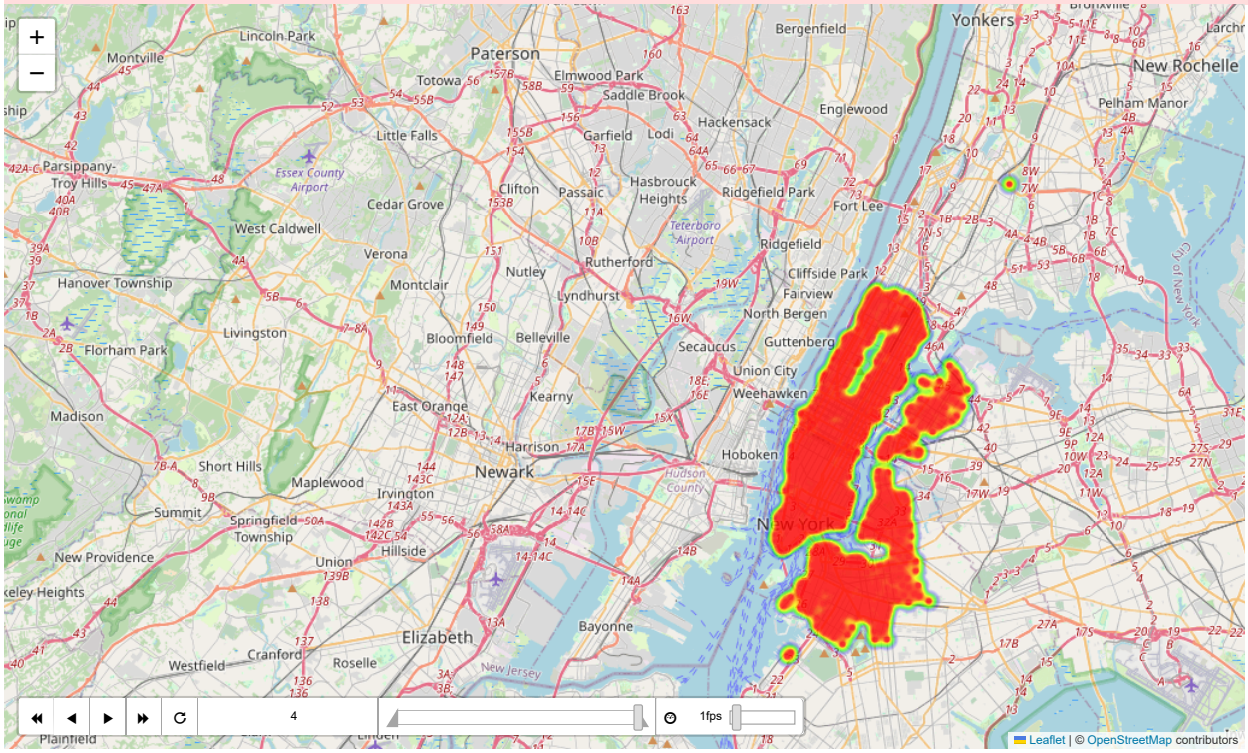
1. определите для каждой станции среднее количество начала поездок и количество завершения поездок:
   - в день
   - утром (06:00-11:59), днем (12:00-17:59), вечером (18:00-23:59), ночью (00:00-05:59)
   - в среду и в воскресенье по временным диапазонам (см. выше)
2. отобразите полученные данные для второго случая в виде тепловой временной карты (HeatMapWithTime)

In [27]:
```
from pyspark.sql.functions import col, avg, count, date_format, hour, dayofweek, when, expr
import folium
from folium.plugins import HeatMapWithTime

# 3.1 Расчет среднего количества поездок

# Добавление временных колонок
bikeshare_df = bikeshare_df.withColumn("day_of_week", dayofweek(col("starttime")))  # 1=Воскресенье, ..., 7=Суббота
bikeshare_df = bikeshare_df.withColumn("hour", hour(col("starttime")))

# Создание временных диапазонов
bikeshare_df = bikeshare_df.withColumn(
    "time_period",
    when((col("hour") >= 6) & (col("hour") <= 11), "Утро")
    .when((col("hour") >= 12) & (col("hour") <= 17), "День")
    .when((col("hour") >= 18) & (col("hour") <= 23), "Вечер")
    .otherwise("Ночь")
)

# Фильтрация данных для среды и воскресенья
filtered_df = bikeshare_df.filter((col("day_of_week") == 4) | (col("day_of_week") == 1))  # 4=Среда, 1=Воскресенье

# Подсчет среднего количества поездок
trip_stats = (
    filtered_df.groupBy("start station id", "start station name", "day_of_week", "time_period")
    .agg(
        count("*").alias("total_trips"),
        avg("distance_meters").alias("Дистанция")
    )
    .orderBy("start station id", "day_of_week", "time_period")
)

print("3.1 Среднее количество поездок:")
print("1=Воскресенье, 4=Среда")
trip_stats.show(truncate=False)
```

```
3.1 Среднее количество поездок:
1=Воскресенье, 4=Среда
[Stage 73:=============================================>             (3 + 1) / 4]
+---------------+--------------------------+-----------+-----------+-----------+------------------+
|start station id|start station name        |day_of_week|time_period|total_trips|Дистанция         |
+---------------+--------------------------+-----------+-----------+-----------+------------------+
|119             |Park Ave & St Edwards St  |1          |Вечер      |2          |813.3457033527137 |
|119             |Park Ave & St Edwards St  |1          |День       |7          |1471.8290642064896|
|119             |Park Ave & St Edwards St  |1          |Ночь       |1          |1491.6851112021525|
|119             |Park Ave & St Edwards St  |1          |Утро       |3          |518.9387919643831 |
|119             |Park Ave & St Edwards St  |4          |Вечер      |3          |1093.42596367486  |
|119             |Park Ave & St Edwards St  |4          |День       |10         |1426.3190034648192|
|119             |Park Ave & St Edwards St  |4          |Ночь       |1          |1527.8200167787124|
|119             |Park Ave & St Edwards St  |4          |Утро       |22         |1242.6003950323684|
|120             |Lexington Ave & Classon Ave|1          |Вечер      |19         |1474.846349780349 |
|120             |Lexington Ave & Classon Ave|1          |День       |49         |1730.5563719955398|
|120             |Lexington Ave & Classon Ave|1          |Ночь       |6          |1478.8049437030752|
|120             |Lexington Ave & Classon Ave|1          |Утро       |24         |1727.5890322854127|
|120             |Lexington Ave & Classon Ave|4          |Вечер      |10         |1987.4954320306817|
|120             |Lexington Ave & Classon Ave|4          |День       |12         |1402.8329734804454|
|120             |Lexington Ave & Classon Ave|4          |Ночь       |1          |496.51348421299514|
|120             |Lexington Ave & Classon Ave|4          |Утро       |55         |2486.8551478909394|
|127             |Barrow St & Hudson St     |1          |Вечер      |43         |1859.917050294973 |
|127             |Barrow St & Hudson St     |1          |День       |156        |1632.7430715830033|
|127             |Barrow St & Hudson St     |1          |Ночь       |6          |1352.9564939635752|
|127             |Barrow St & Hudson St     |1          |Утро       |55         |1259.6292255001674|
+---------------+--------------------------+-----------+-----------+-----------+------------------+
only showing top 20 rows
```

In [28]:
```
from IPython.display import display, HTML
import folium
from folium.plugins import HeatMapWithTime

# Подготовка данных для временной тепловой карты
heatmap_data = []
```

```
time_periods = ["Утро", "День", "Вечер", "Ночь"]

for period in time_periods:
    # Фильтрация поездок по временным диапазонам
    period_data = filtered_df.filter(col("time_period") == period).select(
        "start station latitude", "start station longitude"
    )
    # Преобразование в формат координат для Folium
    heatmap_data.append(
        period_data.rdd.map(lambda row: [row["start station latitude"], row["start station longitude"]]).collect()
    )

# Координаты центра карты (например, центр Нью-Йорка)
map_center = [40.7128, -74.0060]

# Создание карты
m = folium.Map(location=map_center, zoom_start=12)

# Добавление временной тепловой карты
HeatMapWithTime(heatmap_data, radius=10, auto_play=True, max_opacity=0.8).add_to(m)

# Отображение карты в Jupyter Notebook
map_html = m._repr_html_()
display(HTML(map_html))
```



In [29]:
```
# Остановка SparkSession
spark.stop()
```

In [ ]: